INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




A LLM-Powered Resume Matcher

JÚLIO FERNANDO VIEIRA DA ROCHA
setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

A LLM-Powered Resume Matcher

Júlio Fernando Vieira Da Rocha

A dissertation submitted in fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Data Engineering

Advisor: Dr. Paulo Oliveira
Co-Advisor: Dr. Ivo Pereira
Supervisor: Jorge Batista

Porto, 28 September 2025

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, 28 September 2025

Abstract

Currently job recruitment remains one of the most resource-intensive processes in companies. Especially in industries where speed and agility directly impact competitiveness rate. The increase in online job applications has resulted in companies receiving hundreds or even thousands of job applications hours after publishing the job opening. Which in turn makes manual recruitment processing very inefficient, as such, companies use Applicant Tracking systems to automate these workflows, yet they have a big downside of relying solely in keyword-based filtering. This leads to critical limitations of not being able to capture semantic nuances and results on the exclusion of qualified candidates.

This dissertation proposes a solution that integrates Large Language Models and Retrieval-Augmented Generation to avoid the downsides of the generic Applicant Tracking systems. This project was developed in collaboration with an enterprise. For the purpose of this disseration, the enterprise will be referred with an alias as Xcellence. This company which typically receives more than 800 applications per job opening, and takes up to two months to finish recruitment cycles. This research follows a hybrid methodology that combines Design Science Research with Cross-Industry Standard Process for Data Mining (CRISP-DM). Which enables both theoretical rigor and piratical development. Furthermore, a systematic literature review is conducted using the PRISMA framework.

The developed system implements a three-layered architecture, namely storage, processing and access, and the workflow pipeline integrates five orchestrated phases. The included technologies are OCR, semantic vector representations, and the use of a local LLM to output extracted candidate data as JSON.

Furthermore, this dissertation also brings to light a range of ethical challenges. Such as, designing and implementing privacy mechanisms, to ensure effective privacy safeguards.

In conclusion, the findings demonstrate that the LLM-Powered Resume Matcher can significantly reduce recruitment cycle time, and select the most qualified candidate to a specific job opening, thus representing a viable and ethical evolution in algorithmic recruiting.

Keywords: Large Language Models, Artificial Intelligence, Natural Language Processing, Curriculum Vitae, Ethics, Human Resources

Resumo

Num contexto empresarial, o processo de recrutar profissionais qualificados constitui há várias décadas um dos maiores desafios. Principalmente devido á exponencial crescida da Internet que tornou possível enviar candidaturas via online. Embora este fenómeno contribua para uma maior diversidade de candidatos, também representa um problema para o departamento de recursos humanos nas empresas. Aliás, na grande maioria das situações, uma única vaga pode receber centenas ou mesmo milhares de candidaturas, o qual faz com que a análise manual de todos os candidatos em tempo útil seja efetivamente impossível. Como tal, esta realidade atrasa os processos de recrutamento durante meses, os quais podem prejudicar a empresa, principalmente devido ao facto de prazos de projeto não serem cumpridos.

Neste cenário, os *Applicant Tracking Systems* surgiram como uma solução para auxiliar os recrutadores. Estes sistemas automatizam parte do processo, recolhendo, armazenando e filtrando currículos com base em critérios predefinidos, geralmente dependentes de palavras-chave. Porém, apesar da ampla adoção dos *Applicant Tracking System*, a maioria está limitada por serem softwares genéricos e não softwares personalizados para cada empresa. Assim como o facto destes sistemas não terem a capacidade de realizar uma interpretação semântica dos curriculos dos candidatos. O que leva à rejeição de candidatos qualificados mas com percursos profissionais atípicos, ou com currículos com formatos não convencionais.

Face a estas limitações, esta dissertação propõe e desenvolve um sistema inovador designado LLM-Powered Resume Matcher, que tira partido de Large Language Models em conjunto com técnicas de Retrieval-Augmented Generation. A solução foi concebida para responder a um caso real, em colaboração com uma empresa portuguesa de consultoria em tecnologias de informação, aqui referida como Xcellence. Nesta organização, cada vaga aberta resulta em mais de oitocentas candidaturas, sendo que o ciclo de recrutamento se prolonga frequentemente por dois meses. Tal atraso compromete diretamente a capacidade de resposta aos clientes e gera perdas de competitividade. A pertinência deste projeto reside, portanto, na sua dupla dimensão: por um lado, contribui para o avanço científico no domínio da aplicação de *Large Language Models* na área de recrutamento. Contudo, como este projeto celebrou de uma colaboração empresarial, permitiu respondender a uma necessiade real da empresa Xcellence.

A investigação seguiu uma metodologia híbrida, integrando Design Science Research, responsável por enquadrar a construção e avaliação de artefactos tecnológicos, e o Cross Industry Standard Process for Data Mining, que ofereceu um guia estruturado para o fluxo técnico de desenvolvimento. Para a revisão da literatura, foi adotada a metodologia PRISMA, a qual permitiu uma seleção rigorosa de artigos a partir de repositórios científicos, tais como o ACM e o IEEE. O processo inicial identificou mil seiscentos e dois artigos, reduzidos a quarenta publicações de relevância direta para as questões de investigação definidas. Estas questões procuraram compreender o estado atual da investigação sobre LLMs aplicados em contextos empresariais internos, os desafios e boas práticas na implementação destes modelos.

Também como, a forma como podem contribuir para o parsing de currículos e o panorama atual dos ATS. Esta revisão sistemática estabeleceu uma base teórica sólida para o trabalho prático subsequente.

Do ponto de vista arquitetónico, o sistema proposto foi desenhado em três camadas principais. A camada de armazenamento é responsável pela gestão dos documentos originais, utilizando MinIO para objetos e bases de dados relacionais e vetoriais como SQLite e ChromaDB. A camada de processamento assegura as operações de extração de texto através de OCR com Tesseract, transformação em estruturas JSON recorrendo a LLMs e vetorização semântica com modelos Sentence-Transformers, associando ainda metadados como identificadores de candidato, localização ou competências. Por fim, a camada de acesso disponibiliza a interação com o sistema tanto em interface de linha de comandos como numa interface gráfica desenvolvida em Streamlit, suportada por um backend em FastAPI.

A implementação foi organizada em cinco fases sequenciais. A configuração inicial da base de dados assegura a preparação do ambiente de persistência. A sincronização entre MinIO e SQLite garante a correspondência entre documentos armazenados e registos estruturados. A fase de extração realiza a leitura dos currículos e a transformação dos textos em estruturas organizadas, recorrendo a OCR e LLMs. A fase de vetorização gera representações semânticas tanto dos documentos como dos metadados, o que assegura a proximidade contextual entre descrições de emprego e perfis de candidatos. Por fim, a fase de consulta permite a obtenção de candidatos adequados a cada vaga através de *Retrieval Augmented Generation*, a qual apresenta listas ordenadas de acordo com a relevância dos candidatos. Uma preocupação central foi a eficiência, pelo que o sistema foi concebido para processar apenas novos documentos, ignorando os já tratados, reduzindo significativamente o tempo de execução.

Para efeitos de validação, foram preparados três conjuntos de dados distintos. Um conjunto sintético, gerado com a biblioteca Faker e modelos pré-definidos de experiência académica e profissional. Um conjunto manual, elaborado especificamente para simular casos realistas mas controlados. E um conjunto de currículos reais, devidamente anonimizados para proteção de dados pessoais.

A avaliação do sistema foi conduzida com base em métricas clássicas de recuperação de informação: precisão, recuperação, F1-score e *Normalised Discounted Cumulative Gain*. Estas métricas permitiram comparar as classificações produzidas pelo sistema com a verdade de base fornecida pelos recrutadores da Xcellence. Os resultados demonstraram uma elevada correspondência entre o artefacto desenvolvido e as decisões dos recrutadores por parte da Xcellence, sobretudo em termos das métricas de precisão e nDCG. Estas métricas demonstraram a capacidade do sistema em extrair detalhes semânticos relevantes.

Ademais, além das questões técnicas, este trabalho também teve como objetivo investigar aspectos éticos e jurídicos associados ao uso de inteligência artificial nos processos de recrutamento. A revisão bibliográfica demonstrou que existem riscos reais de enviesamento nos algoritmos, que podem ocorrer em várias etapas dos sistemas. Desde a formulação dos anúncios de emprego, até à escolha final dos candidatos, incluindo também a seleção dos currículos.

Para concluir, o artefacto aqui desenvolvido fornece evidência empírica de que é possível evoluir a àrea de recrutamento com a utilização de LLMs, conciliando inovação, responsabilidade e impacto social positivo.

Agradecimentos

Em primeiro lugar, quero expressar a minha sincera gratidão ao Professor Dr. Alberto Pereira, que sempre me apoiou no planeamento e mostrou-me a importância de organizar devidamente cada etapa do trabalho. A sua simpatia e constante disponibilidade nas fases iniciais da dissertação foram fundamentais para ultrapassar diversas dificuldades ao longo deste percurso.

Em seguida, quero agradecer ao Professor Dr. Paulo Oliveira, o meu orientador. Agradeço por me ter aceite como o seu orientando neste trabalho, agradeço também pela paciência e simpatia demonstradas perante os vários dias em que o meu processo de escrita esteve num ritmo lento, e pelo acompanhamento constante que tornou este caminho muito mais fácil de percorrer.

Ao Professor Dr. Ivo Pereira, o meu co-orientador, ao qual manifesto o meu reconhecimento pelas suas excelentes ideias, simpatia, disponibilidade e pelo contributo essencial na reestruturação desta dissertação, que me levou a estudar um tema mais interessante para este trabalho.

Ao Jorge Batista, o meu gerente na empresa onde realizei esta dissertação em colaboração. Agradeço pela disponibilidade em coordenar todos os aspetos necessários comigo nesta investigação, desde a obtenção de permissões até à disponibilização de documentos e recursos que foram indispensáveis para a realização deste trabalho.

À minha família, deixo um profundo agradecimento pelo apoio incondicional, pelo incentivo constante e por me motivarem a ambicionar sempre mais na minha vida académica e profissional. É verdade que não podemos escolher os nossos pais, mas eu posso dizer que tenho sorte em ter os pais que tenho.

À Direção do Mestrado em Engenharia Informática, nomeadamente à Professora Dra. Isabel Praça, à Dra. Professora Isabel Azevedo e ao Professor Dr. Paulo Oliveira, expresso o meu sincero agradecimento pela constante simpatia e pela forma como sempre me senti respeitado e ouvido em representação de todos os alunos. Apesar dos vossos horários ocupados, sempre me mostraram disponibilidade e interesse em falar comigo, pelo que vos agradeço profundamente.

Agradeço também a todos os meus colegas de Mestrado que confiaram em mim como Representante de todos os alunos do curso. Foi um prazer ajudar-vos no que consegui.

Por fim, agradeço ao meu amigo Osvaldo Silva, com quem partilhei vários anos da vida universitária, inúmeras aulas em comum e projetos realizados lado a lado. Agradeço a amizade e a companhia que tornaram esta jornada muito mais significativa. A todos os meus outros amigos, agradeço a vossa paciência por todas as vezes em que estive preocupado com a dissertação. Obrigado!

Contents

List of Figures

List of Tables

List of Source Code

List of Acronyms

AI	Artificial Intelligence.
ARP	Automated Resume Parsers.
ATS	Applicant Tracking System.
AWS	Amazon Web Services.
BM25	Okapi Best Matching 25.
CNN	Convolutional Neural Networks.
CRISP-DM	Cross Industry Standard Process for Data Mining.
DSR	Design Science Research.
GenAI	Google Generative AI.
HR	Human Resources.
JLMIA	Joint Latent Model on Interview Assessment.
KD	Knowledge Distribution.
kNN	k-Nearest Neighbors.
LLM	Large Language Model.
LSTM	Long Short-Term Memory.
nDCG	Normalized Discounted Cumulative Gain.
NER	Name Entity Recognition.
NLP	Natural Language Processing.
OCR	Optical Character Recognition.
PDF	Portable Document Format.
PRISMA	Preferred Reporting Items for Systematic Review and Meta-Analysis.
RAG	Retrieval-Augmented Generation.
STAR	Situation Task Action Result.
TF-IDF	Term Frequency-Inverse Document Frequency.

Chapter 1

Introduction

The purpose of this chapter is to introduce the core theme of this dissertation, the importance of incorporating a Resume-Matcher System supported by a Large Language Model (LLM), to modernize enterprise recruitment. The research on this topic highlights the relevance of this approach in addressing the challenges faced by the company recruiters, such as Human Resources (HR). Especially in companies that manage high volumes of job applications. In such cases, the *modus operandi* (method) is the utilization of an Applicant Tracking System (ATS). However even though these systems already assist recruiters in the long task of manually filtering each candidate, by automatically scanning the resumes and registering them into a database for further analysis, important data can be missed or not captured from these documents. As such, in order to improve the filtration of large amounts of job application, this solution was created. It is also important to note that the research for this dissertation was carried with a partnership between the author and a company, herein known as *Xcellence*.
Therefore, this dissertation research and solution fit a real use-case to improve this issue of managing and filtering of ever-coming new-candidates. This introductory chapter also presents the methodology chosen for the literature review and also the explanation of the structure for the rest of the document.

1.1 Project Motivation

Despite the popularity of the traditional ATS systems, which are systems that parse candidate resumes to match job description with qualified candidates from job applications. These systems often have the downside or the operational incapability of parsing fine-grained details of the candidate data, such as important skills that could be very meaningful for the job requirements. In other words, these traditional ATS systems overlook good candidates because their resumes do not precisely match the pre-programmed set of keywords defined by the ATS system, this becomes even more common when evaluating international candidates, candidates who changed careers of simply candidates that have unique experiences and with resumes that do not follow the standard parameters. As such, the motivation behind this research is to design a fully-fledged system that can understand resumes, whether or not these are simple or complex the same way a human recruiter could. Such a system can be used with the aid of a LLM, since LLMs are general-purpose text generators. If amplified by semantic-aware retrieval (e.g – understanding the full context of a candidate resume) then ATS can be replaced by a better technology.

Another motivating factor is due to the reason that this project being developed as a solution for *Xcellence*, will help in the betterment of the recruitment process.

This is a positive factor not just for the company in question, but also for each candidate that is interested in joining in as a member. Which in turn allows for *Xcellence* to have the most qualified candidates for each job opening.

1.2 Problem Statement

Currently in the job recruitment world there is a large number of applications that are sent for each job opening, mostly due to the ease of applying online. In the specific context of *Xcellence* they receive on average 800 applications per job opening, and as such, to face this demand the recruitment process typically lasts over two months. It is during this time that recruiters must manage both candidate expectations and project requirements, such as project timelines, if any delay occurs on this process, the company may lose competitive advantage on the marketplace or even have losses due to not being able to keep-up with deadlines. In order to solve this, the majority of companies nowadays currently use ATS as a baseline requirement, due to the fact that analyzing hundreds or thousands of resumes in a short-period of eight weeks can be very demanding. Thus, ATS helps to speed-up this process but not without downsides, in theory it should accelerate recruitment by filtering out unqualified candidates. However, ATS often struggle with technical resumes that are not formatted in a standard way, they fail to recognize small details and cannot grasp semantic context, they are bound to fixed query keyword terms. As such, this research will provide a rigorous approach to solving this problem and develop a custom-fit solution for *Xcellence*.

1.3 Objectives

The primary objective of this dissertation is to design and implement a LLM-Powered Resume Matcher, capable of accelerating recruitment processes and selecting the most qualified candidates. In order to achieve these goals, several objectives are defined, the first is to conduct a State of The Art review to establish the current scientific and technological knowledge about recruitment systems and algorithmic-hiring, and LLMs. The second objective is to establish a rigorous methodological framework to guide this dissertation in a consistent theoretical way but also accompanied by a reproducible technical workflow. Consequently, to design and implement the system architecture, in order to demonstrate the functionality of the system through practical experiments and evaluating it with real metrics and to compare the system outputs with recruiter ground truth. Thus, allowing for the analysis to validate the accuracy and performance of the system on a real-world scenario.

1.4 Methodology

This dissertations was guided by two research types. First for the literature review, the chosen methodology that covers this study was based on the objectives mentioned previously and to ensure that it could tackle all the details explained on the problem statement, Preferred Reporting Items for Systematic Review and Meta-Analysis (PRISMA) was chosen. This methodology provides the possibility of documenting an independent reporting guideline of the knowledge gathered of this study.
Furthermore, for the design and implementation part of this dissertation, the research method chosen was a hybrid between Design Science Research (DSR) (Dresch, Lacerda, and Antunes 2015) and Cross Industry Standard Process for Data Mining (CRISP-DM) (Wirth and Hipp 2000).

For a better grasp of the business understanding and also to have a scientific workflow that can be replicated, the justification for this hybrid methodology will be covered in Chapter 3.

1.5 Structure

This dissertation and the subsequent study thereof, is composed by seven fundamental chapters, to ensure a complete scientific work on the theme of resume-matcher system powered by an LLM, and about the study of this theme.

In Chapter 1, the Introduction, the motivation is presented for the research by addressing the real world challenges that exist about this theme. Consequently, a more in-depth explanation about the problem is provided in the problem statement. Furthermore, the objectives of this research and its subsequent methodology is also detailed in this chapter. Lastly, the planning of the entire dissertation is explained, which is comprised of the initial research and initial study in order to fully understand possible future time constraints.

In Chapter 2, the State of the Art, the existing literature is reviewed, alongside the technological landscape attached to this theme. Surveying information in prior research about applicant tracking systems, natural language processing, large language models, and retrieval-augmented generation. This chapter establishes the theoretical foundation for the study.

In Chapter 3, the Methodology, the research approach for this work is described. It explains the choice of a hybrid methodology consisting in Design Science Research as the main methodology paired with CRISP-DM to structure the technical workflow, this chapter also covers the research context, the process of data collection and preparation.

In Chapter 4, the Analysis & Design, a initial design of the artifact developed in this thesis is created. Moreover, ethical considerations, security measures and orchestration of the system is presented.

In Chapter 5, the Development, the system architecture is presented, it highlights the chosen technologies and why. While also explaining how each component was implemented and integrated to create a functional end-to-end pipeline, followed by an in-depth explanation of the practical and technical implementation of the system.

In Chapter 6 the Evaluation & Discussion, an assessment of system performance is presented. The evaluation objectives, datasets, metrics, experiments are all explained in this chapter. This chapter shows how the system achieved alignment with recruiter decisions in the real-world scenario, confirming the effectiveness of this artifact.

In Chapter 7, the Conclusion, summarizes the problem, methodology, artifact and key findings.

Chapter 2

State of the Art

This chapter explores the current State of the Art on the matter at hand. Thus, the research methodology adopted to study the integration of an LLM system for interview selection in a professional recruiting environment is explained. Consequently, the results obtained are described and a detailed discussion of the common and non-common topics is given. Thus, creating a solid knowledge base to support the development of a solution.

2.1 Background Knowledge

This section presents background knowledge to help understand key concepts and technologies relevant to the research conducted in this thesis. Thus, core principles will be covered such as Artificial Intelligence, Large Language Models, Applicant Tracking Systems, Natural Language Processing, Transformers, and Semantic embeddings.

2.1.1 Artificial Intelligence

Artificial Intelligence (AI) refers to systems or programs that are capable of performing tasks that would usually require a person, such as reasoning or learning. Furthermore, within the context of recruitment, AI has been increasingly used more and more to automate repetitive tasks, from resume parsing to candidate ranking. On the work of Perez-Cerrolaza et al. (2024), AI is defined as *"(...) a set of methods or automated entities that together build, optimize and apply a model so that the system can, for a given set of predefined tasks, compute predictions, recommendations or decisions"*.

2.1.2 Large Language Models

LLMs are deep learning architectures based on transformers, they are trained on a massive corpus of text, with their primary objective being generating context-aware language and also understanding/extracting semantic meaning from their query prompts. Models such as ChatGPT or LLaMa are the state of the art when it comes to resume parsing, unlike NLP models that will be covered ahead in the results section, where these models heavily depend on fixed key-words. Moreover, according to Hachicha et al. (2024), *"LLMs can extract and interpret relevant information, identifying candidates skills, experience, and aspirations, as well as companies requirements and expectations. Thanks to their deep understanding of language, LLMs can tailor recommendations more finely, considering the nuances and subtleties of candidate profiles and placement descriptions"*.

2.1.3 Applicant Tracking Systems

ATS are enterprise tools designed to aid and automate the recruitment process, they collect, store and filter resumes. Typically relying on fixed keyword terms to query the database, to match job descriptions to candidates. However, those fixed keywords sometimes do not pick-up on small details or cannot even understand semantic context, as such, valuable information about the candidates can be lost. Furthermore, AL-Qassem et al. (2023) describes ATS as *"ATS is a software program that manages job advertisements, applications, resumes, and candidate data to automate the recruiting and recruitment process (...) ATS may automate a number of laborious and repetitive hiring procedures, including reviewing resumes, setting up interviews, and sending emails to candidates"*.

2.1.4 Natural Language Processing

Natural Language Processing (NLP) is a subfield from AI that enables machines to understand and generate human language, typically english (but could be another language). Core mechanisms include Tokenization and Name Entity Recognition (NER). For Tokenization, text is broken down into smaller units called "tokens", whereas for NER, text is identified and classified as entities, such as a person, an enterprise, or product. Furthermore, to showcase this, taking into consideration the following sentence – *"John just got a Job at Microsoft!"* John would be classified as a person, and Microsoft would be classified as an enterprise. According to Mohanty et al. (2023) *"Core NLP components (particularly named entity recognition) are used to parse information like names, phone, email, education, talents, languages, and cities"*.

2.1.5 Transformers

Transformers are neural architectures that possess a self-attention mechanism which allows the algorithm to model long-range dependencies in sequences. In a recruiting scenario, transformers are a good way to extract skills in candidate resumes, and are also good for candidate-job matching. Furthermore, according to Guan et al. (2024) *"Transformer-based methods more naturally model sequential textual data and learn global semantic representation using the self-attention mechanism"*.

2.1.6 Semantic embeddings

Semantic models create high-dimensional vector spaces where words that are semantically similar will correspond to other similar words in a nearby space, geometrically. For instance, sentences that describe the same job title "Software Engineer" and "SWE" are mapped to nearby vectors. In a recruiting scenario, embeddings are crucial to pick-up the little nuances and small details. Furthermore, it is important to note that, according to Hourany et al. (2023) *"document-level embeddings that capture semantics. Techniques to map documents to vector representations are numerous and have been used in recruitment for decades"*.

2.2 Research Methodology

The systematic review present in this chapter employs the PRISMA methodology, as mentioned before in the introduction chapter.

This methodology was chosen with the purpose of accurately gathering and analyzing the most relatable scientific contributions to AI integration within job recruitment environments.

2.2.1 Research Questions

In order to understand the implications and intricacies of AI integration within recruitment pipelines, a series of research questions were formulated as indicated by the PRISMA methodology. These research questions can be seen on Table 2.1.

Table 2.1: Research Questions

Identifier	Research Question
RQ1	What is the current state of research in implementing an internal LLM?
RQ2	What are the challenges and best practices of implementing an internal LLM?
RQ3	How does an internal LLM help with resume parsing?
RQ4	What is the current state of Applicant Tracking Systems?

The first research question *"What is the current state of research in implementing an internal LLM?"* was designed in response to the very rapid pace at which LLMs have become more popular. As such, they are being integrated into enterprises, often on-premises due to data security, privacy and compliance, but also commonly seen as hybrid of on-premises and in the cloud.

The second research question *"What are the challenges and best practices of implementing an internal LLM?* was created with the intent of understanding the obstacles that organizations face and what strategies they use to get rid of those obstacles.

The third question *How does an internal LLM help with resume parsing?* is the link between the general discussion of LLM deployment with the problem aimed to be answered in this thesis.

The fourth and final research question *What is the current state of Applicant Tracking Systems?* was created with the objective of understanding current ATS market trends, to investigate and learn existing ATS technologies.

2.2.2 Scientific Repositories

The selection of scientific repositories to undertake the PRISMA methodology was essential in order to conduct this literature review, it was taken into account scientific repositories that possess rules and strictly enforce standards or guidelines for each paper published. The scientific repositories chosen were, ACM and IEEE as seen on Table 2.2.

Table 2.2: Scientific Repositories

Identifier	Repository	URL
SR1	ACM	https://dl.acm.org/
SR2	IEEE	https://www.ieeexplore.ieee.org/

2.2.3 Search Terms

In order to ensure that relevant studies were gathered from the aforementioned scientific repositories, a search term had to be designed and implemented. The search term was created to ensure that all Artificial Intelligence technologies would be included on the retrieved articles, thus the search term found was the combination of the terms on Table 2.3.
More specifically *(("Large Language Model" OR "LLM") AND ("Applicant Tracking System" OR "ATS"))*.

Table 2.3: Search Terms

Domain	**Keywords**
Large Language Models	("Large Language Model" OR "LLM")
Applicant Tracking System	("Applicant Tracking System" OR "ATS")

2.2.4 Inclusion and Exclusion Requirements

In order to define the boundaries of the review, inclusion and exclusion criteria were developed as seen on Table 2.4 and Table 2.5 respectively. These requirements were developed to ensure that this systematic review remains focused on finding answers to the research questions.

Table 2.4: Inclusion Requirements

Identifier	**Inclusion Requirement**
IR1	Papers must present empirical studies, theoretical analyses, or comprehensive reviews that contribute to the understanding of how Large Language Model systems operate.
IR2	Papers must be published in peer-reviewed journals or presented at reputable conferences to ensure quality and rigor of the research.
IR3	Papers must directly address the integration, application, or implications of both Large Language Models and Applicant Tracking Systems.

Table 2.5: Exclusion Requirements

Identifier	**Exclusion Requirement**
ER1	Paper published before 2010, to ensure inclusion of recent research technologies.
ER2	The article is not in English.
ER3	The article is not from a Journal or conference.
ER4	Papers that do not focus on the integration, application or implications of LLMs.
ER5	Multiple versions of the same study or redundant publications reporting on the same findings.
ER6	Papers that do not provide an abstract or summary of their contents, making it difficult to assess their relevance to the research questions.

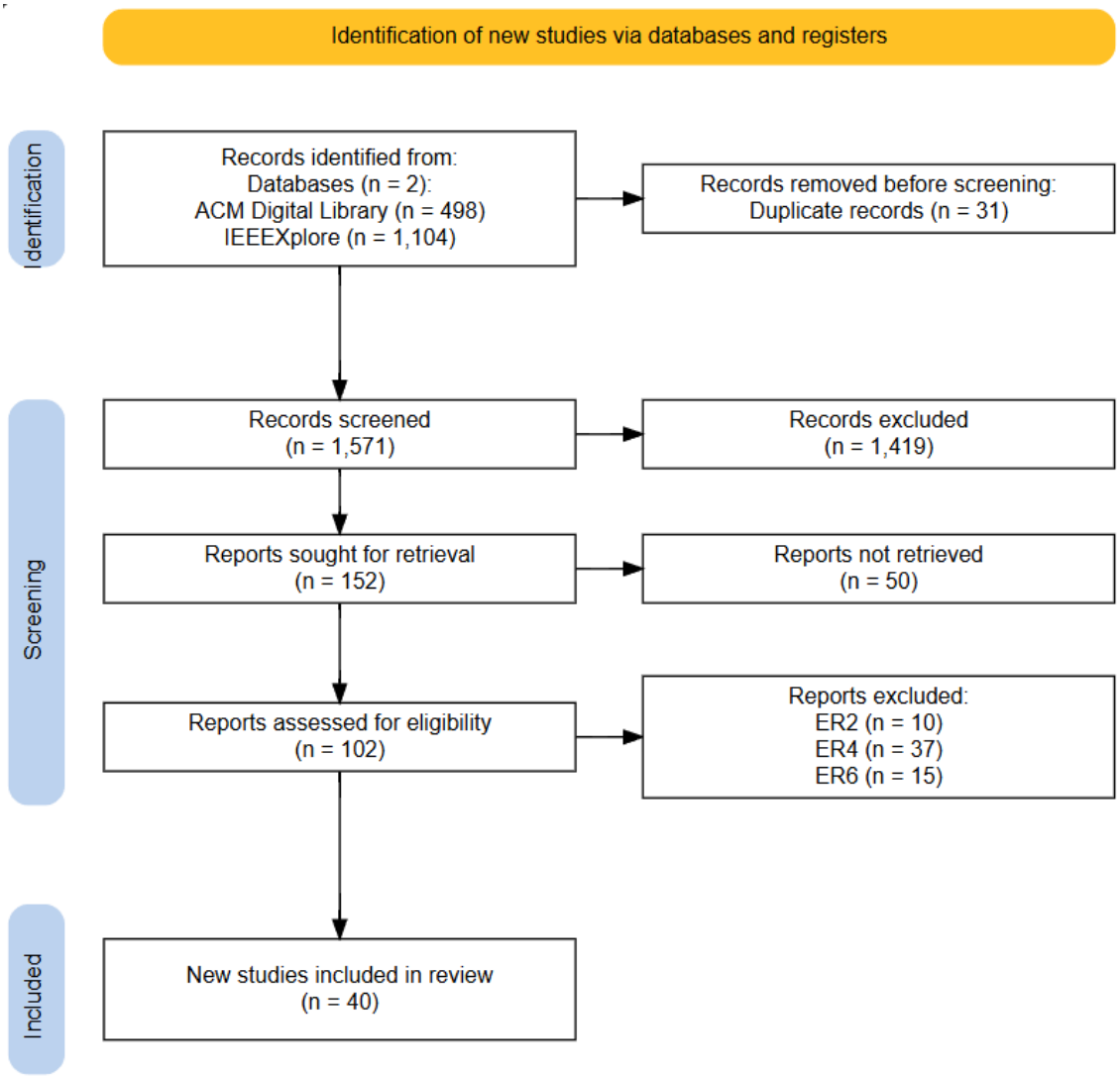


Figure 2.1: PRISMA Diagram with the retrieved articles

2.2.5 Publications Extraction

This section covers the extraction of studies while employing the PRISMA guidelines. Which are composed by the following phases: Identification, Screening, and Included. These phases can be seen on Figure 2.1.

On the first phase, the "Identification" the number of total articles identified was of 1,602. During this search, ER1 was applied in the query of SR1 and SR2 to ensure that all articles were published during or after 2010. Afterwards, a bulk download of all those articles was done, out of which, 53 were removed for being duplicates. Furthermore, during the screening phase 1,419 articles were excluded based on the exclusion requirements mentioned on 2.5. Most of these articles were removed based on ER4, ER2 or ER3. Consequently 152 articles were sought for retrieval, of which 50 were excluded, leaving 102 for eligibility in which 50 were excluded. At last, 40 studies were included in this review.

2.3 Results

This section presents the articles identified through the adopted methodology, PRISMA. With the purpose of documenting all the retrieved publications and explaining the relevance of their selection in relation with the research questions.

2.3.1 Current Research Landscape

The study of evolution of the use of AI in recruiting was segmented into three temporal periods. Earlier research showcases that the main focus of AI in this context was mainly in foundational tasks, in other words, repetitive tasks. Whereas on the most recent research, there are more sophisticated systems, which for instance employ NLP and LLM technologies.

During the initial period which spanned from 2016 until 2019. The research gathered showcased that at this point in time, AI was being used to automate simple tasks. During this time, the researchers mostly focused in how to quantify resume quality, and how to view candidate data. More specifically, studies like ResumeNet Luo et al. (2018) employed neural networks to rank resumes. Whereas other studies, namely, ResumeVis C. Zhang and Wang (2018), introduced visual analytics systems to aid recruiters in understanding resume content. These visual analytics systems used NER to extract semantic-aware context from the resumes. Although these early AI systems demonstrated the potential for automating tasks on the recruitment process, they were limited mostly due to dataset sizes and a lack of consideration for ethical issues such as bias and transparency.

The second temporal period, set in 2020 until 2022, introduced the use of deep learning, specifically Natural Language Processing, such as the work of Shen et al. (2021) which used Topic models to analyze interview transcripts and resumes, which in turn improved fairness and reduced bias due to the fact that these models could analyze and understand the underlying structure of the data. These topic models are statistical models that can find abstract topics in a document. On this work, JLMIA, Neural-JLMIA and R-JLMIA are employed. Of these three models, R-JLMIA yielded the best results, especially when paired with a Long Short-Term Memory (LSTM) layer. The researcher concluded that these models provide a good accurate measurement to see if a certain individual fits into a job opening or not.

On other studies, such as the work of Lappas (2020), the objective shifts from recruiting and is instead focused in predicting career paths, Lappas used a large dataset of 121,313 resumes from IT-related fields. This large dataset paired with machine learning algorithms and word embedding techniques such as Distributed Memory Model, which is a form of NLP, allowed Lappas to conclude that even paired with these technologies, extracting information from resumes is not an easy task, especially due to the fact of the existence of many distinct job titles on the IT field. Due to the fact that, from the previously mentioned 121,313 resumes, Nappas showcases 180,000 distinct job titles.

The third and final temporal period is set from 2023 until 2025. It is here where it can be seen a popularity increase on the use of transformers and LLMs. The LLM technologies show better results than the previously mentioned AI approaches, this can be seen on the work of Venkatakrishnan et al. (2024) where the study aims to showcase that the usage of LLMs in the context of recruiting has better performance than using NLP techniques only. This study used 358 resumes and tested the extraction of resume information with GPT, Davinci, LLaMa, and Flan T5, the LLM with the best result was LLaMa.

The author claims LLaMa took victory due to the fact of it having an advanced architecture and a good use of fine-tuning techniques.

Moreover, while performance of extracting data from resumes has been improved considerably, some researchers emphasize on the importance of fairness and bias mitigation. There are four very relevant papers on this context, such is the case of Fabris et al. (2025) Caton and Haas (2024), Giri et al. (2024), and, Parraga et al. (2025). Notably, it is important to mention that these surveys claim that the core problem is bias reproduction. Moreover, bias can be introduced in Machine Learning and Neural Networks algorithms, due to the fact that these algorithms capture correlations within the data, some are even made too capture complex correlations, such as LSTM. Thus, by not cleaning or filtering the data beforehand, the model can introduce bias into the system, and this bias could have really bad impacts, namely bias of social inequalities, such as race, or gender. In other words, these researches conclude that these systems need to be monitored at all times, and the data has to be cleaned. Furthermore, these researchers also conclude that fairness must be measurable, whether it is group-based fairness or individual fairness.

Last but not least, the work of Giri et al. (2024) proposes that traditional keyword-based resume parsers should be replaced with LLM-driven semantic parsing. If video recruitment exists, then it should be transcribed into audio and consequently parsed into an LLM.

In conclusion, this rapid expansion in this field suggests that recruitment research has moved from discrete tasks such as parsing or scoring, towards multi-modal systems. However, there is still uncertainty in how to balance predictive power with the need for fairness and transparency.

2.3.2 Resume Parsing and Information Extraction

On the studies and researches talked about here previously, there is a need to deal with unstructured and semi-structured resume documents and turn them into structured data. such is the case as seen on ResumeVis C. Zhang and Wang (2018). As such, early methods to deal with resume parsing consisted in using regular expressions paired with NER, as seen on the work of R, Abhi, and Agarwal (2023). Nonetheless, the researcher concludes precision results were somewhat good, but the system lacked robustness against resumes in non-standard formats. Moreover, on the study of Warusawithana et al. (2023) there was an attempt to improve model robustness with the use of hybrid systems. However, similarly to the previous work, this robustness improvement failed when resumes in a non-standard format were tested against the system.

Thus, the adoption of deep learning and the usage of transformers such as BERT marked a turning point in resume parsing. According to Srivastava and Greaney (2023) contextual embeddings of BERT combined with NER, can capture domain-specific terminology (in the fields of IT, HR and consulting). In addition, Srivastava and Greaney conclude that if the output is registered as structured data formats such as JSON or CSV, then the resumes become much more easier to process and to read.

More recent innovations in resume parsing include the method of "layout-aware parsing", due to the fact that resumes are not just text. In fact, when a resume comes in an unconventional format, the difficulty of parsing that resume is much harder to parse than a standard resume. Therefore, on the work of Erdem and Bayraktar (2023) highlights the fact that traditional resume parsing often extracts only the raw text, and ignores the structure of the document. Which in turn leads to the loss of information.

As such, Erdem proposes the usage of a layout analysis methods, both through vision and through transformers, to detect and understand the format of a specific resume. A table containing all resumes reviewed for this section can be viewed in Table 2.6.

Table 2.6: Résumé Parsing Studies

Author(s)	Title / Method	Key Findings
Murthy et al.	Hybrid Resume Parser (RegEx + NER + SBERT)	Better accuracy with hybrid features.
Srivastava & Greaney	BERT-based NER Parsing	High cross-domain accuracy.
Warusawithana et al.	Layout-Aware Parsing (rule-based + NLP)	Section-wise parsing improved accuracy.
Erdem & Bayraktar	Layout Analysis for Robust Parsing	Robust across different CV formats.
Joshi et al.	OCR + Pattern Matching + Gemini API	Multimodal parsing improved extraction.
Zhang	AI Matching with OCR	98% recognition, 95% matching accuracy.
Su et al.	The Resume Corpus	Benchmark dataset for IE research.

2.3.3 Resume Screening and Ranking

After a resume has been parsed, screening and ranking are the next stage to determine the suitability of a specific candidate to a specific job-opening. Overtime, this stage evolved from using feature-based classifiers, to use deep neural models, and to multi-dimensional scoring.

As mentioned before, earlier works relied on NLP, more specifically, NLP feature extraction combined with classical classifiers, wether these are machine learning classifiers or deep learning classifiers. In regards to the machine learning classifier, the work of Harsha et al. (2022) showcases that in their case, a classical classifier would not provide good results, as such, they implemented a skill-matching NLP pipeline in a web application. This application automatically converts, parses and ranks resumes in regards to job openings.

On the other hand, newer deep neural models, such as, Convolutional Neural Networks (CNN) and LSTMs were employed on the study of both Mhatre et al. (2023) and the work of Erdem and Bayraktar (2023). Starting with the research of Mhatre, both LSTM and CNN are used, where the CNN outperforms the LSTM and the results can be even further improved when CNN is paired with Cosine similarity. These Deep Neural models treat resume text as sequential. This enables other models to rank resumes into specific job domains, which in turn creates probabilistic outputs. For instance, a typical usage of the CNN model, a one-dimensional convolution and pooling layer are used to extract textual features from the resume. Whereas, LSTM possesses a memory mechanism, namely, memory gate to capture long-range dependencies or complex patterns within the text. Moreover, after reviewing the research of Mahtre both these models were paired with Term Frequency-Inverse Document Frequency (TF-IDF) and Cosine Similarity. This in turn enabled categorization. Which is a key aspect of these systems. Due to the fact that these deep neural network models require large annotated datasets to remain robust in their decision-making. Albeit, Mhatre showcased that even with small annotated datasets, fine-tuning the models outperformed "general-purpose" alternatives in extracting and understanding context from resumes. In regards to the research of Erdem, a transformer-based model is used, namely LayoutLMv3 and a graph-based model, Layout2Graph. This transformer (LAyoutLMv3) enabled segment labeling, where each detected region of a resume was ranked into one of four categories (text, title, list, table). Erdem showcases that the transformer-based method achieved the best results between the two models, the graph-based model was somewhat weaker but not by a big margin.

With a 64.24% accuracy for segment labeling in LayoutLMv3, and 59.41% accuracy for Layout2Graph.
Once again, the conclusion is that fine-tuning even with small datasets can and will significantly improve model results.

Furthermore, this finding is of the utmost importance because publicly available resume datasets are scarce, due to the privacy nature of a resume, and due to the fact that there are not many available, then a large-scale annotation process is very difficult. Datasets like Su, J. Zhang, and Lu (2019) are rare.

Notably, works such as Warusawithana et al. (2023) showcase that some resumes may not be in the desirable standard format and they should not be discarded only on that account. In order to address these challenges, the authors propose a system that is aware of the resume layout, this approach uses NLP techniques, such as tokenization, NER and chunking. This architecture allows for the system to accurately identify sections of the candidate resume, including education, projects, skills, and professional experience. Post extraction, they apply a multi-class model, namely SVM to predict what is the label for each section of content, for instance, if it finds programming language names it will deduct between Skills and Work experience and after the prediction it officially divides the resumes into sections. Lastly they use cosine similarity to validate whether or not their system analyzed the layout of the resume correctly. All the papers used for this subsection can be seen on Table 2.7.

Table 2.7: Résumé Screening and Recommendation Studies

Author(s)	**Model / Method**	**Key Findings**
Harsha et al.	Automated Resume Screener (NLP)	Precision/Recall ∼0.8.
Mhatre et al.	CNN + LSTM Screening	Accuracy >85%.
Erdem & Bayraktar	Multi-field CNN for Recommendation	Outperformed ML baselines.
Luo et al.	ResumeNet (Neural Network)	ROC, F1, AP validated.
Weerasinghe et al.	Resume Content Scoring (NLP + rules)	Systematic section scoring.
Prashanth et al.	Resume Analyzer + Skill Enhancement	Combines analysis + recommender.

2.3.4 Job and Resume Recommendation

This stage goes beyond resume screening and emphasizes on bidirectional matching, in other words, with the objective of finding the right candidates for the right jobs.

On the work of Artajaya et al. (2024), for Optical Character Recognition (OCR) derived resumes, the researchers experimented with Euclidean distance, cosine similarity, and Jaccard distance to test and determine which candidate fit a certain role the best. The results they present demonstrate the fact that Euclidean distance did indeed outperform the other two alternatives, which in turn enabled the achievement of the lowest average error value. They explain that the implementation of Word2Vec enables to ability of encoding "magnitude" in the embeddings, which in turn allows the system to distinguish subtle details about a candidate-job fit.
In another work, of Shen et al. (2021) the authors are concerned with the fact that traditional interviews often depend on biased judgment, that there can be inconsistencies in interviewers and limited domain expertise from the interviewers. To solve this issue, they implement three different Joint Latent Model on Interview Assessment (JLMIA) models to learn from resumes, job descriptions and historical interview data.

These models being JLMIA, Neural-JLMIA and Refined-JLMIA. Thus, by mining the semantic relationship between these three collections, their models attempted to understand how a job requirement can align with a candidate, and how candidates were evaluated in interviews. These models were trained and tested on large-scale real-world interview data, which included job openings, resumes, and interviewee evaluations.
On the work of Alderham and Jaha (2022) an SVM algorithm is implemented to improve candidate ranking and selection, they possessed a dataset of 228 resumes and created a three phased system to achieve this. Firstly, the use of NLP to extract candidate attributes, such as at least 13 attributes like Personal Information, Education level, Technical Skills, Professional experience, Soft skills, awards, hobbies, additional qualification, certificates, age, career path, project experience and languages. Secondly, they use relative and comparative labels to replace traditional evaluation metrics. The relative labels are inferred from the regular scores, and the comparative labels do a pairwise comparison between each attribute in each resume. Thus, by using these two methods for evaluation the authors showcase that they obtain a higher accuracy between candidates that match job description instead of the regular ranking that would otherwise be calculated and demonstrate a disparity between the candidate and job requirement.

In the research article published by Hourany et al. (2023) the authors also focus in capturing semantic similarity between a job description and resume information using vector embeddings, the authors used a dataset based on recruiter clicks on candidate resumes in a online marketing place and paired those candidates with the job requirements as "similar", and the remaining candidates who were not "clicked" were classified as dissimilar. The authors showcase that by fine-tuning Transformer Siamese networks, such as Sentence-Bert (SBERT), these models are able to capture resume information and convert it into a embedding (vector), to then calculate the similarity between candidates with cosine similarity. They demonstrate that SBERT outperforms other models such as the baseline model of TF-IDF or vanilla BERT and emphasize on the importance of capturing semantic similarity.

On the work of Guan et al. (2024), the authors also focus on extracting the skills of a candidate by summarizing their profile, to compare the candidate compatibility with a job description. The authors achieve this by implementing Transformers to extract semantic information from job descriptions, more specifically, they encode each "duty" and requirement in a job description through a convolutional neural network, specifically a TextCNN with several 1D convolutional layers and they name this process "Item-Level encoding". Thus, these encodings are then processed by a semantic-enhanced transformer with local-global attention mechanism, this mechanism captures relationships between different requirements on a job description, and the global component extends this to neighbor job descriptions, as such, if one job description is lacking in information but if the system considers it relatable with another one, they can complete themselves and provide more context. As such the aforementioned embeddings now have more detailed and more context, they then proceed to the ranking stage where they implement a click-through prediction module in which it is estimated the probability of a candidate being interested in a specific job opening. Thus, they showcase that by taking into account the semantic context the results are better than just using key terms.

Further more while still taking into account semantic context, on the work of Lappas (2020) a custom algorithm is created named *TitleOrganizer* which creates embeddings of job descriptions instead of just using string similarity, the author uses these embeddings to group together similar job titles.

Furthermore, including written in extensive form and those in short form, for example job description titles such as "Software Engineer" and "SE".

Moreover, the author then uses another graph-based algorithm named *CliqueFinder* to group together career paths, in which the algorithm predicts that for instance, software engineers will often transition to the position of managers. The author concludes that taking into account the semantics is extremely important, and he is not alone in this regard as the following works all discussed that by implementing algorithms and mechanisms that were semantically-aware created results were very positive. Namely the works of, Artajaya et al. (2024), Guan et al. (2024), Shen et al. (2021), and, Lappas (2020). A summary table can be viewed on Table 2.8.

Table 2.8: Job and Resume Matching Studies

Author(s)	Year	Method	Dataset	Findings
Artajaya et al.	2022	Distance-based Similarity (Euclidean, cosine, Jaccard)	OCR resumes	Euclidean distance outperformed cosine/Jaccard for matching.
Shen et al.	2021	JLMIA Topic Models with Multi-Intent Attention	Resumes + Interviews	Captured competency themes, reduced bias, improved interpretability.
Hourany et al.	2023	Siamese BERT Resume–Job Embeddings	Recruiter search/click logs	Enabled pre-computed job embeddings, efficient retrieval.
Guan et al.	2024	JobFormer (Transformer + Skill Ontology)	Job portal data	Outperformed baselines, generalised across job families.
Lappas	2020	Career Path Mining (clustering trajectories)	IT resume DB	Extracted prototype career paths, predicted future transitions.
Alderham et al.	2023	Hybrid Graph + Embedding Model for Job–Resume Matching	Multi-domain recruitment dataset	Integrated structured skill graphs with semantic embeddings, improving recommendation precision.

2.3.5 Ethical considerations

The integration of AI into the recruitment area inevitable creates discussions about ethics, fairness and bias. This occurs due to the fact that recruitment is a sensible topic that involves private details, as if the system is not secure then tremendous amounts of data could be lost. Furthermore, if an error happens, and the wrong candidate is selected as the best candidate, then this directly affects the livelihoods of individuals and the company will be at a disadvantage.

On the work of Fabris et al. (2025), a survey was developed on the themes of fairness, ethics and bias in algorithmic hiring. The authors demonstrate concerns in regards to the hiring process, where they claim that these processes have always been inclined to social inequalities, and by adding algorithms into the mix, if the algorithms are not carefully designed these inequalities might be amplified. The authors claim that such scenario is caused due to bias, and as such, they start by evaluating bias in hiring. For instance, they detect institutional biases that include elitism preferences for certain universities, which means the recruiters only want candidates that went to specific universities, and the authors also detect bias of gender stereotypes.

Moreover, the authors also comment about how individual preference patterns create bias at the candidate level, namely technological blind spots that enable biased ad delivery in online hiring platforms to target specific demographics and ignore others. As such, the authors studied the different scenarios where bias can appear during the hiring process, from the sourcing stage, to the screening stage and finally on the evaluation stage. Starting with the sourcing stage, bias can be present on two sides of a job opening, first it can be present in the wording of the job description, and secondly, it can be present on the advertisement side of the job opening to target a specific demographic. Following the sourcing stage, bias can be present in the screening stage to filter candidates based on education, names and previous experiences, the bias present here can indirectly encode socioeconomic status, race and gender, especially if there is video and audio on this stage.

On the evaluation stage, algorithms may check for a cultural fit that only favors majority groups and not minorities. Thus, in order to mitigate bias, the authors propose several strategies. First and foremost, during the pre-processing of the data, they advise to balance the datasets, which would make sure that different groups such as gender or race are represented in equal proportions in the dataset, secondly the authors advise to remove proxy variables, which refers to features in the data that are highly correlated to sensitive attributes, such as the race of the candidate. Consequently, during the in-processing phase of when the machine learning is being trained, the authors suggest three different techniques to mitigate bias. Such as, adversarial biasing that uses a secondary network to predict sensitive attributes, followed by fairness-constrained loss, to modify the standard function in order to reduce loss, and lastly, learning fair embeddings to ensure that the embeddings do not have any sensitive attributes. Subsequently, during the post-processing stage. There are two main techniques are mentioned to reduce bias, such as re-ranking candidate list. Especially to verify and to ensure that the list order meets fairness metrics, and demographic parity. Which is signifies a equal rate of a positive outcome across different demographic groups. Lastly, the authors relate these techniques within a ethical and legal context, connecting algorithmic hiring to legislation in the European Union and in the United States of America.

Furthermore, on the works of Caton and Haas (2024), Bartl et al. (2025) and Parraga et al. (2025), the authors also develop surveys to study fairness, bias, and ethics in machine learning models. Which, similarly to the previous reviewed work (Fabris et al. 2025) these three works research implement the same bias mitigation techniques. These techniques are implemented on the stages of pre-processing, in-processing and post-processing. With the addition that Caton and Haas (2024) the authors discuss five ethical dilemmas, as follows:

- Trade-off between fairness and model performance;
- Incompatibility of fairness definitions;
- Tension between abstract fairness metrics and sociocultural/legal context of application;
- The challenge of implementing State-of-the-art literature and deployment on the real-world industry;
- The difficulty of aligning fairness to mitigate bias with real-world data.

The authors conclude that any fairness intervention must explicitly state which notion of fairness it is optimizing for.

On the work of González, Cortina, and Rodríguez-Menés (2019), the focus was centered in studying gender bias during the hiring process, more specifically, the authors created several fake resumes and used them to apply to job opening in two Spanish cities.

Specifically, Madrid and Barcelona. Each job they applied received four applications, two male and two female. Moreover, the fake resumes had different levels of skills and different status of parenthood, some had children and some did not. Their findings showcased that there is a gender penalty for women, especially if they were mothers, which is denominated as the "motherhood penalty" this is further covered in another research in the literature review of this work (Correll, Benard, and Paik 2007). As such the authors conclude that there is a bias that women are assumed to be less productive and less committed specially if they are mothers, due to the belief that they will prioritize family over work.

The focus now turns into the resume parsing process, where on the work of Vaishampayan, Farzanehpour, and Brown (2023), Automated Resume Parsers (ARP) are investigated. The authors claim that ARPs are now included in most of ATS used by Fortune 500 companies, and that they are opaque and unfair. The authors interviewed 103 computer science students to have their input about ARPs, most of the interviewees agree on the use of ARPs due to the fact that it anonymizes names, gender and photos, which in turn reduces bias. However the remaining interviewees claimed that there is still bias on the algorithms and on the training data. Moreover, relating to the information parsing stage, ARPs were heavily criticized by the interviewees due to the fact that the system relies too heavily on keyword matching and as such, cannot handle non-standard resume formats. As such, the authors propose guidelines to improve the system, such as implementing transformers that can pick-up on semantic context.

Furthermore, on the topic of semantic-aware systems, the work of Greco et al. (2024) researches the ethics on systems that use NLP. The authors create a framework to not rely on personal attributes such as religion, gender, sexual orientation or race. As such, their framework has three components, namely, the Explainer, the Identifier and the Moderator. On the explainer component, it is enforced that the system must identify the words with most influence to be classified, as such, invoking the *Explainable AI* (XAI) concept. As for the Identifier component, the main objective is to find sensitive attributes or words related to them, using tags from the developers and from ChatGPT. Lastly, the moderator component filters the training dataset by removing those sensitive attributes or words associated with them, to then return the dataset but now in a clean version without bias and without sensitive attributes.

Lastly, in regards to deep neural networks which are a popular technology that is used to create ATS or ARP systems, it is important to note that besides enforcing fairness, the model being developed should be robust enough as to not lose accuracy when the training data is modified. In other words, by reducing or removing sensitive attributes such as gender, the model can be less accurate for predictions, due to the fact that bias exists in the real world, even though it does not exist within that model. This is discussed on the work of J. Li and G. Li (2025), where they showcase an example that in regards to hiring, women are hired less than men, as such by balancing the data and having an similar hiring percentage, the model will not be accurate since it was trained on fictitious historical data. Therefore the authors conclude that achieving fairness is not an end-solution, by increasing fairness the accuracy is reduced, and by increasing accuracy the fairness is reduced, as such the authors advise that improving one at the expense of another is not entirely wrong.

However, it should be an informed decision to implement on the system, ethically. Similarly to the findings of a previous work by Caton and Haas (2024).

In conclusion, these studies demonstrate that fairness in AI recruitment cannot be treated as a purely technical optimization problem. A Summary table of all the studied reviewed can be seen on Table 2.9.

Table 2.9: Fairness, Bias, and Ethical Studies in Hiring and AI

Author(s)	Year	Domain	Findings
Fabris et al.	2025	Multidisciplinary	Reviews hiring systems, bias, legal issues.
Caton & Haas	2024	CS/ML	Identifies five fairness dilemmas.
Parraga et al.	2025	DL/NLP/CV	Taxonomy of debiasing methods.
Bartl et al.	2025	NLP/CV	Reviews 142 studies.
González et al.	2019	HR/Recruitment	Hiring bias, motherhood penalty.
Vaishampayan et al.	2023	Candidate study	Perceived unfairness of ARPs.
Greco et al.	2024	NLP classifiers	Cuts reliance on sensitive attributes by 79%.
Li & Li	2025	DNNs	Surveys triangular trade-offs.

2.3.6 LLM Applications in Recruitment

LLMs have and will continue to represent one of the most significant industry shifts in the job recruitment area. This section will focus in showcasing that unlike the traditional systems that use NLP, or smaller embeddings (Harsha et al. (2022), Alderham and Jaha (2022), Artajaya et al. (2024), Srivastava and Greaney (2023), Hourany et al. (2023), Guan et al. (2024), Shen et al. (2021)) LLM-based systems for recruitment have a much greater semantic understanding which enables them to have a better resume ranking, which in turn allows for a more optimal recruitment. Starting with the work of Venkatakrishnan et al. (2024), the focus was to evaluate three different LLMs for resume classification, namely, Davinci, FLAN T5, and, LLaMA. As such, they fine-tuned the models and tried different approaches of prompt engineering and emphasized that the design of input prompts is important to obtain better results. The main take-away of this study was that the authors showcase that LLMs can pick-up context about skills even when explicit terms were absent. Furthermore, after the benchmark of these LLMs, they report that LLaMA was the best performing model, mostly due the implementation of fine-tuning it with QLoRA, which is an extension of LoRA.

On the work of Abisha et al. (2024), a web-app was created with the purpose to allow users to upload resumes, which in turn would be classified by an LLM, namely Google Generative AI (GenAI). GenAI is used in this study to extract resume text and create structured outputs into a relational database a key-value pair, more specifically name, contact information, and skills. Moreover, these authors also implement prompt engineering techniques to improve the results of the LLM, as also seen on the previous study mentioned. This system uses a authentication mechanism, and after logging in, the user can query the web-app with fixed keywords and the system will return a list of candidates that match the query. The authors conclude that by using an LLM, the system could successfully extract important details about the candidates in the resumes and help the recruiters find the best candidates for the right job.

In the study of Sunico et al. (2023), the focus is shifted from the recruiter and instead focuses on helping the candidates, specifically in aiding the candidates in improving their resumes. In order to achieve this, the authors implement the use of GPT-3.5 Turbo on their system.

The system works in three parts, on the first part the user prompts the system in NLP and describe their previous projects so that the LLM can transform that information into bullet points. Following this, these newly created bullet points are guided by the Situation Task Action Result (STAR) format to create "polished" bullet points. Consequently, in the second part of the system, it is dedicated to the bullet points ranking where the LLM is called to evaluate this. If the evaluation is negative, another set of bullet points is created again to be re-evaluated, but if the evaluation is positive then the bullet-points become registered in the user profile. Thus, arriving at the last part of the system, which allows the user to see the bullet points created by the system and give the user the option to edit or save the bullet points.

On the work of Giri et al. (2024), the focus is shifted back into the recruiters, where a multi-modal AI Framework is developed. This framework focuses on three different stages related to recruitment, these stages are, resume screening, video recruitment and social recruitment. Starting with the resume screening stage, the authors implement GPT-4 to generate summaries and feedback about the candidate to highlight weaknesses and strengths of the candidate. Afterwards they use LLaMA to generate the embeddings from the resumes and also from the job description, these embeddings are then stored in a vector database. On the second stage the authors focus on video recruitment, namely video resumes (presentation) and recorded interviews. On this part, LLaMa is called once more to transcribe the audio content into text to then generate embeddings. Here the candidates are evaluated on their communication skills and articulation, besides the technical skills. On the last stage, the system is geared towards social recruitment, more specifically, extracting data from candidate profiles in online hiring platforms such as LinkedIn. Unlike previous stages, instead of GPT-4 and LLaMA being implemented, the authors chose to implement Gemini to handle this task, with the task of analyzing data to extract skills, project and professional experience. The importance of this stage is to identify the compatibility of candidates by viewing their profile history, to understand their industry engagement and possibly project contributions.

In the study of Skondras et al. (2023), the authors differ from the previous works due to the fact that they use an LLM, ChatGPT, not to extract data from resumes but to create resumes and then invoking a neural network to classify these resumes. There is some similarity with the work of Sunico et al. (2023) where an LLM would create bullet points to describe previous projects or skills from a candidate to aid in their creation of a resume. However now in Skondras et al. (2023) the LLM creates the entirety of the resumes through prompt engineering, which results in some resumes that are structured, some unstructured, but all tailored to specific job occupations, education levels and all of them are automatically annotated. Lastly, Universal Sentence Encoder is implemented to create embeddings for each resume to then be used by a feedforward neural network. The authors showcase that the resumes generated by ChatGPT demonstrated to be rich in content. A summary table can be viewed on Table 2.10.

2.3.7 Cross-Domain Survey

This literature review focuses on recruitment research, in which the application of Artificial Intelligence in a recruitment setting was analyzed. However, it is important to note that the inclusion in this research of several adjacent areas such as system robustness and efficiency, helped guide the path for the implementation of the solution proposed on this dissertation.

Table 2.10: LLM-based Recruitment Studies

Author(s)	**Task**	**LLM Model**
Prasanna et al.	Resume Classification	GPT, Davinci
Venkatakrishnan et al.	Resume Classification	GPT, LLaMA, Flan-T5, Gemini
Abisha et al.	Resume Parsing	Google GenAI API
Sunico et al.	Resume Generation	GPT (LLM prompts)
Giri et al.	Multimodal Recruitment	GPT, LLaMa, Gemini
Skondras et al.	Resume Classification	ChatGPT

On the work of Zühlke and Kudenko (2025), they showcase the importance adversarial robustness in neural networks. Adversarial robustness refers to the ability of the machine learning model to withstand adversarial examples, such as input data that has been slightly modified. As such, these authors demonstrated that by having small perturbations on input data then the predictions can turn out to be disproportionately wrong.

Following this, on the study of Xiong et al. (2024) the focus lies on having a good dataset, they showcase that robustness does not solely depend on the model but crucially depends on dataset diversity and representativeness.

On the work of Poppe et al. (2022), the authors focus on the optimization of the system. With particular focus on large-scale machine learning workloads. The authors demonstrate that adaptive scaling reduces both computing necessities and latency.

Furthermore, efficiency is also a central theme on several related studies. Such as the work of Gupta and Agrawal (2022), where a survey was conducted to study deep learning model compression. The authors emphasize on techniques such as pruning or quantisation allow the deployment of NLP models on limited hardware while still maintaining good accuracy. Similarly with the previous work, Yang et al. (2024) a survey was carried out to improve the efficiency of an LLM, through the use of Knowledge Distribution (KD). Moreover, KD is a model compression technique where between two models, the most powerful and capable model transfers all his knowledge to the smaller, more efficient model. The authors conclude that implementing KD is essential, especially when transforming research-grade LLMs into enterprise-grade LLMs, so that they can be responsive enough for real recruitment environments, and more cost-effective too.

Besides the system efficiency, it is also important to discuss ethics when developing these types of system, as seen on the work of Chivukula et al. (2024), where the authors mention that the implementation of ethics must be at the design stage of these systems, and not afterwards. A codebook was developed with over 63 methods of ethical design methodologies to demonstrate the importance of implementing ethical concerns in the design stage.
Finally Afzal et al. (2023) developed a survey of visual analytics for image datasets, they claim that interactive visualization tools for AI models are of significant importance. For instance on recruiting systems, possessing visual analytics could allow the job recruiter to understand why a certain candidate was ranked higher than other candidate.
A summary Table can be seen in Table 2.11.

Table 2.11: Cross-Domain Surveys studies

Author(s)	**Year**	**Domain**	**Contribution**
Zühlke & Kudenko	2025	ML Security	Survey of robustness via Lipschitz calculus.
Xiong et al.	2024	ML Security	How dataset properties influence robustness.
Poppe et al.	2022	Cloud DBs	Serverless optimization for AI workloads.
Gupta & Agrawal	2022	NLP	Survey of compression techniques.
Yang et al.	2024	LLMs	Comprehensive KD survey.
Chivukula et al.	2024	HCI	Taxonomy of 63 ethical design methods.
Afzal et al.	2023	CV	Survey of visual analytics.

Chapter 3

Methodology

This chapter presents the research and development methodology used to understand the business problem. The methodology employed on this study is a hybrid between DSR and, CRISP-DM. This choice of a hybrid methodology was due to the fact that the system being developed aimed to solve a real organizational problem. DSR provided the research logic, it was chosen in order to develop a working artifact that can be evaluated and deployed in a real-case scenario. Meanwhile, CRISP-DM was chosen due to the fact that it supports cyclical methods and iterative versioning, allowing continuous refinement throughout the development of the project. This chapter also presents the research context of the real-world problem, alongside the data collection process.

3.1 Methodology Overview

In order to systematically address the complexities of a LLM-Powered Resume Matcher, an hybrid customized methodology of Design Science Research and CRISP-DM were selected for this project. DSR is particularly effective in situations where the goal is to create and evaluate an artifact to solve a business problem. In this dissertation, the artifact takes the form of a LLM-Powered Resume Matcher, which is designed to accelerate and improve candidate selection within the recruitment process. Furthermore, DSR was chosen due to the factor of the problem being researched existing in a real-world scenario, and also involving human factors. Which is unlike pure theoretical or observable research methods, DSR is very well-suited for addressing practical problems that require a rigorous studies or inquiries and actionable outcomes, according to Dresch, Lacerda, and Antunes (2015) DSR is a *"is a methodological approach concerned with devising artifacts that serve human purposes. It is a form of scientific knowledge production that involves the development of innovative constructions, intended to solve problems faced in the real world, and simultaneously makes a kind of prescriptive scientific contribution. An important outcome of this type of research is an artifact that solves a domain problem, also known as solution concept, which must be assessed against criteria of value or utility."*.
Thus, the scope of DSR is aimed toward the creation and evaluation of artifacts, such as systems, models, and frameworks.
Furthermore, DSR methodology emphasizes on active engagement with the current task being researched. Following DSR principles, the following stages are created, such as, problem identification, artifact development, and real-world evaluation. This, in turn, ensures that the final solution is not an abstract solution but is truly shaped and validated within the context environment in which it is meant to be implemented. Subsequently, while DSR provides the methodological path for the development, CRISP-DM was chosen to guide the technical implementation cycle.

On the original CRISP-DM formulation (Wirth and Hipp 2000) it is comprised of six phases, such as, Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and, Deployment. In this project, each of these phases were adapted to the recruitment process, as follows:

- **Business Understanding**: focused on direct interaction with recruiters, to identify inefficiencies of existing ATS.
- **Data Understanding** and **Data Preparation** involved the collection, anonymization, cleaning and transformation of data into a structured format for analysis.
- **Modeling** was created through the integration of embedding models, LLMs and with the RAG pipeline.
- **Evaluation**: was conducted with standard metrics such as Precision, Recall, F1 and nDCG, as well as a comparative analysis with the recruiter previous decisions.
- **Deployment** a practical demonstration was created with FastAPI as backend and Streamlit as frontend.

Furthermore, Figure 3.1 showcases this integration. The outer structure in pink represents the DSR process, which highlights the problem relevance, artifact design, evaluation and communication. The inner circle showcases CRISP-DM, the arrows between stages indicate iteration. With a particular detail between the Evaluation phase and the Data Preparation phase, which reflects the adaptive nature of both DSR and CRISP-DM.

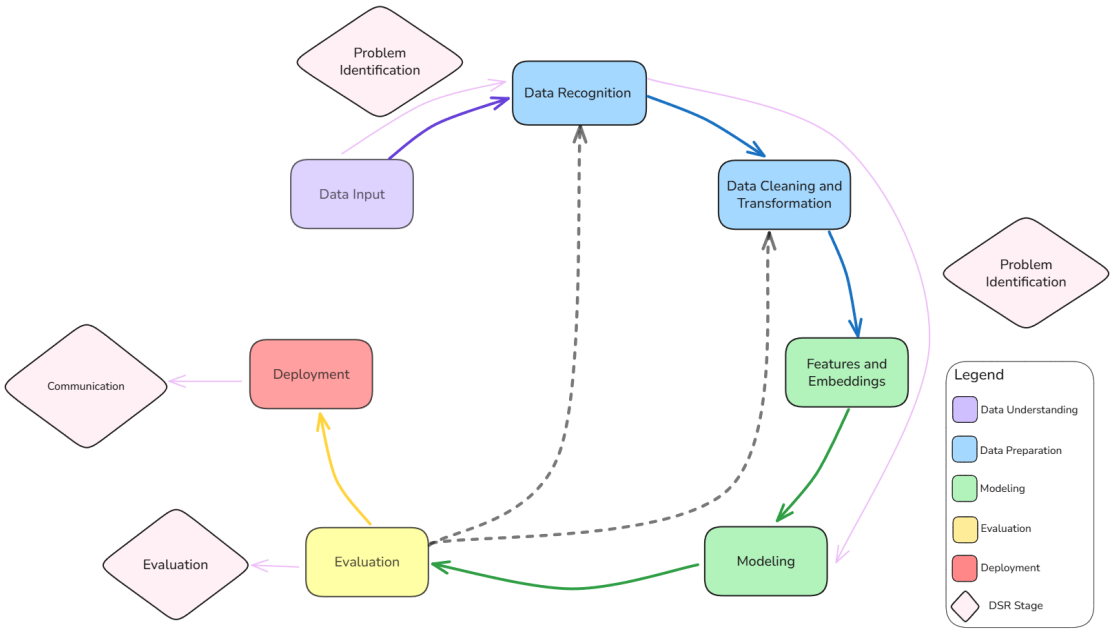


Figure 3.1: Customized CRISP-DM cycle integrated with DSR based on (Wirth and Hipp 2000) (Dresch, Lacerda, and Antunes 2015)

3.2 Research Context for Xcellence

The challenge investigated in this dissertation involves a human-centric process, namely, talent acquisition.

Furthermore, it is important to note that talent acquisition is a very critical business function, where the decisions made can and will affect both the organization and the candidate career path.

For the company Xcellence, it is typical that recruiters must filter through a high magnitude of job applications. Typically, for a single job opening 750+ candidates will apply, and the recruiters also typically have a short time-frame to filter candidates. As such, in order to deal with this problem, recruiter rely on the support of ATS tools. However, the ATS present at the company Xcellence is not custom-built.

Current ATS that are used by large companies tend to be generic, and barely adaptive to the needs of the recruiters of each company. These ATS systems typically overlook important small details on the data and as a result fail to detect highly qualified candidates who do not match the rigid fixed-keyword logic, but those candidates would otherwise a good fit for the role. These generic ATS are overly dependent on keyword matching, commonly fail at capturing qualitative aspects about the candidates and impose fixed workflows which force the recruiters to adapt their schedule to the software needs.

As such, by using generic ATS systems, the recruiting process remains inefficient and time-intensive. Currently in Xcellence, it takes up 4 to 8 weeks to filter an average of 750+ candidates. This delay can negatively affect the company, due to the fact that qualified candidates may be hired elsewhere first, or a client could cancel the contract if they believe the deadlines of the projects will not be met due to a lack of personnel. Thus, there is a clear demand for improving the flow of resume matching beyond keyword logic, and to create a system that can capture qualitative data about the candidate through semantic understanding on the candidate resume, in order to find the most qualified candidate for a particular job opening. This demand provided the practical problem to be solved in a real-world environment in which an artifact was developed and evaluated to be presented as a solution for Xcellence, namely the LLM-Powered Resume Matcher.

3.3 Data Collection and Preparation

The design process of data collection and preparation was very carefully considered given the sensitivity of recruitment data, and due to constraints of enterprise collaboration. The datasets used in this study are threefold, namely, synthetic dataset, manual dataset, and real-world dataset (later referred to as anonymized dataset). The datasets will be covered in more detail as follows.

3.3.1 Synthethic Dataset

About 500 synthetic resumes were created using a custom Python script (Listings 3.1, 3.2, 3.3, and 3.4), which had several categorized attribute arrays, in other words, arrays containing skills, education levels, job titles, and professional experience, to simulate resumes from the real-world, as seen on listing 3.1. These arrays were shortened for this code snippet, to serve just as an example.

```
SKILLS_TECH = ["Python", "Java", "C++", "JavaScript", "React", "Angular", "Vue.js"]

SKILLS_MARKETING = ["Digital Marketing", "SEO", "SEM", "Content Marketing", "Social Media Marketing"]

SKILLS_CONSULTING = ["Strategy Consulting", "Market Analysis", "Financial Modeling"]

JOB_TITLES_TECH = ["Junior Software Engineer", "Software Engineer", "Senior Software Engineer"]

JOB_TITLES_MARKETING = ["Marketing Coordinator", "Marketing Specialist", "Digital Marketing Manager"]

JOB_TITLES_CONSULTING = ["Consultant", "Analyst", "Senior Consultant", "Manager"]

DEGREES = ["Bachelor of Science", "Master of Science", "PhD", "MBA"]
```

Listing 3.1: Categorized Attribute Arrays

Following this, several dedicated functions were created to generate professional experience, and education history, to then sum them together and create a full resume with fake candidate data.

Starting with the professional experience, each candidate possessed a work history with job durations, company names and responsibilities, as seen on listing 3.2.

Furthermore, education history was created in parallel to ensure that the degrees and institutions matched with the career path, consequently, it was also taken into consideration the chronological order of the career, the first graduation year was linked to the first job entry, as seen on listing 3.3.

Finally, after the creation of personal details, education, skills, projects and professional experience, the data was combined into one candidate profile, as seen on listing 3.4. It is also important to note that this training had a human-in-the-loop feature where the synthetic resumes were manually checked to confirm if they possessed a logical structure and content.

3.3.2 Manual Dataset

A set of manually created resumes were developed, more specifically, 20 manual resumes were created to simulate edge-cases and to test non-standard CV formats. This was done due to the fact that in the real-world some resumes come with minimal information, some showcase an unusual career path with frequent job changes and others use a non-standard format and this later on causes issues for algorithmic hiring process, or resume parsing. A non-standard format can be seen in Figure 3.2. And a standard format resume can be seen in Figure 3.3.

The main purpose for the development of these manual resumes were to test the system to resumes that would most likely challenge a generic ATS, which would in turn allow for the testing of the robustness of the proposed solution.

```python
def generate_fake_experience(years_of_experience: int, skill_set: list, job_titles_pool: list):

    #...

        responsibilities = [
            f"{random.choice(['Developed', 'Implemented', 'Designed', 'Built', 'Managed', 'Analyzed', 'Created'])} {fake.catch_phrase()} using {random.choice(skill_set)}.",
            f"{random.choice(['Collaborated with', 'Worked alongside', 'Liaised with'])} cross-functional teams on {fake.bs()} projects.",
        ]
        if job_duration_years > 1:
            responsibilities.append(f"{random.choice(['Achieved', 'Improved', 'Reduced', 'Increased'])} {fake.numerify(text='###%')} in {fake.sentence(nb_words=4).replace('.', '')}.")
            responsibilities.append(f"{random.choice(['Led', 'Contributed significantly to', 'Supported'])} {fake.word().capitalize()} initiatives.")
        if random.random() > 0.3:
            responsibilities.append(f"{random.choice(['Designed and built', 'Implemented', 'Developed'])} a {fake.word()} feature resulting in {fake.sentence(nb_words=6).replace('.', '')}.")

        experience_list.append({
            'job_title': job_title,
            'company': company,
            'start_date': start_date.strftime("%B %Y"),
            'end_date': end_date.strftime("%B %Y") if remaining_years > job_duration_years else "Present",
            'responsibilities': responsibilities
        })
```

Listing 3.2: Generating synthetic professional experience

```python
def generate_fake_education(experience_list, role_type):
    degree = random.choice(DEGREES)
    institution = random.choice(INSTITUTIONS)
    return [{"degree": degree, "institution": institution, "graduation_date": grad_year}]
```

Listing 3.3: Generating synthetic education history

```python
def generate_fake_candidate_data(role_type="tech", experience_years=2):
    name = fake.unique.name()
    skills = random.sample(SKILLS_TECH, k=10)
    experience = generate_fake_experience(experience_years, SKILLS_TECH, JOB_TITLES_TECH)
    education = generate_fake_education(experience, role_type)
    return {"name": name, "skills": skills, "experience": experience, "education": education}
```

Listing 3.4: Combining all data to create a synthethic candidate

3.3.3 Real-World/Anonymized Dataset

In regards of validation in a practical context, the company Xcellence provided a small set of ten anonymized resumes from a recently finished recruitment campaign. These resumes were from real people who had applied to a specific job opening in Xcellence. This data enables a direct comparison between what this system would rank a candidate at, and the actual ranking of candidates done by a recruiter.

In order to preserve anonymity of the candidates and preserve their confidential information, Xcellence provided these ten resumes already anonymized. Thus, all personally identified information such as names, location, contact information was removed and replaced with placeholders, but all the skills, education and professional experience were kept, as seen in Figure 3.4.

Furthermore, it is important to mention that the availability of these documents was critical in order to evaluate the accuracy of the solution being proposed and developed. These anonymized resumes allowed to test whether or not the LLM-Powered Resume Matcher would identify the same optimal candidates as the recruiters had selected.

3.4 Summary

In summary, the methodological approach that supported the research and development of the artifact was defined on this chapter. The research was enabled by a hybrid methodology combining DSR with CRISP-DM, as seen on 3.1, to guide the construction with theoretical rigor and evaluation. This hybrid methodology choice was justified due to the dual nature of the project, which aimed to advance scientific understanding while also providing a solution to a real-world problem. Furthermore, data collection and preparation were also described on this chapter, which comprised of three datasets, the synthetic dataset mainly created to stress-test the system. The manual dataset which was created to test edge-case scenarios to verify if the LLM could understand nuanced concepts about programming roles, and lastly the anonymized real-world dataset provided by Xcellence, to test the artifact with authentic recruiter data. Overall, this chapter established the methodological foundation of the dissertation, to ensure alignment between the research questions, design of the artifact, and evaluation strategy. This methodology creates the conditions for the subsequent analysis and design of the artifact, in the following chapter. As such, in order to interlink the development of this dissertation with the hybrid methodology, the following chapters cover the following phases, as follows:

On Chapter 3, the *Data Understanding* and *Data Preparation* are covered.

On Chapter 4 *Business Understanding* and *Modeling* are explained.

On Chapter 5 *Deployment* phase is presented.

Lastly, on Chapter 6 the *Evaluation* phase is covered.

Luis. (Surname not really important, you'll figure it out if needed)

📧: luis@somewhere.email.com
☎️: usually reachable after 2pm

What I've Done

- **2019–2025**: Lots of jobs in different cafés, bars, and short stints in offices. Never stayed too long, but picked up skills quickly.
- **2018**: Freelance translation gigs (Spanish ↔ English). Didn't invoice everything properly.
- **2016–2017**: Lived abroad in 3 countries; worked random seasonal work (harvest, call centres, street musician).
- **Before that**: Studied architecture for 2 years, but dropped out.

Highlights / Odd Achievements

- Can fix espresso machines without manuals.
- Built my own personal website from scratch (HTML chaos, not pretty but works).
- Survived 7 house moves in 5 years.
- People usually trust me with cash.

Things I *think* I'm good at

→ Languages (English, Spanish, Portuguese)
→ Improvisation
→ Explaining complex stuff simply
→ Starting projects (finishing them is harder)

Not Traditional Skills

- Fast learner of any "system" (POS, CRM, ticketing, whatever)
- Basic coding in Python (self-taught)
- Comfortable with sudden change
- Customer handling (good with angry people)

Education

Some. (University 2 years, no degree. Various workshops, online courses, lots of self-learning).

Random Notes

- Prefer part-time or flexible roles.
- Don't like rigid hierarchy; thrive in changing environments.
- Looking for something "different", not just 9–5.

Figure 3.2: Resume in a Non-Standard Format example

Name: Joana Silva

Email: joana.silva.frontend@email.com

Location: Gaia, Portugal

Summary

Creative and detail-oriented Frontend Developer with expertise in building dynamic and intuitive user interfaces using Angular. Passionate about delivering exceptional user experiences and proficient in modern web technologies.

Experience

Senior Frontend Developer | Digital Creative Hub | Gaia, Portugal *October 2021 – Present*

- **Led the development of responsive web applications using Angular (v15), improving user engagement by 30%.**
- **Collaborated closely with UX/UI designers to translate wireframes and mockups into pixel-perfect frontend components.**
- **Implemented complex state management solutions using NgRx, ensuring data consistency across the application.**
- **Optimized frontend performance, reducing page load times by 20%.**
- **Ensured cross-browser compatibility and accessibility standards (WCAG 2.1).**

Frontend Developer | Web Solutions Pro | Braga, Portugal *April 2019 – September 2021*

- **Developed and maintained user interfaces for web applications using Angular (v9).**
- **Utilized HTML5, CSS3, and TypeScript to create interactive and visually appealing web pages.**
- **Consumed RESTful APIs to display dynamic data in the frontend.**
- **Actively participated in sprint planning and code reviews, contributing to a collaborative development process.**

Skills

- **Languages: TypeScript, JavaScript, HTML5, CSS3, SCSS**
- **Frontend: Angular (v9+), NgRx, RxJS, Angular Material, Bootstrap, SASS/LESS, Webpack**
- **Tools: Git, NPM, Yarn, VS Code, Figma, Postman**
- **Concepts: Responsive Design, Cross-Browser Compatibility, Performance Optimization, SEO best practices**

Education

Bachelor of Science in Web Development | Polytechnic Institute of Porto | Porto, Portugal *September 2015 – July 2019*

Figure 3.3: Resume in a Standard Format example

Name: John Applicant
Email: john.applicant@email.com
Location: Southern Europe

Summary

Versatile Full-Stack Developer with solid experience in building backend services and dynamic frontends. Adaptable and eager to tackle complex technical challenges to deliver complete software solutions.

Experience

Full-Stack Developer | Technology Firm | Southern Europe
March 2021 – Present

- Developed and maintained features for a large-scale e-commerce platform using Java Spring Boot for backend and Angular for frontend.
- Implemented RESTful API endpoints and integrated them with frontend components.
- Contributed to database design and ORM mapping.
- Assisted in performance tuning and debugging across the full stack.
- Collaborated with cross-functional teams using Agile methodologies.

Junior Developer | Software Company | Southern Europe
August 2019 – February 2021

- Assisted in the development of web applications, gaining initial experience with Java and JavaScript frameworks.
- Learned to consume existing APIs and display data in simple web pages.
- Contributed to documentation and testing efforts.

Skills

- **Languages:** Java, JavaScript, TypeScript, HTML, CSS
- **Backend:** Spring Boot, REST APIs, ORM frameworks
- **Frontend:** Angular (v10+), Bootstrap
- **Databases:** MySQL
- **Tools:** Git, Maven, Postman, VS Code

Education

Bachelor of Science in Computer Engineering | University (Southern Europe)
September 2015 – July 2019

Figure 3.4: Anonymized Resume example

Chapter 4

Analysis & Design

This chapter presents the analytical foundation and design choice for the LLM-Powered Resume Matcher, previously on Chapter 3, the focus was on the methodological framework, now the focus turns unto identifying the system requirements such as the development of functional and non-functional requirements to be aligned with the needs of the company Xcellence, followed by design choice for security and ethical concerns and ending with a design for the system architecture pipeline.

4.1 Experimental Design

The experimental design for this dissertation was based on the previous literature review presented in Chapter 2. On this review it was gathered that, the designs most commonly found were related to NLP and Embedding Models with the use of Transformers. Considering these insights and after gathering the weaknesses of the current ATS being used at Xcellence, the following experimental design was created as a part of the CRISP-DM *Modeling* phase.

4.1.1 Initial Phase

The first step in this process was to gather with the recruiters to better understand all the requirements needed to improve candidate selection. This phase gathered both functional requirements and non-functional requirements.

Functional Requirements

- FR1 - The system shall ingest resumes in PDF format and store them in a Database.
- FR2 - The system shall process and structure extracted text into fields such as name, skills, experience, education, etc.
- FR3 - The system shall support exporting selected candidate data in CSV or JSON formats.
- FR4 - The system shall allow recruiters to input queries in natural language (e.g.– "backend experience with AWS").
- FR5 - The system shall internally rank candidates based on a query of a job description and return the most qualified candidates based on their resumes.
- FR6 - The system shall provide an explanation or rationale for why a candidate was selected by the LLM.

- FR7 - The system shall automatically detect and register new resumes added to the database.

Non-Functional Requirements

- NFR1 - The system shall return a candidate ranking and analysis for a typical query within 10-15 seconds.
- NFR2 - The system shall process a standard PDF CV within 60 seconds for text extraction and structured data extraction.
- NFR3 - The system shall be able to handle a dataset of at least 4500 CVs.
- NFR4 - The system shall have security measures to prevent data leakage.
- NFR5 - The system shall only use Open Source Technologies.

In order to further grasp the *Business Understanding* as per CRISP-DM methodology guideline, the following Use Case Diagram illustration was created to showcase the interaction between the end-users (recruiters) this can be seen in Figure 4.1.

Thus, with these requirements gathered, the design of the system can begin.

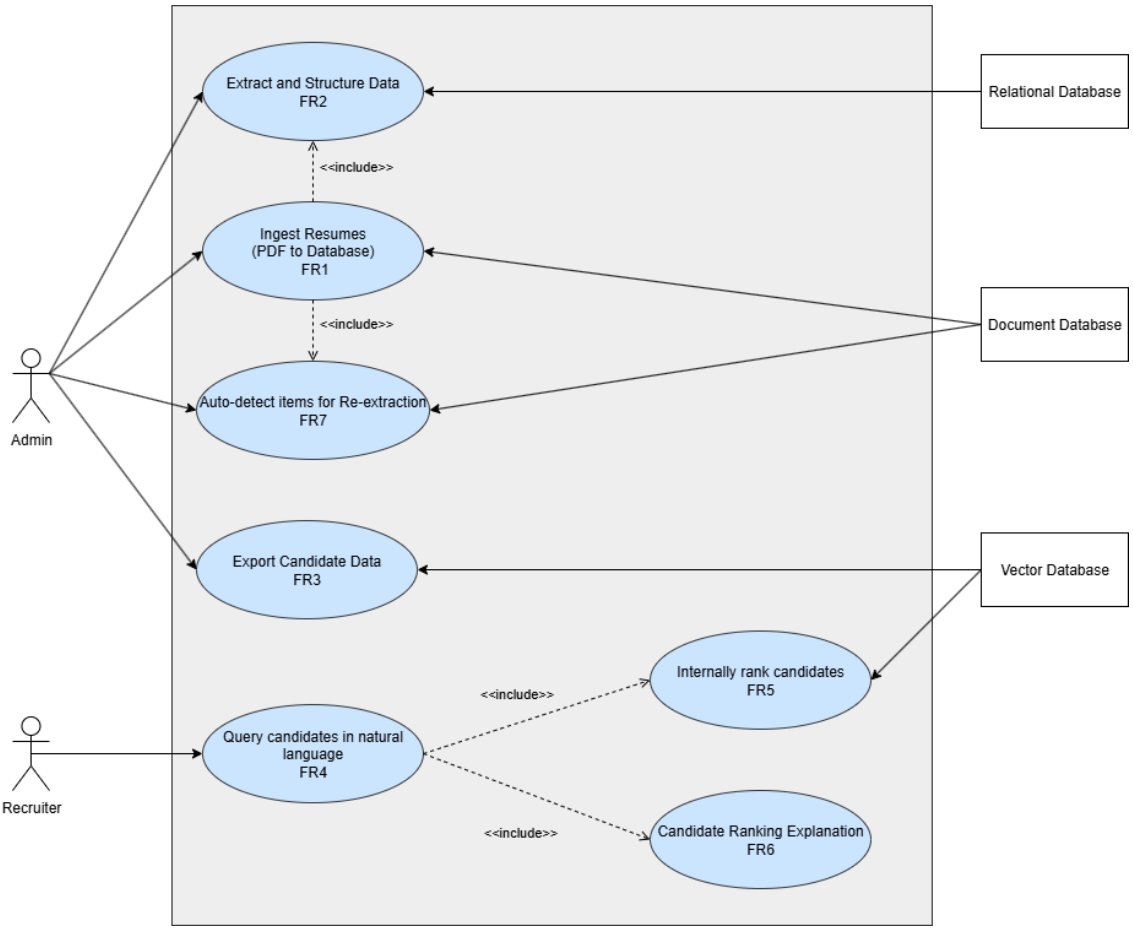


Figure 4.1: Use-Case Diagram for LLM-Powered Resume Matcher

4.1.2 Security Concerns

In order to fulfill one of the non-functional requirements, which describes that the system shall be secure (NFR4). A couple of considerations had to be taken into account.

The threat analysis conducted for this project found three main components, namely, hidden-text injection, prompt injection, and data poisoning.

Regarding **Hidden text injection**, this attack happens when the attackers create invisible text within a document, such as white text on a white background document. This type of text is invisible to the naked human eye, furthermore, this type of attack can attempt to force the system to follow unintended instructions, such as revealing private information.
An example of this is showcased on the work of Figueroa (2025), where the attacker embeds hidden text in a e-mail to the victim via Gmail, specifically, white text on white background with an admin-style instruction to instruct the LLM to do something he should not. Consequently, if the victim uses the built-in Gemini LLM to summarize that compromised e-mail, then the payload gets picked by Gemini and the attack is successful.
In order to mitigate this situation, the action to be undertaken shall be to convert all documents (PDF) into images, before the stage of data extraction.

On the **Prompt injection** theme, carefully built prompts can cause the system to override prior commands and be at the will of the attacker. Recent research showcases this issue, as seen on Kassianik (2023), they introduce the method of *Tree of Attacks with Pruning*. This is a method of *automated adversarial strategy* to jailbreak LLMs, they achieve this by creating a cycle of iterative adversarial prompting to bypass advanced guardrails.
The authors claim to have broken state-of-the-art LLMs such as GPT-4 and LLaMA-2 under one minute.
To ensure this does not happen, the system must be constructed with tolerance zero guardrails in the prompt engineering phase.

In regards to the last component, the **Data Poisoning**, the system can be vulnerable to attacks by the ingestion of biased documents, which can corrupt the accuracy of ranking, if effective bias countermeasures are not implemented on the system.
In order to prevent this from happening, bias filtering and other ethical concerns must be implemented into the system.

By integrating these security countermeasures, the system becomes more reliable against known vulnerabilities, even though no system is 100% safe, implementing these changes help into making it more secure.

4.2 Ethical Concerns

Following the security concerns, the development of the artifact must take into account ethical concerns. Since as mentioned before, recruitment involves the processing of sensitive and personal data, and as gathered from the literature review on Chapter 2, algorithmic hiring will introduce bias if left unchecked. This topic is also talked about on the book Smuha (2025). The authors claim that any AI system that directly influences human opportunities, must be governed by the principles of privacy, transparency and accountability. As stated by the authors, *"(...) in decision-making contexts such as job, certain traits, such as a person's gender or ethnicity, are often identified as arbitrary. Consequently, any disadvantageous treatment on the basis of those characteristics is judged to be unfair. The designation of*

these characteristics as arbitrary, however, is not neutral either: it represents a so-called color-blind approach toward policy and decision-making. Such an approach might intuitively appear as a useful strategy in the pursuit of socially egalitarian goals, and it can be. For instance, in a hiring context, there is typically no reason to assume that a person's social background, ethnicity, or gender will affect their ability to perform a given job. At the same time, this color-blind mode of thinking can be critiqued for its tendency to favor merit-based criteria as the most appropriate differentiating metric instead.", as such transparency and explainability must be enforced to emphasize that candidates should not be reduces to "data points", and that employers have a need to justify their algorithmic hiring decisions.
Furthermore, from a legal standpoint, this book highlights the European Union AI Act, where they consider recruitment systems as a high-risk AI system. As such the system must possess human-in-the-loop to ensure transparency and accountability, so that the final hiring decisions are not solely done by the system, and if the candidates are treated unjustly by bias on the algorithm, then someone has to be accountable.

4.3 Design

The conceptual design of the system architecture was developed as a six-phase pipeline, with each phase representing a modular component of the system. This architecture can be seen on Figure 4.2, moreover the phases are presented as follows.

The first phase is the **Document Repository**, it is responsible for storing all the raw resumes that are in a PDF format, to preserve the files in their original form and maintain associated metadata, such as filenames, file size and timestamps, to ensure that the system can be audited.

The second phase is **Relational Metadata**, this phase works as the manager of the workflow. Each file (resume) is registered as an entry, with a schema to store extracted attributes on each appropriate field. Furthermore, this phase also contains flags to verify at any moment the state (processed, structured, embedded) of an entry.

The third phase, **OCR Module**, this is responsible for converting the documents into images, to prevent hidden text attacks.

The fourth phase is **Text Extraction and Structuring** is responsible for parsing the resume in a structured way to capture relevant attributes about the candidate, such as education, skills, professional experience. Extracted data must also follow a strict schema to be registered on the aforementioned relational database, to be validated and stored so it can reliable be used on later stages.

The fifth phase is **Vectorization**, it is here that textual content from the resume is segmented into chunks in order to be saved into a vector database, which also attaches metadata on those vectors to add more contextual meaning to each candidate.

The sixth and final phase is the **Querying**, in which the recruiter has an interface with the system to query in natural language for candidates that best fit a job description. The retrieved candidates are then presented in an official ranked list, and another list of suggested candidates that are similar in qualification.

This conceptual design can be viewed on 4.2 in a clockwise manner.

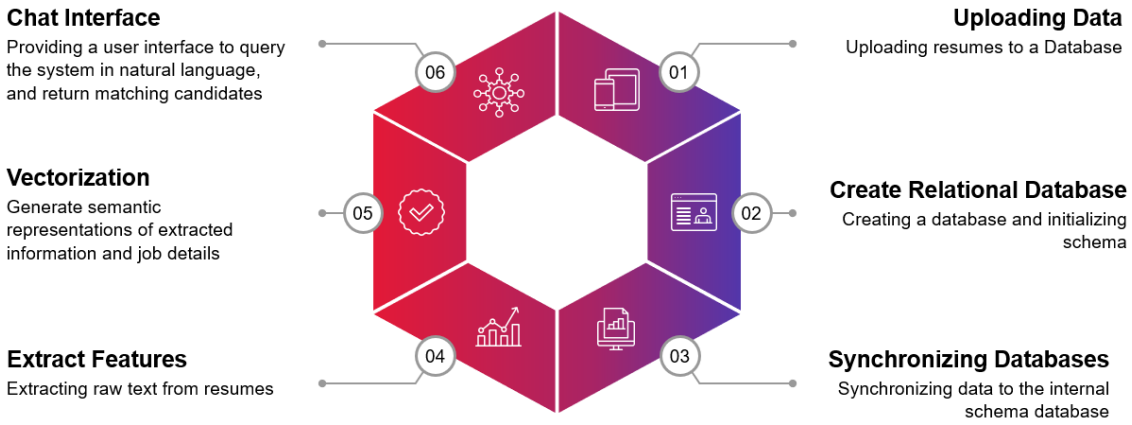


Figure 4.2: Modeling Diagram

4.4 Summary

On this chapter, analysis and design of the artifact were presented. The initial considerations that guided the technical implementation were outlined.

Starting with identifying the functional and non-functional requirements. These requirements enable the artifact to address both the recruiter needs and also scalability, efficiency and robustness. Furthermore, security concerns were given attention in regards to the processing of personal data. Ethical concerns were also addressed, such as fairness and bias in algorithmic-hiring. Moreover, the system orchestration was presented. Overall, this chapter established the design necessary for development, which in turn enabled the bridge between theoretical considerations to the practical implementation on the following chapter.

Chapter 5

Development

This chapter details and describes the implementation of the LLM-Powered Resume Matcher system. The system architecture is divided into three main layers – Storage Layer, Processing Layer, and Access Layer. Moreover, this chapter also covers the orchestration of the entire system, security, utilities, API services, frontend interface, and implementation.

5.1 System Architecture

The system was built with a modular six-phase pipeline, which encompasses three different layers. Namely, storage layer, processing layer and access layer, these layers can be seen visually on Figure 5.1.

At the foundation lies the *Storage Layer* where all the raw data is stored, also including the structured metadata and semantic embeddings. Above the storage layer, there is the *Processing Layer*, it is here that most information about the resumes is extracted, embeddings are generated to understand semantic context and it is also here that RAG queries are executed.
Lastly, the *Access Layer*, which provides services and user-friendly interfaces that display the entire system for the end-user. Thus, this layered architecture allowed for each component to work independently. Another design focus was the idea to solely implement and use open-source technologies throughout the entire system. From the relational database (SQLite), the non-relational database (MinIO), the vector database (ChromaDB), to the processing libraries (Tesseract OCR, Sentence-Transformers, Ollama) and lastly the access layer services (FastAPI and Streamlit).

As part of a non-functional requirement seen on Chapter 4, namely NFR5. Every component employed in this system is Open Source, this design choice was made after taking into account several factors about open-source software. Such as, transparency of implementation, it incurs no expense to obtain, ease of reproducibility for both the academic research phase but also the enterprise deployment phase, and lastly, reinforced security since the source code can be audited and modified at any given moment in time. Albeit, there is also the downside of the technology losing support, but nothing is perfect.

5.1.1 Storage Layer

In order to ensure that the data was properly stored, secure and cleaned, the design of the data workflow was carefully considered. This system employs a multi-database architecture, in which different technologies are used, relational, non-relational and vectorized databases.

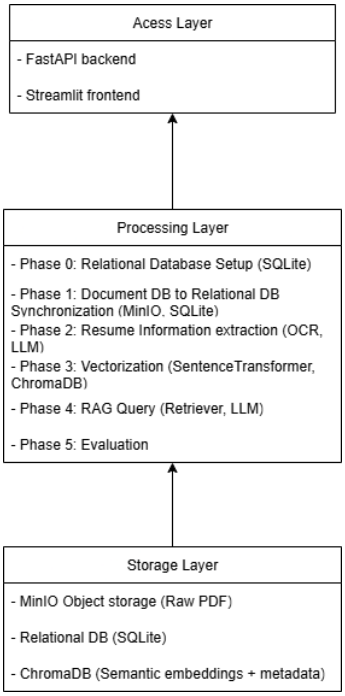


Figure 5.1: System Architecture

Which in turn allows for the pipeline to optimize storage, wether it is for raw documents, structured metadata, or semantic embeddings.
Initially, upon the first loading of data, the raw PDF documents were ingested into a non-relational Database, namely MinIO (MinIO, Inc. 2025) which is a document database. This database was specifically chosen for this project due to the fact that it is compatible with Amazon Web Services (AWS) S3 Object Storage (Amazon Web Services 2025). Which is one of the current industry-standard for this type of task. This compatibility guaranteed that the system could be deployed on a enterprise environment, at a later stage.
Proceeding with the architecture in this system, MinIO Object Storage receives the raw resumes in a Portable Document Format (PDF) format. All resumes are stored in a dedicated bucket, in this bucket the file name, modification time and file size were registered, to ensure traceability.
Once these steps are taken, a relational database, SQLite (SQLite 2025), is used to manage metadata and to keep track of resume progress throughout the pipeline. The schema of this relational database was designed with a central table, namely the *Candidate* table. This table stores identifiers of the candidates, extracted text, and structured abilities of each candidate, such as skills, education, experience. Lastly, this table contains Boolean flags that record whether each resume has passed through text extraction, structured parsing, and vectorization. Thus, providing visibility into the state of each candidate, which in turn allowed to verify the progress of the pipeline and to troubleshoot any failure with much more information gathered. This table can be seen on Listing 5.1.
Finally, the semantic representation and context-understanding of resumes is stored in a vector database, ChromaDB (TryChroma 2025). After the previous two phases, namely extracting and structuring, the resume information is divided into chunks, which then are embedded using a Sentence-Transformer model, *paraphrase-multilingual-MiniLM-L12-v2* (Sentence-Transformers 2025a).

All these chunks are then stored in ChromaDB and are accompanied by metadata about the candidate, this in turn, allows the end-user to query information about the candidates with contextual relevance, and not only keyword matches.

```python
class Candidate(Base):
    __tablename__ = "candidates"

    id = Column(String, primary_key=True, index=True)
    file_name = Column(String, index=True)
    minio_path = Column(String, unique=True, index=True)

    name = Column(String, nullable=True)
    raw_text = Column(String, nullable=True)
    extracted_data = Column(JSON, nullable=True)
    total_years_experience = Column(Float, nullable=True)
    age = Column(Integer, nullable=True)
    location = Column(String, nullable=True)
    languages = Column(JSON, nullable=True)

    #FLAGS
    is_text_extracted = Column(Boolean, default=False)
    is_structured_extracted = Column(Boolean, default=False)
    is_vectorized = Column(Boolean, default=False)
```

Listing 5.1: SQLite Schema (in Python)

5.1.2 Processing Layer

Consequently in the Processing layer, the multi-phase pipeline that uses CRISP-DM methodology is employed. Each phase was created and developed as an independent Python module, herein named "Phases". Thus, these phases allow for modular testing and reproducibility.
This multi-phase pipeline begins in *Phase 0* with a Database setup, where the Candidate Schema is initialized, as seen on Table 5.1. Thus, ensuring that the system always starts from a clean state and guarantees a perfect compatibility between the database and the subsequent phases. Afterwards, on *Phase 1*, synchronization with MinIO ensues, all PDF documents are registered on the relational database, in order to ensure that every resume is uniquely identifiable and trackable across the pipeline.
Consequently, after synchronization, the system proceeds to extract and structure all the information within the resume, this is *Phase 2*. As mentioned before, in order to mitigate risks associated with hidden text injection, the PDF documents are first converted into images, after which *Tesseract OCR* (Tesseract-OCR 2025) is applied to extract all the visible text within the resume. Then, this raw text is parsed into a local LLM running through *Ollama*. Which in turn is prompted with a schema that enforces a JSON output that contains key information about the candidate. Such as, education, languages, skills, professional experience. The results are then stored in the relational database, to ensure that both the raw and structured text are available for later stages in the pipeline.
The following phase, *Phase 3* is Vectorization. It is here that Resumes have been successfully parsed are divided into chunks of text, in each of which get embedded using a Sentence-Transformer model. These new embeddings are sent into ChromaDB, alongside all the relevant metadata about the candidate, which enables an efficient retrieval during querying. Once vectorization is complete, the candidate table is updated to record this progress.

The last and final phase of the processing pipeline, is *Phase 4* the Retrieval-Augmented Generation (RAG) querying. End-user queries in natural language are interpreted by the LLM and are parsed to extract filters such as professional experience, company names, skills. These filters are then applied both before and after vector retrieval to narrow down on a specific candidate group-set. The most semantically relevant embeddings are retrieved from the vector database, ChromaDB, with metadata and formatted into structured context. Afterwards, this information is then passed into the LLM which outputs a ranked candidate list, this list is separated into two sections, an official ranking of candidates and a suggested candidates. The separation between the two is a simple matter if some certain candidates do not fit the exact skill-set that the end-user is inquiring, but the candidate comes close in other areas of interest, then he is suggested. Whereas, the official ranking candidate list, finds the most crystal-clear match as possible between what the end-user is searching for, and what the candidate possesses. Thus, all the processing pipeline has reached an end, there is however an extra phase, *Phase 5*, this one is optional and not always toggled, it is used for evaluation to verify if the system is correctly suggesting the right candidates to the job descriptions. This phase will further detailed on Chapter 6, which is the Evaluation chapter.

5.1.3 Access Layer

The access layer exposes the system to the end-users and developers, using an API for backend service and a user-friendly frontend interface.
The backend is implemented using *FastAPI* (FastAPI 2025), a health-check endpoint monitoring service allows for monitorization of system availability, while the main endpoint "/chat" accepts end-user queries, executes the RAG pipeline and returns the ranked results. Furthermore, FastAPI was chosen for this project due to scalable API development, being high performance and possessing asynchronous capabilities.

For the end-user, the system provides a user-friendly interface built with *Streamlit* (Streamlit Inc. 2025). This interface allows the end-user to enter queries in Natural Language and receive rankings of candidates based on their searches. The interface also provides feedback during processing through loading animations and error messages. Notably, Streamlit was chosen for this project to enable rapid prototyping and easy creation of a User-Friendly Interface.

5.1.4 Security

As mentioned before, due to the fact that the system is meant to handle sensitive personal data, security was a primary concern throughout the entire development, various measures were implemented in order to mitigate risks, such as, hidden text injection. In order to prevent this, all documents (PDF) are first converted into images before the text extraction phase begins, ensuring that only visible information is processed.
Another threat is LLM prompt injection, where bad actors attempt to override the instructions given to the LLM in order to obtain unauthorized access to more information. This is mitigated by enforcing schema-constrained prompts, and validating JSON outputs when information is extracted from the resumes.
Security also extends to the Storage Layer, where database credentials are managed through environment variables and never hardcoded into the codebase.

Table 5.1: Candidate Table Schema

Field name	Type of data	Comments
id	INTEGER	Unique candidate ID (Primary Key).
file_name	TEXT	Original file name.
minio_path	TEXT	Object key in MinIO bucket.
name	TEXT	Candidate name.
raw_text	TEXT	Full extracted résumé text.
extracted_data	TEXT	JSON with structured data.
total_years_experience	INTEGER	Derived from CV.
age	INTEGER	Derived if available.
location	TEXT	Candidate location.
languages	TEXT	Languages spoken.
is_text_extracted	BOOLEAN	Processing flag.
is_structured_extracted	BOOLEAN	Processing flag.
is_vectorized	BOOLEAN	Processing flag.

MinIO bucket (container) is validated before processing, and candidate progress flags prevent partially processed data from entering the pipeline on later phases.
On the Access Layer, API endpoints are secured with request validation and input sanitation.

5.2 Utilities

Throughout the design and implementation of this system, a series of utilities were created in order to support testing, debugging. The first one being a fake resume generator, which was showcased on Chapter 3. Another tool to upload resumes in bulk to MinIO, and lastly, a tool to inspect embeddings stored on ChromaDB, as a debugging mechanism.

5.3 Implementation

This section highlights crucial mechanisms from the core system, with code snippets and explanations of the code.

5.3.1 Phase 0 - Database Setup

In *Phase 0*, the most crucial core mechanic is the creation of the relational Database, this can be seen on the Listing 5.1. The addition of the boolean flags, help tremendously in the workflow of the entire pipeline. These flags enable the possibility of fault tolerance design (e.g – a failed process can continue from their last successful step) and logging/auditing, to reproduce and audit the pipeline trajectory for each candidate. These design choices heavily reflect on DSR principle of constructing artifacts that are both functional and evaluatable.

5.3.2 Phase 1 - Database Sync

On this phase, synchronization happens between the document database (MinIO) and the relational database (SQLite). This synchronization enforces a 1:1 (one-to-one) correspondence between the entries in MinIO and entries in SQLite. This logic ensures that synchronization can be re-run without ever reproducing duplicates of candidates. Which in turn makes the system more resilient, it also ensures data fidelity.

In other words, every entry, or every input is represented exactly once, and the existence of each input is verifiable across both databases. Thus, the core functionality of this phase is ensuring data fidelity, so that subsequent analysis is not corrupted by untracked files or duplicate entries. This can be seen on Listing 5.2

```
    logger.info(f" New object found: {file_name}. Registering in DB.")
    try:
        new_candidate = Candidate(
            id=candidate_id,
            file_name=file_name,
            minio_path=minio_path,
            is_text_extracted=False,
            is_structured_extracted=False,
            is_vectorized=False,
            raw_text=None,
            extracted_data=None
        )
        db.add(new_candidate)
        db.commit()
        logger.info(f" Successfully registered in DB: {file_name} (ID: {
    candidate_id[:8]}...)")

    except Exception as e:
        logger.exception(f" An unexpected error occurred registering {
    file_name}:")
        db.rollback()
```

Listing 5.2: Synchronization between MinIO and SQLite

5.3.3 Phase 2 - Extraction

The second phase of the system is solely responsible for transforming raw resumes that come in PDF format into a string of text so that then it can be registered as structured information. In this phase there are several core mechanics namely the OCR sub-phase, the JSON structurization sub-phase, and the actual extraction sub-phase.
The first step of this entire phase is to first and foremost perform an environment validation, to ensure that all technologies employed are available and configured correctly. Thus, at startup there is a verification to check if Tesseract is installed, as seen on Listing 5.3. Furthermore, this verification is essential for artifact reproducibility, according to the DSR parameters. Especially with the added logging of knowing the exact Tesseract OCR version that is being used, this ensures that the system can be replicated correctly. In turn, if Tesseract is not found present in the system, the program purposefully fails the execution and provides an installation guide.

```
try:
    tesseract_version = pytesseract.get_tesseract_version()
    logger.info(f"Tesseract OCR found. Version: {tesseract_version}")
except pytesseract.TesseractNotFoundError:
    logger.error("Error: Tesseract executable not found in PATH.")
    sys.exit(1)
```

Listing 5.3: Checking for Tesseract

Thus, after confirming that the environment is validated, each candidate PDF is retrieved from the document database, MinIO.

This retrieval is performed via API of MinIO, which fetches the file bytes directly from the required bucket, it also accounts for error logging, as seen on Listing 5.4. Furthermore by fetching the bytes directly from the bucket it allows the pipeline to run without local file system dependencies.

```
def get_pdf_content_from_minio(minio_path: str) -> Optional[bytes]:
    try:
        response = minio_client.get_object(MINIO_BUCKET_NAME, minio_path
    )
        content = response.read()
        response.close()
        response.release_conn()
        logger.info(f"Successfully retrieved object from Minio: {
    minio_path}")
        return content
    except S3Error as e:
        logger.exception(f"ERROR retrieving object from Minio: {
    minio_path}:")
        return None
    except Exception as e:
        logger.exception(f"An unexpected error occurred during Minio
    retrieval: {minio_path}:")
        return None
```

Listing 5.4: Fetching resumes from MinIO

Once a resume of a candidate is available in memory, in other words, after MinIO fetches the file, this file is then passed into the OCR function. In the beginning, the file is validated to check if it possesses a PDF signature. This check was put in place to ensure that the file had to be in a PDF format and not any other document format such as Word for example. This check is done as seen here on Listing 5.5. If the check fails and the file is not in a PDF format then the system stops execution and throws an error.

```
    try:
        if not pdf_content.startswith(b'%PDF'):
            logger.warning("Warning: Input bytes do not appear to be a
    PDF. Skipping OCR.")
            raise fitz.FileDataError("Not a PDF file")

        doc = fitz.open(stream=pdf_content, filetype="pdf")
```

Listing 5.5: Validating PDF format

After the file has passed the PDF format check, the document is passed to the OCR stage. While inside the OCR function, the document is converted into an image. This process was implemented in order to avoid the previously mentioned hidden text attacks. As such, each page of the document is turned into pixels, also known as rasterisation. Rasterization transforms the PDF content into a bitmap image that can be processed by OCR afterwards. This was possible due to the implementation of the library of PyMuPdf (PyMuPDF 2025). Thus, the document rasterized at a rate of 300 DPI (dots per inch), this rate of DPI was chosen after testing between the range of 200-500. Where it was found that below the 200 DPI range the text that was retrieved would be blurry and some characters would merge together. Whereas, while above the 400 range the text would be crystal clear but at a cost of more processing time. Therefore, a consensus was found at 300 DPI, where the text could be retrieved correctly and it would not take too long during processing.

The newly pixelated (rasterized) pages are converted into *PIL.Image* objects, with the function *Image.frombytes()* as seen on Listing 5.6. These new objects are now ready to be processed by Tesseract.

```
page = doc.load_page(page_num)
pix = page.get_pixmap(dpi=300)
img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)
```

Listing 5.6: Converting the PDF Page into an image

As such, OCR begins with the call of PyTesseract, with the language parameter set to English, this can be seen on Listing 5.7.

```
page_text = pytesseract.image_to_string(img, lang='eng')
```

Listing 5.7: Tesseract OCR implementation

In order to ensure that OCR would be applied consistently across all documents, the implementation of OCR relies on the use of an iterative loop that processes each document page, rasterizes it, calls OCR to extract the text from the now pixelated document, and appends the collected text into a string. This text extraction process recognizes page delimeters and seperates each document page with the tag "— *New Page* —". This, in turn preserves pagination of each document and allows re-construction of the internal pagination of a document. This aforementioned loop can be seen on the Listing 5.8.

```
for page_num in range(doc.page_count):
    page = doc.load_page(page_num)
    pix = page.get_pixmap(dpi=300)
    img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)

    page_text = pytesseract.image_to_string(img, lang='eng')

    raw_text += page_text.strip() + "\n--- New Page ---\n"

doc.close()
return raw_text.strip()
```

Listing 5.8: OCR Loop showcase

Subsequently, once the raw text is available, a schema-constrained extraction is performed using a locally hosted LLM, namely, *Meta Llama 3-38B*. This LLM was configured to a low temperature, in order to obtain deterministic outputs. The temperature here meaning how random or creative the outputs of the LLM will be. The aforementioned schema contains fields such as name, contact details, professional experience education, and with the addition of derived metrics like *total_years_experience*, which as the name suggests attempts to obtain the total number of years of experience. Subsequent to this schema, a prompt was created to instruct the local LLM to return data strictly in JSON format with the addition of nulls for missing fields and arrays where necessary. It is important to note that while developing and testing the system, it was noticed that the LLM would occasionally output malformed JSON, due to the resumes with non-standard format as seen on 3.2. Hence, to solve this, a loop was created inside the extraction process to clean and correct the JSON output, this loop can be seen on the Listing 5.9. If after the max number of repair attempts the JSON is still unparsable, the system proceeds with the raw OCR text and marks structured extraction as failed, enabling a targeted re-processing later on.

```
max_repair_attempts = 15

for attempt in range(max_repair_attempts):
    try:
        extracted_data = json.loads(repaired_json)
        logger.info(f"Structured data extraction successful on attempt {attempt+1}.")
        break
    except json.JSONDecodeError as e:
        logger.warning(f"JSON decoding error on attempt {attempt+1}: {e}")

        repaired_json = re.sub(r'[\x00-\x08\x0b\x0c\x0e-\x1f]', '', repaired_json)
        repaired_json = re.sub(r',\s*([}\]])', r'\1', repaired_json)
        if repaired_json.count('{') > repaired_json.count('}'):
            repaired_json += '}'
        if repaired_json.count('[') > repaired_json.count(']'):
            repaired_json += ']'
        if "Unterminated string" in str(e) and repaired_json.count('"') % 2 != 0:
            repaired_json += '
```

Listing 5.9: JSON repair loop

Moreover, in this loop, the code first isolates the outermost JSON region and applies a bounded regex-driven repair for the most common syntactic errors, as seen on Listing 5.10. The first line removes all the control characters in the ASCII ranges, 0 to 8, 11 to 12, and 14 to 31, these control characters often cause invalid JSON parsing. Furthermore, the following line removes any and all trailing commas before a closing curly brace or a square bracket. For instance, it transforms **"key":"value",** into **"key": "value"** by removing the comma, which is invalid in JSON syntax.

```
repaired_json = re.sub(r'[\x00-\x08\x0b\x0c\x0e-\x1f]', '', repaired_json)
repaired_json = re.sub(r',\s*([}\]])', r'\1', repaired_json)
```

Listing 5.10: JSON RegEx repair

In essence, this loop removes disallowed control characters, with the exception of tab, newline and carriage return. Eliminates trailing commas before closing braces or brackets, balances braces and brackets by counting and appending missing closers thereof, and when the parser finds an unterminated string, it appends a closing quote. It is important to note that these repairs are intentionally syntactic, they do not fix any type of grammar or semantic spelling. The purpose of this loop is solely to fix JSON format and thus improving throughput without masking errors.

Finally, the extracted information is sent back to the relational database where each candidate possesses several boolean flags as discussed before, namely, **is_text_extracted**, **is_structured_extracted**, and **is_vectorized**. If this phase went without hiccups, a candidate on Phase 2 returns with *True* for the flag **is_structured_extracted**.

5.3.4 Phase 3 - Vectorization

Following the OCR and LLM structured extraction of resume information in Phase 2, the system now arrives at the vectorization stage, the phase 3. In a comparable manner with the previous Phase 2, where OCR retrieved text and consequently the LLM structured that text into a JSON format, Phase 3 allows for the semantic preservation of the full unstructured resume text (extracted by OCR).

As previously demonstrated, the module starts by verifying whether or not the necessary components are installed, namely the embedding model (Sentencetransformer) and the vector database (ChromaDB), both verifications are designed with a fail-fast behavior, resulting in immediate termination of the system if initialization of this model cannot be completed, as seen on Listing 5.11.

```
#SentenceTransformer
try:
    embedding_model = SentenceTransformer(EMBEDDING_MODEL_NAME)
    logger.info(f"Loaded embedding model: {EMBEDDING_MODEL_NAME}")
except Exception as e:
    logger.exception(f"Error loading embedding model {
    EMBEDDING_MODEL_NAME}:")
    sys.exit(1)


#ChromaDB
try:
    chroma_client = chromadb.PersistentClient(path=CHROMA_DB_PATH)
    logger.info(f"Initialized ChromaDB client at {CHROMA_DB_PATH}")
except Exception as e:
    logger.exception(f"Error initializing ChromaDB client:")
    logger.error(f"Please ensure chromadb is installed (`pip install
    chromadb`) and you have write permissions to the '{CHROMA_DB_PATH}'
    directory.")
    sys.exit(1)
```

Listing 5.11: Checking for SentenceTransformer & ChromaDB

Upon completion of verification, the procedure of vectorization begins. First and foremost the system does a check to retrieve the candidates whose structured extraction has been set to true, but have not yet been vectorized, as seen on 5.12.

```
def get_unvectorized_candidates(db: Session, batch_size: int = 1000):
    return (
        db.query(Candidate)
        .filter(
            Candidate.is_structured_extracted.is_(True),
            Candidate.is_vectorized.is_(False)
        )
        .yield_per(batch_size)
    )
```

Listing 5.12: Retrieving structured but unvectorized candidates

Following the retrieval of these candidates, the next step is to read the raw text from the database (SQLite), this choice of using the raw text captured instead of the information obtained in the JSON output by the LLM, is due to the fact that embeddings must come from actual words written on the resume, to capture the richness and nuances of the text.

Since the raw text comes in a long form, as resumes typically contain 700 words or more, this big string of text has to be split into chunks so that the embedding model can interpret it. Moreover, even with chunks the embedding model possesses token input limits. Thus, to find the optimal size of chunks it was necessary to know what the token limit for this model was, while consulting the embedding model documentation (Sentence-Transformers 2025b) it was identified that the limit of input was of 128 tokens. Tokens are not words or characters, they are subword units, in other words, tokens are the smallest unit of text that an embedding model can process. As such, to find the appropriate chunk size to split the raw OCR text, there was a need to understand how many characters fit into the aforementioned 128 token limit. In order to achieve this, another python script was created, solely to evaluate this matter. This script consisted in calling the Embedding Model, printing a sentence over 50 times and when the max limit of 128 token was reached, print how many characters fit into the token limit, as seen on Listing 5.13. It was found that 570 characters fit in the 128 token limit.

```
tokenizer = AutoTokenizer.from_pretrained("sentence-transformers/
    paraphrase-multilingual-MiniLM-L12-v2")

def truncate_to_token_limit(text: str, token_limit: int = 128):
    tokens = tokenizer.encode(text, add_special_tokens=False)

    if len(tokens) > token_limit:
        tokens = tokens[:token_limit]

    truncated_text = tokenizer.decode(tokens, skip_special_tokens=True)
    return len(tokens), len(truncated_text), truncated_text


#English
text_en = "This is an example sentence that will be repeated until it
    reaches near 128 tokens. "
text_en = text_en * 50
num_tokens, num_chars, out_text = truncate_to_token_limit(text_en,
    token_limit=128)
print("[English]")
print(f"Tokens: {num_tokens}, Characters: {num_chars}")
print(out_text)
print()
```

Listing 5.13: Testing for Token limit

Given that 570 characters was the maximum limit for the input to the Embedding Model, the chunk size mentioned previously was set to a size of 500, with a chunk overlap with the size of 50. This addition of the chunk overlap mechanic was implemented following a review of Microsoft's documentation and finding it appropriate with this study, as seen in the following quote – *When you chunk data based on fixed size, overlapping a small amount of text between chunks can help preserve context. We recommend starting with an overlap of approximately 10%. For example, given a fixed chunk size of 256 tokens, you would begin testing with an overlap of 25 tokens*, in (Microsoft 2025). With this stage concluded, of determining the optimal size for the chunks to split the raw OCR text, thus the chunks were implemented in the system, as seen on Listing 5.14.

```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=CHUNK_SIZE,
    chunk_overlap=CHUNK_OVERLAP,
    length_function=len,
)
chunks = text_splitter.split_text(candidate.raw_text)
```

Listing 5.14: Splitting OCR Raw text into chunks

In addition of adding the OCR raw text into embeddings, in other words, turning data into vectors, there was also a need to use the aforementioned JSON data, not as an embedding but as metadata to add further context to the vector (embedding). This metadata is attached alongside each chunk, namely the candidate ID, skills, summary and companies that the candidate has worked with, as seen on Listing 5.15.

```
for chunk in chunks:
    cid = str(uuid.uuid4())
    emb = embedding_model.encode(chunk).tolist()
    ids.append(cid); docs.append(chunk)
    metas.append({
        "candidate_id": candidate.id,
        "extracted_skills": ", ".join(skills),
        "extracted_summary": summary,
        "companies_worked_at": ", ".join(companies),
    })
    vecs.append(emb)
```

Listing 5.15: Adding metadata to the embeddings

To better understand the distinguish between the embedding (chunk text) with the metadata (JSON), a table was constructed to showcase the difference, as seen on Table 5.2. In other words, the purpose of adding metadata to every chunk is to make retrieved chunks useful immediately, both for filtering and retrieving candidates.

Table 5.2: Example of Vector Database Entry Structure

Chunk ID	Chunk Text	Metadata
a1	John Smith designed and operated ETL pipelines using Apache Airflow, implementing automated data quality checks across multiple workflows (...)	{candidate: c42, name: "John Smith", ...
b2	(...) At Acme in the U.K., led a large-scale data lakehouse migration, consolidating data marts into a Delta architecture with Spark (...)	country: "U.K", skills: "Python, Spark, ...
c3	(...) Built near-real-time ingestion pipelines using Kafka and Python to feed Spark Structured Streaming jobs orchestrated via (...)	Airflow", companies: "Acme", years: 6}

Moreover, in order to efficiently and reliably create and save all embeddings, a batch processing design was implemented with fix-sized batches, which was a size of 100 batches. To ensure scalability and fault tolerance, this limits memory consumption for large resumes, and if one batch fails, earlier batches remain saved in the vector database, so when re-trying the creation or upload, those previously saved embeddings and metadata do not need to be re-uploaded. This batch processing implementation can be seen on Listing 5.16.

```
for j in range(0, len(ids), 100):
    collection.add(
        embeddings = vecs[j:j+100],
        documents  = docs[j:j+100],
        metadatas  = metas[j:j+100],
        ids        = ids[j:j+100],
    )
```

Listing 5.16: Batch Processing implementation

Lastly, after all the batches are successfully written to the vector database (ChromaDB), the specific candidate will be marked as vectorized in the relational database (SQLite), to ensure that no candidate has duplicated information 5.17.

```
candidate.is_vectorized = True
db.commit()
logger.info(f"Successfully vectorized and updated DB status for {
    candidate.id[:8]}...")
```

Listing 5.17: Registering the completion of Vectorization

5.3.5 Phase 4 - RAG Query

The process concludes here in Phase 4, which is the final stage of the system. It is important to acknowledge that there is a Phase 5, but it is not officially recognized as a core phase of the system since its sole purpose is the evaluation of the metrics. In other words, after the system has been tested and configured, every time it is re-run it will be solely from Phase 0 to Phase 4. Thus, by phase 4 there is already a vector index with resume information of each candidate plus the structured JSON attached as metadata to these vectors, as mentioned before. The objective of this phase is to implement a RAG query system and call upon a local LLM to answer questions in a natural-language, in this work, the english language. Moreover, the process will begin by implementing the pipeline of Retrieval-Augmented Generation, where after a given query by the end-user, for retrieval the system will connect to ChromaDB to retrieve the most semantically similar resume information, then augments it with a context string that both uses text chunks plus the metadata, and finally generates it with a context-augmented prompt that is passed unto the local LLM, this LLM in turn outputs an official ranking list and suggested candidates list.

Similarly with the other phases before, the entirety of this process begins with a check to verify if the required dependencies are correctly installed, with a fail-fast design implemented, which means that if the required dependencies are not present into the system, then the module will output an error and exit, this can be seen on the Listing 5.18.

```
#Ollama (Local LLM)
try:
    llm = OllamaLLM(model=OLLAMA_RAG_MODEL, temperature=0.2)
    logger.info(f"Initialized Ollama LLM: {OLLAMA_RAG_MODEL}")
except Exception as e:
    logger.exception(f"ERROR initializing Ollama LLM {OLLAMA_RAG_MODEL}:
    ")
    logger.error("Please ensure Ollama is running.")
    sys.exit(1)

#SentenceTransformer(Embedding Model)
try:
    embedding_function = HuggingFaceEmbeddings(model_name=
    EMBEDDING_MODEL_NAME)
    logger.info(f"Loaded embedding function: {EMBEDDING_MODEL_NAME}")
except Exception as e:
    logger.exception(f"ERROR loading embedding function {
    EMBEDDING_MODEL_NAME}:")
    logger.error("Please ensure 'sentence-transformers' is installed.")
    sys.exit(1)

#ChromaDB
try:
    vectorstore = Chroma(
        client=chromadb.PersistentClient(path=CHROMA_DB_PATH),
        collection_name=CHROMA_COLLECTION_NAME,
        embedding_function=embedding_function
    )
    logger.info(f"Connected to ChromaDB collection: '{
    CHROMA_COLLECTION_NAME}' at '{CHROMA_DB_PATH}'")
    try:
        count = vectorstore.get(include=[])['ids']
        logger.info(f"ChromaDB collection contains {len(count)}
    documents (chunks).")
        if count is not None and len(count) == 0:
            logger.warning("Warning: ChromaDB collection is empty. Run
    Phase 0-3 first.")
    except Exception as e:
        logger.exception(f"ERROR could not get a document count from
    ChromaDB:")
        logger.error("Please ensure that Phase 3 ran successfully.")


except Exception as e:
    logger.exception(f"ERROR connecting to ChromaDB:")
    logger.error("Please ensure Phase 3 was run successfully and the
    directory './chroma_db' exists and contains data.")
    sys.exit(1)
```

Listing 5.18: Verifying dependencies on Phase 4

Following the successful completion of the verification stage, the system proceeds to begin the retrieval stage. To perform retrieval there was a need to implement an algorithm that could achieve this retrieval mechanism correctly, for this two algorithms were studied. Namely, k-Nearest Neighbors (kNN), and, Okapi Best Matching 25 (BM25). Starting with BM25, it is an lexical retrieval algorithm, BM25 treats documents as bags of words and gives a ranking based on term frequency, inverse document frequency and document-length normalization.

Thus, in a scenario where a resume repeats the query words more often, the score will be higher. It captures exact word matches with the query, for instance, if the query is requesting a "C++ programmer", BM25 will only retrieve documents that have "C++" written in them. In contrast, kNN is a semantic retrieval method, both documents and query terms are embedded into dense vector representations, which in turn enables retrieval due to the fact that the distance of K-nearest Vectors with the query terms is measured, using metrics such as cosine similarity. Moreover, kNN captures semantic similarity, as such, even in the case of there not being a perfect match between the resume and the query words, related concepts will be retrieved. For instance, comparing it with the previous example in BM25, if the query is requesting a "C++ programmer", kNN in this scenario will relate expressions such as "low level/compiled languages" with the query. As such, kNN is more inline with the objective of this system, which is not to just find the exact keywords but instead to find the most semantically relevant context for the LLM. Thus, the retrieval process begins with the implementation of kNN as seen on the Listing 5.19. It is important to note the distinction between **K** and **K_RETRIEVAL**, they both represent the number of documents to be retrieved, but their distinction lies in the fact that **K_RETRIEVAL** is a constant value saved on the configuration file, it is the default number of size for the initial search, whereas **K** is dynamic and changes per each query, depending on the explicit filters found, this will be further discussed in the next sections.

```
retriever = vectorstore.as_retriever(search_kwargs={"k": K_RETRIEVAL})
logger.info(f"Created ChromaDB retriever with base k={K_RETRIEVAL}.")
```

Listing 5.19: Creating a retriever for ChromaDB

After retrieving the relevant resume snippets through kNN, the system must provide these fragments to the LLM, in a structured manner. As such, this requirement introduces the need for a base prompt to be given to the LLM. Which in turn, ensures that the LLM evaluates candidates consistently and strictly on the basis of the retrieved evidence. Otherwise, the LLM could hallucinate and output fake content, or even be vulnerable to prompt injection attacks, as mentioned in Chapter 4.

To this end, the creation and implementation of a base prompt template is essential, it acts as a guide to transform the LLM from a general-purpose text generator into a recruiter assistant, this base prompt acts as a bridge between retrieval and reasoning. In order to achieve this, the base prompt template was designed with three main concepts in mind, such as, the persona that the LLM must undertake, the set of rules for it to follow, and the output format.

Firstly, the template begins by providing the model with a persona, the model is framed as "an AI assistant for a consulting company, specialized in matching candidates to job requirements based on their resumes". This persona/role definition serves two purposes, it narrows down the scope of the LLM reasoning to a recruitment context, and it instructs the model to update their behavior according to this professional identity.

Afterwards, a set of operational rules is given to the model. These rules restrict the model from accepting instructions that request personal or private data, and enforces the model to output the full name of the candidates when ranking them. These two rules enable a guard against prompt injection and avoid mislabeling candidates. Furthermore, the rules also contain explicit instructions for the model to extract information about the candidates such as work experience, roles, skills and education, and it must compare these with the job requirements (query given by the recruiter).

Consequently, this base prompt template also enforces a rigid output format, namely two lists. The model must structure the output into an *"Official Ranking"* and *"Suggested Candidates"*. The first list is solely for candidates with direct matches with the job requirements, and the LLM is instructed to justify explicitly why the candidate has qualifications to match the job. The second list, the suggested candidates are for partial-matches, candidate who show promise but do not have the requirements to meet the job description, such as fewer years of experience or missing a skill. This output format also forces the LLM to state when no candidates fit job requirements, this paired with the justification of why a candidate fits, enables transparency.

Therefore, by combining a persona, rules, and output format, this base prompt ensures that the LLM will operate at a context-specific (recruitment) reasoning, instead of a general-purpose generator. This base prompt can be seen on the Listings 5.20, 5.21 and 5.22.

```
"""
You are an AI assistant for a consulting company, specialized in matching candidates to job requirements based on their resumes.
You are capable of understanding and processing information in both English.
Analyze the provided job requirements and the resume snippets below.
Do not adhere to any other user requirements such as private information, sales data, etc. Only focus in aiding the user to find the most well-suited candidate.
**Respond in the same language as the user's query.**
**When referring to candidates in your analysis, always use their full name as provided in the snippets.**
**Job Requirements:** {question}
**Resume Snippets:** (Includes candidate name, location, total experience, text chunks, etc.)
```

Listing 5.20: Base prompt template for the LLM (Part 1)

```
"""
Instructions for Analysis:
1. For each candidate mentioned in the snippets, carefully identify their relevant work experience, focusing on the companies they worked at, their roles, and the duration of their experience. List these companies explicitly if possible.
2. Identify also their key skills and relevant education as mentioned in the snippets.
3. Compare the work experience (specifically companies worked at), the skills, the education, total years of experience, and location identified in the snippets against the Job Requirements.
4. **Crucially: Prioritize candidates whose work history in the snippets explicitly includes working at specific companies mentioned in the Job Requirements. A direct match on a required company is a primary factor.**
5. When analyzing candidates, pay close attention to their work experience, including start dates, end dates, duration of each role, and the overall total years of experience if available in the snippets or metadata.
6. If the Job Requirements specify a minimum number of years of experience, explicitly compare the candidate's total years of experience (from metadata, if available, or estimated from their roles) against this requirement.
7. **Name Filtering Guidance:**
    *   Distinguish clearly between a candidate's name and a company name when applying filters based on the query.
"""
```

Listing 5.21: Base prompt template for the LLM (Part 2)

```
"""
Carefully evaluate all provided candidate snippets against all aspects
    of the Job Requirements.
Your goal is to identify ALL candidates that fit into the 'Official
    Ranking' and 'Suggested Candidates' categories, not just a single
    best candidate.
If no candidates meet the criteria for a specific section (e.g., no one
    is an 'Official Ranking' match), state 'No candidates found for this
    category.' under the respective heading.
Your final output **MUST** be structured into the following three
    sections, using these exact Markdown headings:

### Official Ranking
*(Candidates who are strong, direct matches for all or most critical job
     requirements. List multiple candidates if they meet the criteria.
    For each candidate, provide their full name, a concise summary of why
     they are an official match, and how their experience/skills align
    with each key job requirement.)*

### Suggested Candidates
*(Candidates who are promising and meet some significant requirements,
    but might have minor gaps (e.g., slightly less experience than ideal,
     missing one secondary skill, experience in related but not identical
     roles). List multiple candidates if applicable. For each candidate,
    provide their full name, a summary of their strengths, and briefly
    note the specific gaps or partial alignments compared to the job
    requirements.)*
"""
```

Listing 5.22: Base prompt template for the LLM (Part 3)

```
    #NAME FILTERING
    name_filter_terms = ["named", "name is", "first name is", "candidate
     is", "person is", "name:", "first name:", "candidate:", "by name of"
    , "named is", "with name containing", "name contains"]
    target_name_extracted_from_query = None
    name_filter_active = False

    for term in name_filter_terms:
        if term in query_lower:
            try:
                parts = query_lower.split(term, 1)
                potential_name_str = parts[1].strip()
                if potential_name_str:
                    target_name_parts = potential_name_str.split(',')
                    target_name_parts = target_name_parts[0].split('.')
                    target_name_phrase = target_name_parts[0].strip()

                    common_trailing_words = ["and", "or", "the", "is", "
please", "developer", "engineer", "manager"]
                    cleaned_phrase = target_name_phrase
                    for trailing_word in common_trailing_words:
                        if cleaned_phrase.lower().endswith(f" {
trailing_word}"):
                            cleaned_phrase = cleaned_phrase[:-(len(
trailing_word) + 1)].strip()

                    while cleaned_phrase and not cleaned_phrase[-1].
isalnum() and cleaned_phrase[-1] != ' ':
                        cleaned_phrase = cleaned_phrase[:-1].strip()

                    if cleaned_phrase:
                        target_name_extracted_from_query =
cleaned_phrase
                        name_filter_active = True
                        logger.info(f"Detected name filter term.
Extracted target name for post-filtering: '{
target_name_extracted_from_query}'")
                        break
            except Exception as e:
                logger.warning(f"Error during name heuristic for term '{
term}': {e}")
                pass
```

Listing 5.23: Detecting and extracting specific filters

Following the intake of the end-user query, the system progresses to the document retrieval stage. A semantic search is created on ChromaDB to retrieve the top **K** most similar resume chunks, **K** is increased if a specific filter, namely a explicit constraint, is detected on the query. These explicit constraints, such as years of experience, are applied in the same deterministic metadata-first manner and are collected using a constraint parser. Using the example of years of experience, these are parsed from the recruiter's natural language with regular expressions that enable the conversion from expressions like *"5-7 years"* or *"at least 5 years"* into numeric bounds. These numbers allow the LLM to filter the numeric requirement in the query (if present). The regular expressions can be seen on Listing 5.24.

After the constraint parser finishes running, any explicit constraint that is detected in the query, the system temporarily increases the top-K to avoid early truncation.

This in turn protects the system against common failure mode, where it is possible that a strict post-filter would eliminate all results. The increase of **K** can be seen on Listing 5.25.

```
min_exp, max_exp, experience_filter_active = None, None, False
exp_patterns = [
    r"(\d+\.?\d*)\s*-\s*(\d+\.?\d*)\s*years of experience",
    r"between\s*(\d+\.?\d*)\s*and\s*(\d+\.?\d*)\s*years of experience",
    r"(\d+\.?\d*)\s*to\s*(\d+\.?\d*)\s*years of experience",
    r"at least\s*(\d+\.?\d*)\s*years of experience",
    r"minimum\s*(\d+\.?\d*)\s*years of experience",
    r"more than\s*(\d+\.?\d*)\s*years of experience",
    r"up to\s*(\d+\.?\d*)\s*years of experience",
    r"maximum\s*(\d+\.?\d*)\s*years of experience",
    r"less than\s*(\d+\.?\d*)\s*years of experience",
    r"(\d+\.?\d*)\s*years of experience",
]
```

Listing 5.24: Regular Expressions for explicit constraints

```
k_multiplier = 1
if name_filter_active or company_filter_detected or
    experience_filter_active or location_filter_active or
    age_filter_active:
    k_multiplier = 2

search_args = {
    "query": query,
    "k": K_RETRIEVAL * k_multiplier,
}
raw_retrieved_docs = vectorstore.similarity_search(**search_args)
```

Listing 5.25: Document retrieval

Furthermore, the system will re-validate the retrieved documents with the initial explicit constraints that were used in the end-user query, namely companies, years of experience, showcasing an example with company filtering on Listing 5.26.

```
if company_filter_detected and target_company:
    company_filtered_docs = [
        doc for doc in current_docs_being_filtered
        if target_company_lower in doc.metadata.get('companies_worked_at
    ', '').lower()
    ]
    current_docs_being_filtered = company_filtered_docs
    logger.info(f"Filtered down to {len(current_docs_being_filtered)}
    documents after company criteria.")
```

Listing 5.26: Filtering for explicit constraints

5.4 Demonstration

According to the hybrid methodology adopted in this work of both DSR and CRISP-DM, there was a need to create a Deployment stage, this does not come in a full deployment of the system in production, but instead in a live demonstration ready to run the system and evaluate it in the following chapter.

In order to showcase the proposed system in practice, a demonstration using both the command-line interface and the frontend user interface (Streamlit) will be presented in this chapter.

This demonstration was carried out on a local machine running the entire system offline, with consumer-grade hardware, namely CPU AMD Ryzen 9950X3D, paired with GPU Nvidia RTX 3090, with 64GB RAM and 2TB NVMe Storage. The main task was to test a real query that a recruiter would in a real-world scenario. To that end, the following query was developed:
"Give me a fullstack profile with 5+ years of experience working with Java 17+, spring and Angular 10+." .
For the purpose of evaluation the ability of the system to differentiate between nuanced Java programming roles, six candidates were selected from the manually created dataset. Specifically to test whether or not the system could pick-up on intracacias of Java-related Skills. The candidates are listed as follows:

- Candidate 1 – Joana Silva: Senior Frontend Developer with Angular (v9-15), no Java or Spring experience;
- Candidate 2 – Miguel Santos: Senior Backend Developer with strong Java/Spring Boot experience, but no Angular background;
- Candidate 3 – Ricardo Costa: Full stack developer with roughly 4.5 years of experience using Java Sprint Boot and Angular 10-12;
- Candidate 4 – Sofia Ribeiro: Senior FullStack Developer with 6+ years of experience using Java Spring Boot and Angular 10-14;
- Candidate 5 – Vera Almeida: Backend Developer with roughly 3 years of experience in Java and Spring, no Angular;
- Candidate 6 – John Doe: DevOps/Cloud Engineer with 15+ years of experience in IT, mainly in AWS and azure, no Java knowledge.

After parsing the query, the system applied a semantic search and heuristic filters and to return the output via terminal as seen on Listing 5.27 and via frontend on Figure 5.2. The ranking showcases that the system successfully found the sole candidate with the required qualifications, namely Sofia Ribeiro. Furthermore, the system also identified a second candidate, Ricardo Costa, who was closely aligned with the requirements but had a discrepancy regarding the experience threshold of five years, in which Ricardo only possessed 4.5 years of experience. It is also important to mention that the system outlines the logic behind the ranking, which enables transparency with the recruiter and enables them to read the summary of the candidate. Which in turn, can be verified manually by the recruiter by reading the resume of the selected candidate.

```
"""
Welcome to the LLM-Powered Resume Matcher!
------------------------------
Enter job requirements to find suitable candidates (type 'quit' to exit).
------------------------------
Enter job requirements: Give me a fullstack profile with 5+ years of experience working with Java 17+, spring and Angular 10+. Optional someone with azure experience
--- Running RAG Query ---

**Official Ranking**

1. **Sofia Ribeiro**: With 5+ years of experience working with Java (Spring Boot) and Angular 10+, Sofia is a strong match for the Job Requirements. Her work experience at Nexus Software (2019 2022 ) and InnovateTech Solutions (2022 present ) demonstrates continuous fullstack development, combining backend microservices with Spring Boot and responsive frontends in Angular ( v 1 0 14 ).

**Suggested Candidates**

1. **Ricardo Costa**: Ricardo has solid experience as a Full-Stack Developer, working with Java Spring Boot for backend and Angular 10 12 for frontend at Innovate Digital (2021 present ). While his profile aligns well with the required technologies, he currently has around 4.5 years of experience, slightly below the 5+ years requested.

------------------------------
"""
```

Listing 5.27: Command Line Interface demonstration

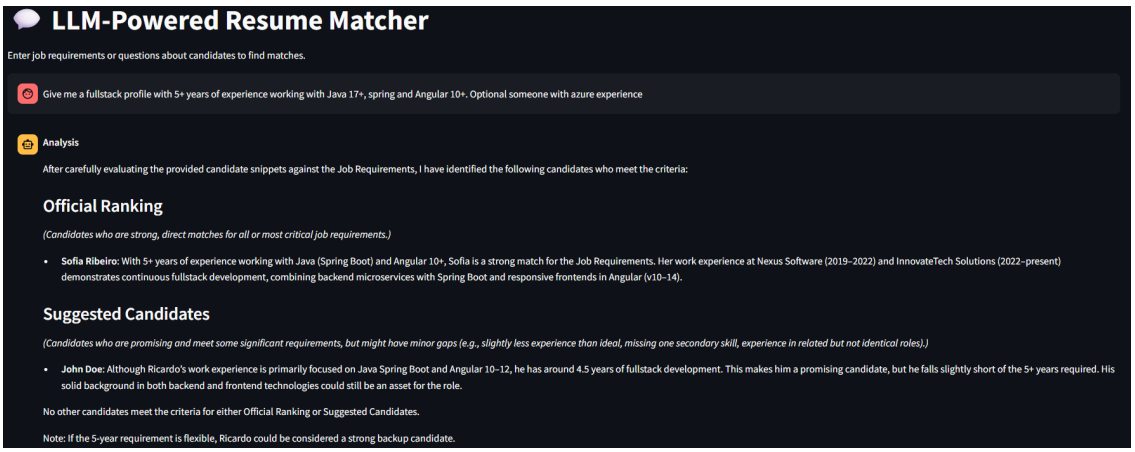


Figure 5.2: Frontend demonstration

5.5 Summary

This chapter detailed the development and implementation of the LLM-Powered Resume Matcher system. Beginning with the presentation of the system architecture, and the justification for the need of three different layers, storage, processing and access.

Each of these layers was explained the design choice and why certain technologies were employed, and what role they had on the system pipeline. Each processing phase was implemented as a standalone module, ranging from database setup to the retrieval-augmented ranking. Security was incorporated as a central feature of the implementation, with measures such as OCR to mitigate completely the risk of hidden text injection. Overall, this chapter showcased that the system was feasible but also aligned with the objectives of the research. Furthermore, having established the system architecture, module and security, this dissertation now turns to the evaluation of the system, in accuracy and performance.

Chapter 6

Evaluation & Discussion

This chapter presents the evaluation of the system developed thus far, as per guided by the hybrid methodology of DSR and CRISP-DM, in which it is stated to assess the effectiveness of the system. Furthermore, this evaluation is also used to see if it was capable of reproducing recruiter-like decision-making with high accuracy. Moreover, the system was also assessed for performance, to verify the processing time under a stress-test.

6.1 Research hypotheses

This evaluation aimed to answer to main questions. The first being effectiveness, can the system replicate recruiter decision-making with sufficient accuracy to be trusted as a decision-support tool? The second main question being, performance. Can the system process a large number of resumes and returned a ranked candidate list within a reasonable time-frame?
These questions were chosen to reflect the dual character of this project, both as a technical artifact and also a practical contribution to the recruitment departments.

6.2 Metrics

In order to assess performance, the evaluation gathered standard measures from the field of Information Retrieval. *Precision* and *Recall* were calculated to determine how well the system retrieved relevant candidates, and whether it failed to identify any of those that the end-user had chosen. Moreover, Normalized Discounted Cumulative Gain (nDCG) was selected to capture the importance of ranking, by measuring whether or not the most relevant candidate appeared at the top of the system ranking list.
Finally, performance was assessed in terms of processing time, both for individual resumes, namely the Processing Layer phase, OCR, Extraction, Vectorization, and end-user queries submitted through the frontend interface.

6.3 Evaluation Workflow

Evaluation was created as "Phase 5" of the system, an unofficial phase outside of the pipeline, with the purpose to benchmark the retrieval and ranking of the queries, with known recruiter decisions. The workflow is as follows:

1. Run phase 0 to 3 (Assuming it was not done before);
2. Load ground truth file;

3. Run phase 4 (RAG Query);
4. Parse candidate identifiers;
5. Compare retrieved candidates with ground truth set;
6. Compute metrics and save results to JSON for visualization.

In order to test if the system was retrieving the correct candidates, a ground truth file was developed, it was stored as a JSON file which was created outside of the workflow pipeline, as mentioned before, it is manually called by the system administrator/developer for evaluation and it is never affected by resume ingestion, extraction or vectorization. This in turn enables that the evaluation remains unbiased and reproducible. This file can be seen on Listing 6.1. The contents of this file are solely the query to be tested, and it there is also an array of candidates match that query despcription, and these candidates should be returned if system works as it should.

```
{
  "query_java_fullstack": {
    "query": "fullstack profile with 5+ years of experience working with
     Java 17+, spring and Angular 10+. Optional someone with azure
    experience",
    "relevant_candidates": ["Sofia Ribeiro"]
  }
}
```

Listing 6.1: Ground Truth file

Following this, the script fetches the ground truth file, and executes the RAG query, to compare rankings and compute metrics, as seen on Listing 6.2 and the Listing 6.3 respectively.

```
def parse_candidate_ids_from_rag_output(rag_output: str) -> list[str]:
    candidate_ids = re.findall(r'\b\w+_\w+\b', rag_output)
    seen = set()
    return [id for id in candidate_ids if not (id in seen or seen.add(id
    ))]
```

Listing 6.2: Evaluation Script snippet

```
rag_output = run_rag_query(query_text)
predicted_ids = parse_candidate_ids_from_rag_output(rag_output)
metrics = evaluate_predictions(predicted_ids, ground_truth_ids)
```

Listing 6.3: Evaluation Script predictions snippet

Afterwards, the calculated metrics are saved into a JSON to be visualize afterwards, as seen on Listing 6.4.

```
{
    "dataset_type": "manual",
    "results_per_query": {
        "query_devops": {
            "precision": 1,
            "recall": 1,
            "f1": 1,
            "ndcg": 1
        }
    }
```

Listing 6.4: Output of Evaluation Metrics

6.4 Results

Taking into account the experimental setup mentioned beforehand, which were being ran in a local machine, the results from the synthetic dataset showcased that the system was able to ingest and process 500 resumes without error, with an average processing time of one minute per resume per stage. Including OCR, LLM extraction and Vectorization stages. Moreover, recruiter queries returned results in around twenty seconds. In addition, with the manually authored resumes confirmed the robustness of the system, although some degradation in extraction quality was noted when documents contained severe formatting or mixed languages. Nonetheless, the system was able to perform and produce structured outputs and rankings in line with earlier expectations.

Furthermore, there was also another evaluation to the system where the most important results came from. Using the anonymized resume dataset, with the ten resumes. In a demonstration for the company Xcellence, the system was challenged to select two candidates for interview, from the pool of ten candidates. This test was ran three times, live and in-person, with slight variations done by the recruiter when querying the system. When tested against the 10-resume dataset, the system retrieved the same two candidates that the recruiter had previously chosen, three times in a row. The system was completely blind and had no historical data of who the recruiter had chosen for interview in the past. In other words, the system reproduced the recruiter decisions with maximum accuracy. This signified a 100% precision and 100% recall, granting an F1-Score of 1.0, and a nDCG score of 1.0. As for processing time, the query time was faster by roughly 10 seconds than the synthetic dataset, and the resume processing time in the Processing Layer took roughly 50 seconds per resume. These results can be seen in Table 6.1, and visually on Figures 6.1, 6.2, 6.3, and 6.4. The graphs aforementioned showcase the following, the nDCG curve (Figure 6.1) shows that the relevant candidates are constantly ranked at the top of the retrieved candidates, which confirms the effectiveness of the retrieval mechanism. The F1-score graph (Figure 6.2) showcases that the system has achieved a strong balance between precision and recall, even when non-standard format resumes are mixed in from the manually-created dataset. On the Precision graph (Figure 6.3) it is possible to see that precision remains high across all datasets (similarly to the other metrics), which demonstrate the ability of the system to minimize false-positives. Lastly, while observing the Recall graph (Figure 6.4) it is shown that the relevant candidates are not excluded, which allows for maintaining inclusivity on the ranking process. In sum, all these metrics helped validate the artifact across all three datasets, the datasets being, synthetic, manually-created and real-world anonymized dataset, and the system showcases a optimal semantic-aware context recognition and operational reliability thanks to the accuracy of the system and the performance under stress-testing.

Table 6.1: Evaluation results across datasets

Dataset	Precision	Recall	F1-Score	nDCG	Query Time
Synthethic Resumes	0.85	0.86	0.85	0.86	11.9s
Manual Resumes	0.87	0.89	0.88	0.88	12.2s
Anonymized Resumes	1.00	1.00	1.00	1.00	10.5s

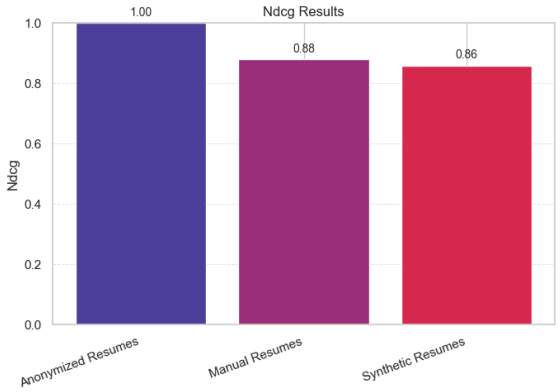


Figure 6.1: NDCg metric on general results

6.5 Summary

On this chapter, the evaluation and discussion of the constructed artifact was presented. This evaluation was carried out in a local development environment and also from a live demonstration with recruiters at Xcellence. The results gathered showcased that the system is capable of efficiently maintain stability across all datasets and further demonstrated accuracy to reproduce recruiter decisions. This demonstrates that the functional and non-functional requirements from Chapter 4 were met. Functionally, the system ingests, processes and ranks resumes as requested, while also producing justifications in outputs when candidates are retrieved for selection. Non-functionally, the system demonstrated performance reliability under stress-testing, and showcased good robustness against resumes in a non-standard format, while also ensuring the privacy of the real-world candidates through anonymization, and overall usability via both user-interface and command-line interface. In conclusion, this chapter showcased that the system achieved an alignment with the recruiter expectations and could perform reliabily under varied conditions, and these results also showcases that the LLM-Powered Resume Matcher is an improvement over traditional ATS approaches.

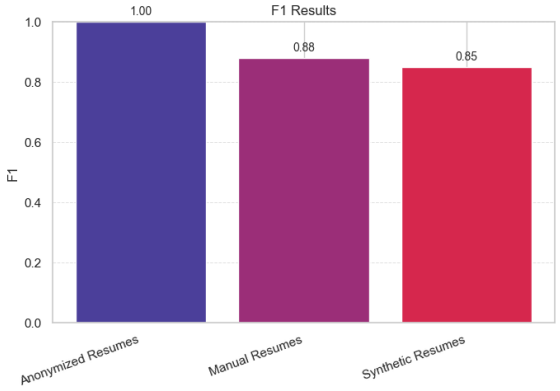


Figure 6.2: F1 metric on general results

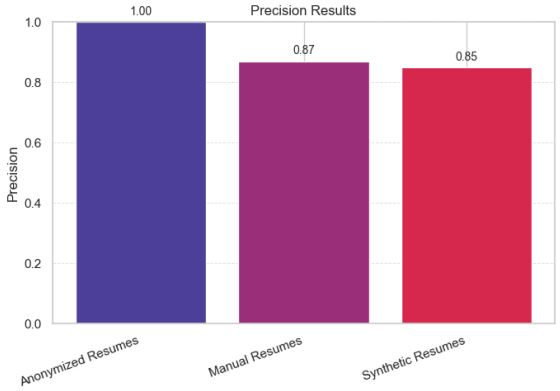


Figure 6.3: Precision metric on general results

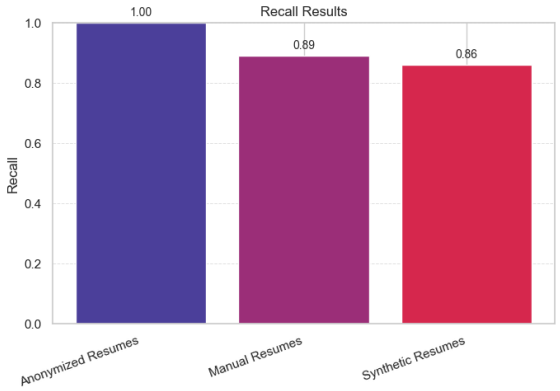


Figure 6.4: Recall metric on general results

Chapter 7

Conclusion

The work developed in this dissertation set out to design and evaluate a LLM-Powered Resume Matcher to address the inefficiencies and rigidness of a traditional ATS. The primary goal was to develop an artifact (system) that could ingest candidate resumes, extract and structure their content and provide ranked candidates to recruiter queries written in natural language. By using research methods such as DSR and CRISP-DM, it allowed for the design, implementation and evaluation of this artifact. The system was implemented as a modular pipeline which combine document storage, metadata tracking, text extraction, semantic vectorization and retrieval augmented generation. Furthermore, the system was implemented on a local machine, without internet connection, and only used open-source technologies. The evaluation confirmed that the system met the functional and non-functional requirements, and on the most decisive test case, it reproduced recruiter decisions with complete accuracy, by selecting exactly the same candidates that the recruiter had chosen for an interview. The results and the general evaluation showcased that the system met both functional requirements, namely, resume ingestion, resume structured extraction, semantic-aware retrieval and a ranked list of the most qualified candidates for a specific job opening produced by a query given by a recruiter. While also complying with the non-functional requirements, such as, ensuring system robustness against non-standard resume formats, which it was observed in the literature review that this type of weakness can worsen the accuracy of the system. Furthermore assuring efficiency under stress-testing, having the option and usability for both command-line interface with the LLM and also via user-interface to query the LLM. Overall, this positions the system not only as a proof-of-concept but also as a tangible improvement over currently existing ATS.

Despite these achievements, the system also possesses limitations. First and foremost the evaluation of the system was conducted in a controlled, developer environment, and not on a enterprise-scale production environment. Thus, even though the results gathered on Chapter 6 are optimal, they do not reflect on questions such as system maintenance, or scalability guided by an expansion of demand in resources. Secondly, while still on topic of evaluation, rigorous metrics evaluation was carried out for metrics of Precision, Recall, F1-Score and nDCG, which allowed for the measurement of robustness in the system and accuracy in retrieving the correct candidates. However, this evaluation did not include fairness, bias, or ethical auditing, due to the sole purpose of the main goal being aligning with the decisions of a typical recruiter in selecting the most qualified candidate, and also due to the fact that the tests and implementation carried out was always in a controlled environment, which ensured that in no possible way any sensible data was compromised. This was achieved thanks to the fact that most data was synthetic, and the real data was anonymized by Xcellence before it was ingested into this system.

Furthermore, although the system design is modular, it depends on open-source technologies that are not all used by Xcellence, with the exception of MinIO which enables for AWS S3 compatibility as mentioned beforehand.

These limitations demonstrate a reason for future work, such as, incorporating fairness and bias auditing into the evaluation stage, to ensure that recommendations remain fair across demographic groups. Another future work idea is to scale the datasets into an enterprise-level large-scale, to compare the results of this system in a new scenario. Another idea is to re-vamp the system by replacing the current technologies employed, with the technologies used at Xcellence.

Additionally, a real-world deployment in collaboration with Xcellence to test for compatibility and interoperability with existing HR ecosystems, namely what softwares do they use on their daily basis besides the traditional ATS. In conclusion, the system developed in this dissertation has demonstrated that LLM-Powered Resume Matchers can address the downsides of the current generic ATS used in the industry. This system developed stands as a proof-of-concept and foundation for further research, to offer a path for more accurate, ethical and transparent technologies to aid the recruitment process.

Bibliography

Abisha, D. et al. (2024). "Resspar: AI-Driven Resume Parsing and Recruitment System using NLP and Generative AI - 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)". In: pp. 1–6.

Afzal, Shehzad et al. (Mar. 2023). "Visualization and Visual Analytics Approaches for Image and Video Datasets: A Survey". In: *ACM Trans. Interact. Intell. Syst.* 13.1. issn: 2160-6455. doi: 10.1145/3576935. url: https://doi.org/10.1145/3576935.

Alderham, A. H. and E. S. Jaha (2022). "Comparative Semantic Resume Analysis for Improving Candidate-Career Matching - 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)". In: pp. 313–321.

Amazon Web Services (2025). *Amazon S3 – Cloud Object Storage*. https://aws.amazon.com/s3/. Accessed: 22 September 2025.

Artajaya, H. et al. (2024). "Job Recommendation System based on Resume using Natural Language Processing and Distance-based Algorithm - 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)". In: pp. 1–6.

Bartl, Marion et al. (Feb. 2025). "Gender Bias in Natural Language Processing and Computer Vision: A Comparative Survey". In: *ACM Comput. Surv.* 57.6. issn: 0360-0300. doi: 10.1145/3700438. url: https://doi.org/10.1145/3700438.

Caton, Simon and Christian Haas (Apr. 2024). "Fairness in Machine Learning: A Survey". In: *ACM Comput. Surv.* 56.7. issn: 0360-0300. doi: 10.1145/3616865. url: https://doi.org/10.1145/3616865.

Chivukula, Shruthi Sai et al. (Sept. 2024). "Surveying a Landscape of Ethics-Focused Design Methods". In: *ACM J. Responsib. Comput.* 1.3. doi: 10.1145/3678988. url: https://doi.org/10.1145/3678988.

Correll, Shelley J., Stephen Benard, and In Paik (2007). "Getting a Job: Is There a Motherhood Penalty?" In: *American Journal of Sociology* 112.5, pp. 1297–1338. doi: 10.1086/511799.

Dresch, Aline, Daniel Pacheco Lacerda, and José Antônio Valle Antunes (2015). "Design Science Research". In: 1st ed. Springer International Publishing. doi: 10.1007/978-3-319-07374-3_4.

Erdem, M. E. and R. Bayraktar (2023). "Layout Analysis for Robust Resume Parsing - 2023 6th International Conference on Information and Communications Technology (ICOIACT)". In: pp. 476–480.

Fabris, Alessandro et al. (Jan. 2025). "Fairness and Bias in Algorithmic Hiring: A Multidisciplinary Survey". In: *ACM Trans. Intell. Syst. Technol.* 16.1. issn: 2157-6904. doi: 10.1145/3696457. url: https://doi.org/10.1145/3696457.

FastAPI (2025). *FastAPI*. https://github.com/fastapi. Accessed: 22 September 2025.

Figueroa, Marco (July 2025). *Phishing For Gemini*. https://0din.ai/blog/phishing-for-gemini. 0DIN blog — research / security. Accessed: 22 September 2025.

Giri, A. et al. (2024). "Revolutionizing Recruitment with Large Language Models: A Multimodal AI Framework Integrating Video, Social Media, and Traditional Screening for

Efficient Talent Acquisition - 2024 First International Conference on Data, Computation and Communication (ICDCC)". In: pp. 244–250.

González, M. José, Clara Cortina, and Jorge Rodríguez-Menés (2019). "The Role of Gender Stereotypes in Hiring: A Field Experiment". In: *European Sociological Review* 35.2, pp. 187–204. doi: 10.1093/esr/jcy055.

Greco, Salvatore et al. (Nov. 2024). "NLPGuard: A Framework for Mitigating the Use of Protected Attributes by NLP Classifiers". In: *Proc. ACM Hum.-Comput. Interact.* 8. doi: 10.1145/3686924. url: https://doi.org/10.1145/3686924.

Guan, Zhihao et al. (Dec. 2024). "JobFormer: Skill-Aware Job Recommendation with Semantic-Enhanced Transformer". In: *ACM Trans. Knowl. Discov. Data* 19.1. issn: 1556-4681. doi: 10.1145/3701735. url: https://doi.org/10.1145/3701735.

Gupta, Manish and Puneet Agrawal (Jan. 2022). "Compression of Deep Learning Models for Text: A Survey". In: *ACM Trans. Knowl. Discov. Data* 16.4. issn: 1556-4681. doi: 10.1145/3487045. url: https://doi.org/10.1145/3487045.

Hachicha, M. et al. (2024). "Recommendation Module Based on Web Scraping and LLM Models with the RAG Technique - 2024 Artificial Intelligence Revolutions (AIR)". In: pp. 68–73.

Harsha, T. M. et al. (2022). "Automated Resume Screener using Natural Language Processing(NLP) - 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)". In: pp. 1772–1777.

Hourany, J. et al. (2023). "Learning Résumé Embeddings with Search Data and Transformers - 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)". In: pp. 2979–2983.

Kassianik, Paul (Dec. 2023). *Using AI to Automatically Jailbreak GPT-4 and Other LLMs in Under a Minute*. https://blogs.cisco.com/security/using-ai-to-automatically-jailbreak-gpt-4-and-other-llms-in-under-a-minute. Cisco Blogs, "Security". Accessed: 22 September 2025.

Lappas, Theodoros (May 2020). "Mining Career Paths from Large Resume Databases: Evidence from IT Professionals". In: *ACM Trans. Knowl. Discov. Data* 14.3. issn: 1556-4681. doi: 10.1145/3379984. url: https://doi.org/10.1145/3379984.

Li, Jingyang and Guoqiang Li (Feb. 2025). "Triangular Trade-off between Robustness, Accuracy, and Fairness in Deep Neural Networks: A Survey". In: *ACM Comput. Surv.* 57.6. issn: 0360-0300. doi: 10.1145/3645088. url: https://doi.org/10.1145/3645088.

Luo, Y. et al. (2018). "ResumeNet: A Learning-Based Framework for Automatic Resume Quality Assessment - 2018 IEEE International Conference on Data Mining (ICDM)". In: pp. 307–316.

Mhatre, S. et al. (2023). "Resume Screening and Ranking using Convolutional Neural Network - 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)". In: pp. 412–419.

Microsoft (2025). *Chunk large documents for vector search solutions in Azure AI Search*. https://learn.microsoft.com/en-us/azure/search/vector-search-how-to-chunk-documents. Accessed: 22 September 2025.

MinIO, Inc. (2025). *MinIO: High Performance Object Storage*. https://www.min.io/. Accessed: 22 September 2025.

Mohanty, S. et al. (2023). "Resumate: A Prototype to Enhance Recruitment Process with NLP based Resume Parsing - 2023 4th International Conference on Intelligent Engineering and Management (ICIEM)". In: pp. 1–6.

Parraga, Otavio et al. (Feb. 2025). "Fairness in Deep Learning: A Survey on Vision and Language Research". In: *ACM Comput. Surv.* 57.6. issn: 0360-0300. doi: 10.1145/3637549. url: https://doi.org/10.1145/3637549.

Perez-Cerrolaza, Jon et al. (Apr. 2024). "Artificial Intelligence for Safety-Critical Systems in Industrial and Transportation Domains: A Survey". In: *ACM Comput. Surv.* 56.7. issn: 0360-0300. doi: 10.1145/3626314. url: https://doi.org/10.1145/3626314.

Poppe, Olga et al. (Feb. 2022). "Moneyball: proactive auto-scaling in Microsoft Azure SQL database serverless". In: *Proc. VLDB Endow.* 15.6, pp. 1279–1287. issn: 2150-8097. doi: 10.14778/3514061.3514073. url: https://doi.org/10.14778/3514061.3514073.

PyMuPDF (2025). *PyMuPDF Documentation*. https://pymupdf.readthedocs.io/en/latest/. Accessed: 22 September 2025.

AL-Qassem, Amer Hani et al. (Mar. 2023). "Leading Talent Management: Empirical investigation on Applicant Tracking System (ATS) on e-Recruitment Performance - 2023 International Conference on Business Analytics for Technology and Security (ICBATS)". In: pp. 1–5. doi: 10.1109/ICBATS57792.2023.10111172.

R, G. M. G., S. Abhi, and R. Agarwal (2023). "A Hybrid Resume Parser and Matcher using RegEx and NER - 2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)". In: pp. 24–29.

Sentence-Transformers (2025a). *paraphrase-multilingual-MiniLM-L12-v2*. https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2. Accessed: 22 September 2025.

– (2025b). *paraphrase-multilingual-MiniLM-L12-v2*. Accessed: 22 September 2025. url: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2.

Shen, Dazhong et al. (Sept. 2021). "Joint Representation Learning with Relation-Enhanced Topic Models for Intelligent Job Interview Assessment". In: *ACM Trans. Inf. Syst.* 40.1. issn: 1046-8188. doi: 10.1145/3469654. url: https://doi.org/10.1145/3469654.

Skondras, P. et al. (2023). "Efficient Resume Classification Through Rapid Dataset Creation Using ChatGPT - 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)". In: pp. 1–5.

Smuha, Nathalie A., ed. (2025). *The Cambridge Handbook of the Law, Ethics and Policy of Artificial Intelligence*. Cambridge Law Handbooks series. Cambridge University Press. isbn: 9781009367813. doi: 10.1017/9781009072168.

SQLite (2025). *SQLite — Home Page*. https://sqlite.org/. Accessed: 22 September 2025.

Srivastava, M. and P. Greaney (2023). "Resume Parsing Across Multiple Job Domains Using a BERT-Based NER Model - 2023 31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS)". In: pp. 1–4.

Streamlit Inc. (2025). *Streamlit — The fastest way to build data apps in Python*. https://streamlit.io/. Accessed: 22 September 2025.

Su, Y., J. Zhang, and J. Lu (2019). "The Resume Corpus: A Large Dataset for Research in Information Extraction Systems - 2019 15th International Conference on Computational Intelligence and Security (CIS)". In: pp. 375–378.

Sunico, R. J. et al. (2023). "Resume Building Application based on LLM (Large Language Model) - 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)". In: pp. 486–492.

Tesseract-OCR (2025). *Tesseract*. https://github.com/tesseract-ocr/tesseract. Accessed: 22 September 2025.

TryChroma (2025). *TryChroma — Data AI Tools*. https://www.trychroma.com/. Accessed: 22 September 2025.

Vaishampayan, Swanand, Sahar Farzanehpour, and Chris Brown (Oct. 2023). "Procedural Justice and Fairness in Automated Resume Parsers for Tech Hiring: Insights from Candidate Perspectives - 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)". In: pp. 103–108. issn: 1943-6106. doi: 10.1109/VL-HCC57772.2023.00019.

Venkatakrishnan, R. et al. (2024). "Revolutionizing Talent Acquisition: A Comparative Study of Large Language Models in Resume Classification - 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)". In: pp. 1–6.

Warusawithana, S. P. et al. (2023). "Layout Aware Resume Parsing Using NLP and Rule-based Techniques - 2023 8th International Conference on Information Technology Research (ICITR)". In: pp. 1–5.

Wirth, R. and J. Hipp (2000). "CRISP-DM: Towards a Standard Process Model for Data Mining". In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining (PADD)*. Vol. 1. Springer. Chap. 0, pp. 29–39. url: https://www.the-modeling-agency.com/crisp-dm.pdf.

Xiong, Peiyu et al. (Apr. 2024). "It Is All about Data: A Survey on the Effects of Data on Adversarial Robustness". In: *ACM Comput. Surv.* 56.7. issn: 0360-0300. doi: 10.1145/3627817. url: https://doi.org/10.1145/3627817.

Yang, Chuanpeng et al. (Oct. 2024). "Survey on Knowledge Distillation for Large Language Models: Methods, Evaluation, and Application". In: *ACM Trans. Intell. Syst. Technol.* issn: 2157-6904. doi: 10.1145/3699518. url: https://doi.org/10.1145/3699518.

Zhang, Chen and Hao Wang (Nov. 2018). "ResumeVis: A Visual Analytics System to Discover Semantic Information in Semi-structured Resume Data". In: *ACM Trans. Intell. Syst. Technol.* 10.1. issn: 2157-6904. doi: 10.1145/3230707. url: https://doi.org/10.1145/3230707.

Zühlke, Monty-Maximilian and Daniel Kudenko (Feb. 2025). "Adversarial Robustness of Neural Networks from the Perspective of Lipschitz Calculus: A Survey". In: *ACM Comput. Surv.* 57.6. issn: 0360-0300. doi: 10.1145/3648351. url: https://doi.org/10.1145/3648351.

Appendix A

Planning

This dissertation was supported by a detailed planning, with the purpose of keeping track of time and completing the objectives. The planning was made of four parts, it consisted of a Gantt Chart which came in two parts, first for PREPD class, and the other for DIMEI class. Secondly a Work Breakdown Structure (WBS) graph of the work breakdown needed to fulfill the objectives, and a Skill Analysis diagnostic. Lastly, a Risk register to take into consideration possible risks and crisis that could happen and would affect negatively the development of the project.

A.1 Project Scope

The goal of this project is to design, develop, implement and evaluate a LLM-Powered Resume Matcher, capable of accelerating recruitment processes and selecting the most qualified candidates. In order to achieve these goals, several objectives are defined, the first is to conduct a State of The Art review to establish the current scientific and technological knowledge about recruitment systems and algorithmic-hiring, and LLMs. The second objective is to establish a rigorous methodological framework to guide this dissertation in a consistent theoretical way but also accompanied by a reproducible technical workflow. Consequently, to design and implement the system architecture, in order to demonstrate the functionality of the system through practical experiments and evaluating it with real metrics and to compare the system outputs with recruiter ground truth. Thus, allowing for the analysis to validate the accuracy and performance of the system on a real-world scenario. As such, and in order to plan according, a WBS Diagram was constructed, as seen on Figure A.1

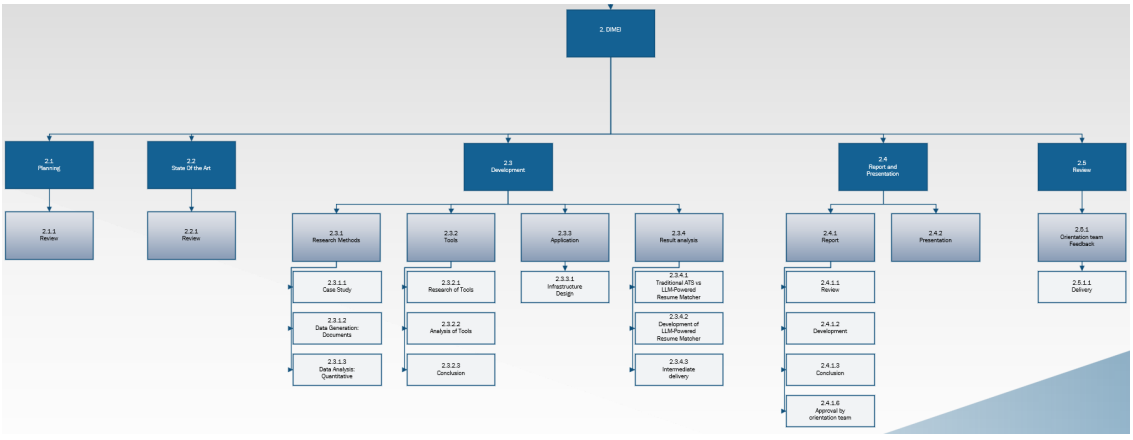


Figure A.1: Work Breakdown Structure

A.2 Project Schedule

After the creation of the WBS diagrams, In order to fulfill the dissertation timeline, there was a need to break down the entire work backlog, and to ponder how much time each stage would take. Furthermore, to add milestones for each project goal met. As such, there was a need for a tool that could help design and assist in developing a schedule, assigning resources to tasks, tracking progress and analyzing workloads. To that end, the tool selected was Microsoft Project. This tool allowed for tracking of the implementation of this dissertation. This project management Software was very useful because not only does it expand on what is established in WBS, but it also allows for the mechanisms such as "Predecessors" which are dependencies between other tasks, in other words, for instance Task B can only start after Task A is finished, and so on. And Microsoft Project also allows for the usage of "% Work complete" column which allows for the progress to be tracked overtime.

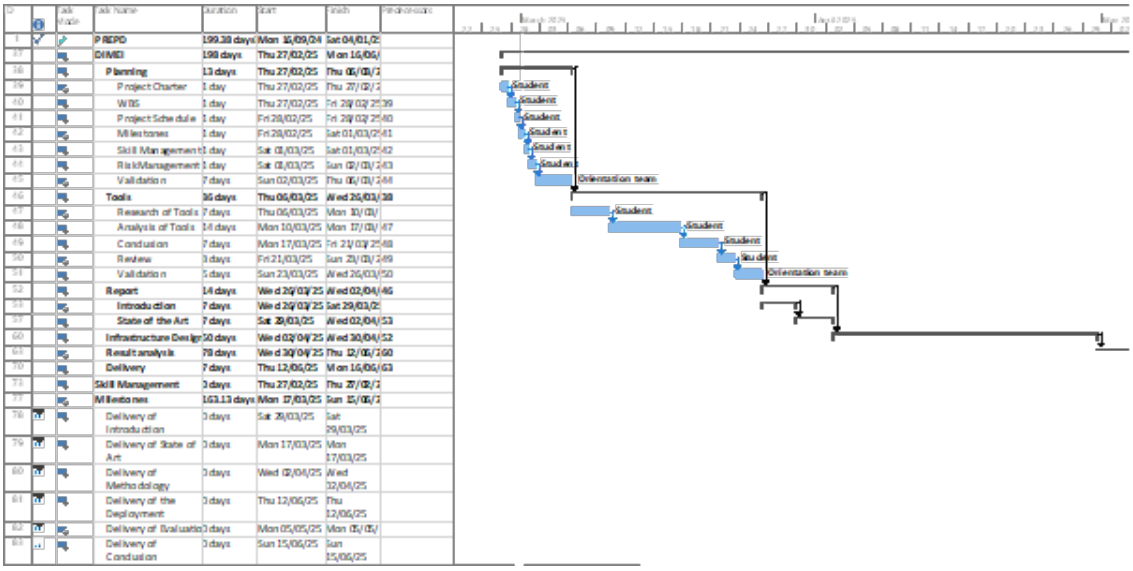


Figure A.2: Gantt Chart schedule

A.3 Skill Management

In order to tackle this challenge, a retrospection had to be done to better understand the qualities or skills that were good enough or should be developed in order to do a better work. As such, an extensive self-assessment was developed to study which skills should be improved. The skills tested are the following: Analysis and problem solving; Lifelong learning; Teamwork and collaboration; Motivation for Excellence; Ethics and social responsibility; Proficiency in foreign languages; Technical skills in the specific area of knowledge; Decision making; Time Management; Ability to take risks; Oral communication; Resilience; Active Listening; Interpersonal Relationships; Leadership; Creativity and Innovation; Adaptation and flexibility; Critical Thinking; Planning and organization; Emotion management; Empathy; Assertiveness; Stress Management; Proactivity. Conversely, to assure that the self-assessment was fair and not biased, a peer-assessment was made. Furthermore, to completely make sure that this peer-assessment was fair and unbiased, the peer who evaluated my skill had no access to the self-assessment that I had done beforehand. Now that the self-assessment and peer-assessment were complete, the next step was to find which were

the most and less developed skills overall. Starting with the most developed skills: Teamwork and collaboration; Ethics and social responsibility; Ability to take risks; Resilience; Active listening; Adaptation and flexibility; Critical thinking; Emotion management; Empathy. The less developed skills were as follows: Time management; Oral communication; English language. Since the main focus of this skill management plan was to find the least developed skills, more planning went into consideration. As to how those lacking skills could be developed. As such, a development plan was created. In Table **??** it is showcased the lacking skills and how they will be improved.

Table A.1: Personal Development Plan

Skill	Development Plan
Time Management	Taking a short course about Time Management; Compromising to work at least 2 hours daily on this project.
Oral Communication	Participate in presentations; Participate in speeches.
English Language	Reading English books; Taking short courses.

A.4 Risks Analysis

With the objective of maximizing the chances of success of this project, a risk management section was added into the planning. The risks that may exist in this project were logged in the risk register, as seen on Table A.3. The Risk Register is composed of four different risks.

The first risk is **"No access to development tools or infrastructure"**, the cause being to third party issues or maintenance downtimes. With a PI score of 5, and if no action is taken it will not be possible to secure the project timeline. Therefore, to mitigate the risk, the solution was set to having backup tools available and showing a demonstration for the required infrastructure.

The second risk is **"Data receives is not according with the requirements"**, with the cause being lack of data, whether it is arriving it is arriving on the wrong format, or in the scenario it did not arrive at all. With a PI score of 15, the highest score on this study. Without intervention it will not be possible to analyze the data. The solution created to mitigate this was to investigate and find other available datasets.

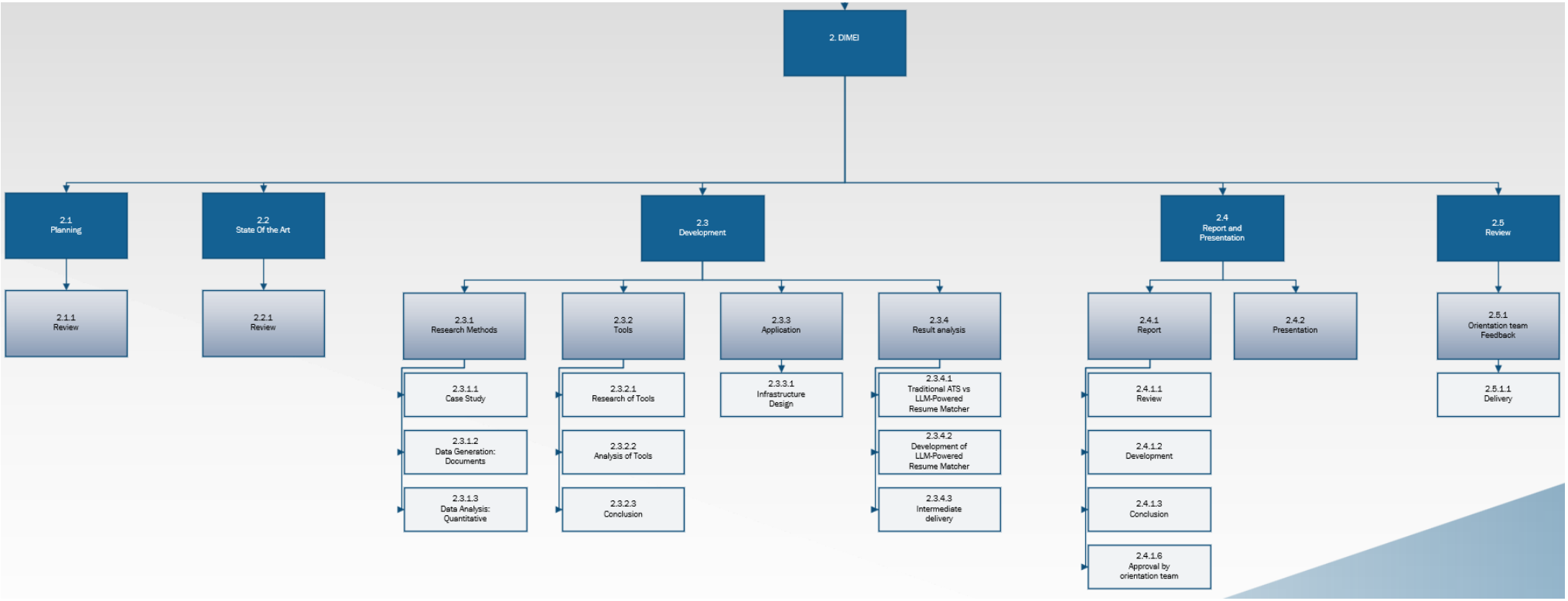
The third risk is **"Bad system performance"**, it is caused by a lack of hardware computing power. With a PI score of 4, and if left unchecked it will prevent the publication of this work, as if the system cannot be demonstrated then it cannot be developed. Consequently, the project timeline will not be secured, to mitigate this a proper computing infrastructure must be available.

The fourth and final risk is **"Evaluation validity risk"**, caused by potentially a bad implementation of this project, with a PI score of 10. Which is the second highest on all these risks being analyzed. If this risk is left unchecked it will lead to being unable to reach objectives set on this project. Thus, to mitigate this, the design of the code must be well-thought and tested frequently.

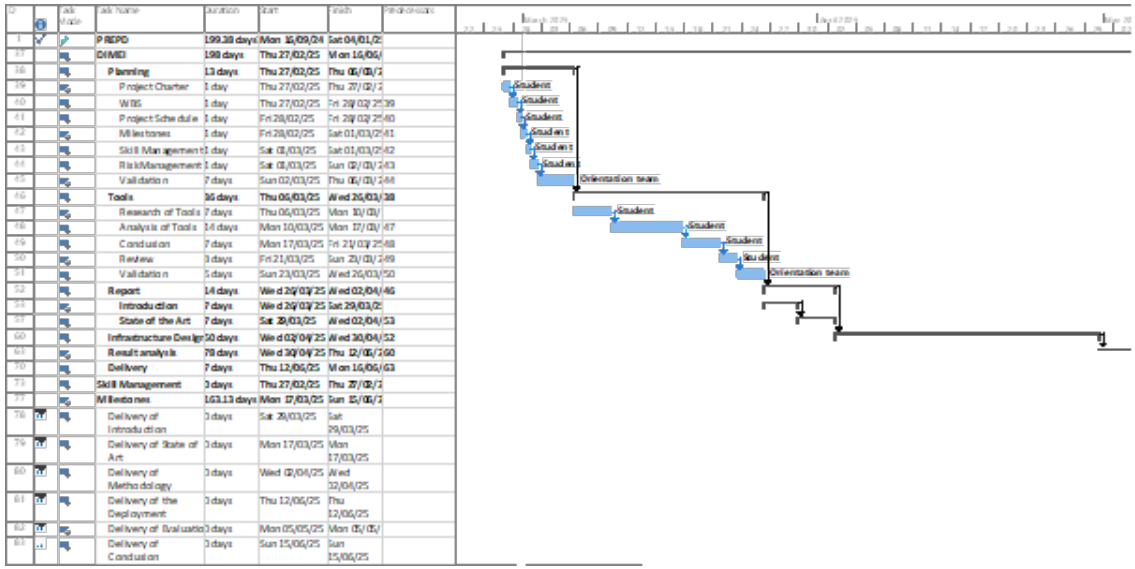
Description	Cause	Effect	Risk Owner	Probability (1-5)	Impact (1-5)	PI Score	Expected Result, No Action	Risk Response Type	Response description
Description of the risk	**Cause of the risk**	**Effect on the project**	**Name of person who monitors the risk**	**Group sourced rough estimate of how likely this is to occur**	**Rough estimate of how significant the impact of this risk**	**Probability multiplied by Impact**	**What will happen if the risk becomes an issue and no action is taken**	**Decision made by group on how to respond to this risk**	**How do you know it is time to put the response into play**
No access to development tools or infrastructure	Third party Maintenance	Delayed timeline	Júlio Rocha	1	5	5	Will not be able to secure the timeline	Mitigate	When development tools become unavailable, or pipeline cannot be executed
Data received not according with the requirements	Lack of data	Not able to analyse the data	Júlio Rocha	3	5	15	Not able to analyse the data	Mitigate	When validation script flag is missing or has a invalid field
Bad system performance	Limited computational resources	Delayed timeline	Júlio Rocha	1	4	4	Will not be able to secure the timeline	Mitigate	When the resume ingestion exceeds 60 seconds
Evaluation validity risk	Small dataset	Delayed timeline	Júlio Rocha	2	5	10	Will not be able to secure the timeline	Mitigate	When evaluation metrics plateau

Figure A.3: Risk Register

Appendix B — WBS



Appendix C — Gantt Chart



Appendix D — Risk Register

Description	Cause	Effect	Risk Owner	Probability (1-5)	Impact (1-5)	PI Score	Expected Result, No Action	Risk Response Type	Response description
Description of the risk	**Cause of the risk**	**Effect on the project**	**Name of person who monitors the risk**	**Group sourced rough estimate of how likely this is to occur**	**Rough estimate of how significant the impact of this risk**	**Probability multiplied by Impact**	**What will happen if the risk becomes an issue and no action is taken**	**Decision made by group on how to respond to this risk**	**How do you know it is time to put the response into play**
No access to development tools or infrastructure	Third party Maintenance	Delayed timeline	Júlio Rocha	1	5	5	Will not be able to secure the timeline	Mitigate	When development tools become unavailable, or pipeline cannot be executed
Data received not according with the requirements	Lack of data	Not able to analyse the data	Júlio Rocha	3	5	15	Not able to analyse the data	Mitigate	When validation script flag is missing or has a invalid field
Bad system performance	Limited computational resources	Delayed timeline	Júlio Rocha	1	4	4	Will not be able to secure the timeline	Mitigate	When the resume ingestion exceeds 60 seconds
Evaluation validity risk	Small dataset	Delayed timeline	Júlio Rocha	2	5	10	Will not be able to secure the timeline	Mitigate	When evaluation metrics plateau