Republic of Tunisia
Ministry of Higher Education
and Scientific Research

University of Sfax

National School of Electronics
and Telecommunications of Sfax

Engineer in :
Electronic Systems Engineering
and Communication

Option :
Data Engineering and Decision Support
Systems

Dissertation

Presented at

National School of Electronics and Telecommunications of Sfax

in order to obtain the

National Engineering Diploma in :
Electronic Systems and Communication Engineering

Option:
Data Engineering and Decision Support Systems

by

Hammoudi Wissem

Resume Screening Application

Acknowledgments

I would like to extend my deepest gratitude to my internship supervisor, Mr. Sofien Mhatli, for his warm welcome, the time we spent together, and his daily sharing of expertise. His trust and support allowed me to fully achieve my goals and excel in my assignments.

I also wish to thank the entire team for their hospitality and team spirit, which greatly contributed to my experience and success.

Finally, I am profoundly grateful to my loved ones for their unwavering support in developing my professional project and for their help throughout the preparation of this internship report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

RAG	Retrieval Augmented Generation
CRISP-DM	Cross-Industry Standard Process for Data Mining
TDSP	Team Data Science Process
LLM	Large Language Model
NLP	Natural Language Processing
NER	Named Entity Recognition
API	Application Programming Interface
RRF	Reciprocal Rank Fusion
HR	Human Resources
SSD	Single Shot MultiBox Detector
CNN	convolutional neural networks

GENERAL INTRODUCTION

Resume screening is challenging for many organizations as it involves identifying only a few potentially matching candidates from a diverse applicant pool. With organizations receiving up to 250 applications on average for a job posting[1], this task becomes even more difficult due to the large volumes of candidates. However, the situation is that existing manual screening methods are usually labour-intensive and prone to biases. This is because they involve working with different recruiters with varying levels of experience and knowledge. This situation highlights the need for automated solutions that can perform resume screening more efficiently.

Automated systems like keyword matching have been used to improve screening efficiency but often rely on rigid, rule-based approaches that can introduce bias. More advanced methods, such as classification systems and similarity-based retrievers, have been explored but still struggle with the complexities of natural language in resumes.

The main issue with current screening methods is their inability to effectively handle the nuances of natural language, which can lead to misinterpreting or overlooking qualified candidates. To address this, there is a need for methods that better manage the complexities of natural language. Large Language Models (LLMs) offer a promising solution for these challenges.

In the first chapter, we will outline the project's scope and describe the problem along with existing solution gaps. The second chapter will cover the background of LLMs and Retrieval-Augmented Generation (RAG). In the realization section, we will discuss our methodology, including business understanding, data collection, information extraction, and deployment.

Chapter

1

Project Scope

Contents

1.1 INTRODUCTION

In this chapter, we place our project in its general context.We discuss the existing study and the limitations encountered then we highlight our proposed solution by describing our mission.

1.2 Problem Statement

1.2.1 Description of Existing Methods

At present, resume screening is performed manually by HR personnel who review and analyze each resume individually. This approach, while thorough, is limited by its reliance on human judgment and its inability to scale effectively. The lack of automation means that the process is slow and resource-intensive, and it can struggle with the diverse presentation styles of resumes.

1.2.2 Critique of Existing Solutions

The current solution relies on manual screening for resume evaluation, which presents several notable challenges:

- **Accuracy Issues:** Manual screening can lead to inconsistencies and errors in evaluating resumes due to human fatigue, bias, or oversight.
- **Processing Inefficiencies:** Manual screening is time-consuming and labor-intensive, making it impractical for handling large volumes of resumes efficiently.
- **Paper Waste:** Manual processes frequently involve printing physical copies of resumes, contributing to unnecessary paper consumption.
- **Limited Scalability:** As the volume of resumes increases, manual screening becomes even more impractical, resulting in delays and bottlenecks in the hiring process.

- **High Costs:** The manual evaluation of resumes demands significant time from HR professionals, which translates into higher costs for companies. As more time is dedicated to screening, this increases operational expenses, making the process less cost-effective.

1.2.3 Proposed Solution

The proposed solution involves a combination of a resume parser and a chatbot designed to optimize the recruitment process.

- **Resume Parser:** This component will leverage advanced text extraction techniques and Natural Language Processing (NLP) to accurately extract key details from resumes.
- **Chatbot:** The chatbot will interact with users to align resumes with job descriptions. It will evaluate resumes based on the specific job requirements, assisting in identifying the most suitable candidates for each position.

Together, these tools will streamline the recruitment process by automating both the extraction and matching of resumes, significantly reducing manual effort while enhancing the overall efficiency and accuracy of candidate evaluation.

1.3 Mission

The primary objectives of the resume parser and chatbot project are:

- Develop a resume parser that accurately extracts key information from various resume formats and layouts.
- Create a chatbot that can interact with users to match resumes with job descriptions, identifying the best candidates for specific positions.
- Improve the accuracy and effectiveness of candidate evaluations through reliable information extraction and job description alignment.

1.4 Conclusion

In this chapter, we have outlined the context of our project, offering a thorough overview of its background and objectives. We started by detailing the current state and challenges of resume screening processes. We then introduced our proposed solution, which includes a resume parser and chatbot aimed at improving the efficiency and accuracy of the recruitment process.

The next chapter will focus on the methodologies employed in our work and provide a review of the literature on Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG).

Chapter 2

Background

Contents

2.1 INTRODUCTION

In this chapter, we will begin by discussing the methodologies of work. Following that, we will explore the background and conduct a literature review on Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG). We will cover the fundamentals of LLMs, focusing on their transformer-based architecture. Next, we will introduce the RAG framework, which enhances LLMs by incorporating external knowledge sources. Finally, we will examine RAG Fusion, an advanced variant that optimizes the integration of retrieved knowledge into the generation process.

2.2 Methodologies

2.2.1 CRISP-DM

The CRISP-DM lifecycle consists of six phases, and their sequence is flexible, allowing for iterative movement between stages as needed. For example, if a model's accuracy is unsatisfactory, revisiting earlier phases like data understanding and preparation may be required to improve performance.

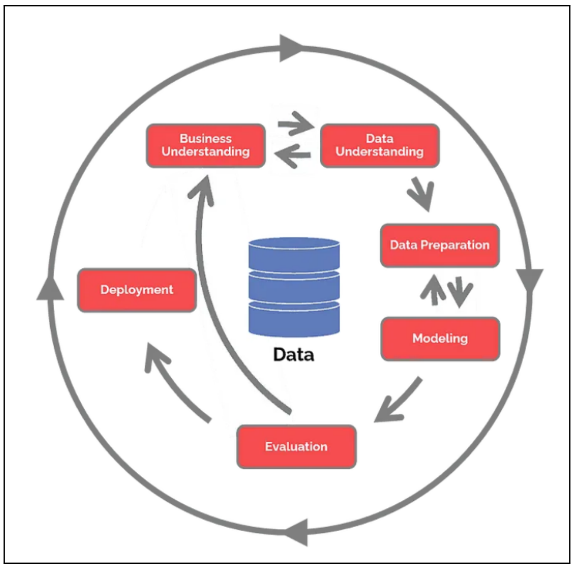


Figure 2.1: CRISP-DM process model.

Below, we briefly outline each phase of the CRISP-DM process:

- **Business problem understanding:** This initial phase focuses on understanding the project goals, objectives, and requirements from a business perspective, then converting this knowledge into a data mining problem definition and a preliminary plan with a specific set of tasks and desired outcomes to achieve project-level objectives.
- **Data Understanding:** Collect initial data and perform activities like feature description and exploratory analysis to identify data quality issues and form hypotheses.
- **Data Preparation:** The data preparation phase covers all activities needed to construct the final dataset fed into the modeling tools from the initial raw data. Data preparation tasks are likely to be performed multiple times and not in any prescribed order. Tasks include data table, feature selection, feature engineering, as well as feature transformation and cleaning of data for modeling tools and techniques.
- **Modeling:** In this phase, various modeling techniques depending on the problem statement, are selected and applied, and their parameters are calibrated to find optimal modeling performance. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements for the format of data it needs. Therefore, going back to the data preparation phase is often necessary at this stage.
- **Evaluation:** The evaluation phase is one of the most crucial phases of any data science project lifecycle. At this stage in your project, you must have built machine-learning models. Models might be performing well on your training data, but it is necessary to test and evaluate it on unseen data for the model to achieve its objectives. Appropriate evaluation metrics are measured and well-tested at this stage. At the end of this phase, a decision on using the data mining results should be reached.
- **Deployment:** The creation of the model is generally not the end of the project. Even if the purpose of the model is to analyze the data and increase the understanding of the data, the knowledge or insight gained from the modeling needs to be presented so that the end users of the model can use it.

2.2.2 TDSP

TDSP is an agile and iterative methodology to build and deploy predictive analytics solutions. It helps to improve teamwork. TDSP covers the many stages of the project and allows iterative execution with flexibility.

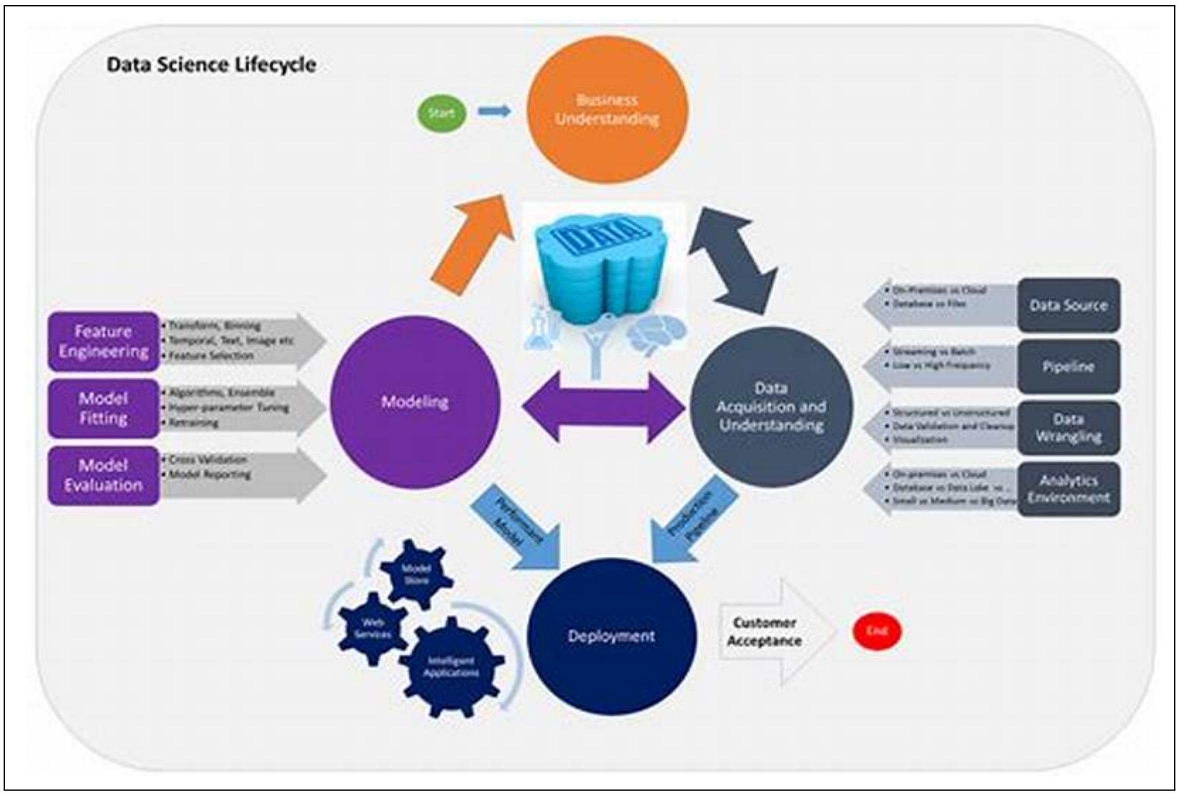


Figure 2.2: TDSP process model.

Below, we briefly outline each phase of the TDSP process :

- **Business understanding:** This stage includes building an understanding of the business problem, stakeholder needs, and the use case. It may require reading through the knowledge base and project documents, and meetings / interviews with the stakeholders.

- **Data Acquisition and Understanding:** This stage includes understanding and acquiring the data sources, pipeline, frequency, environments, and data structure. It further includes data exploration. It may also include data cleaning, wrangling, and transformation.

- **Modeling:** After Data Acquisition and Understanding stage comes the Data modeling stage. This stage includes Feature engineering steps such as Data transformation and feature selection. The next step is model training. It includes training the model and tuning its hyperparameters. The last step in this stage is model evaluation. It can be achieved via multiple methods such as cross-validation, etc.

- **Deployment:** The model deployment includes deploying and serving the model. For example, it might involve deploying on the cloud or integrating the model as part of an application. It would also include monitoring the performance of the model. During the monitoring phase, if any defects or trouble reports are generated, the team might have to iterate through some or all of the above stages and steps in the lifecycle.

- **Customer acceptance:** Once the defects have been fixed, and the model is deployed as per customer satisfaction, the project comes to an end. In some instances, models are continuously improved. In that case, after the first deployment, the team will iterate through the above lifecycle continuously or until all the project phases are completed.

2.3 Large Language Models

2.3.1 The Idea behind LLMS

2.3.1.1 The Transformer Architecture

A Large Language Model (LLM) is a probabilistic model trained on a massive corpus of text to capture the complex statistical relationships of natural language. While there are different models for various use cases, most LLMs are constructed based on the transformer architecture.

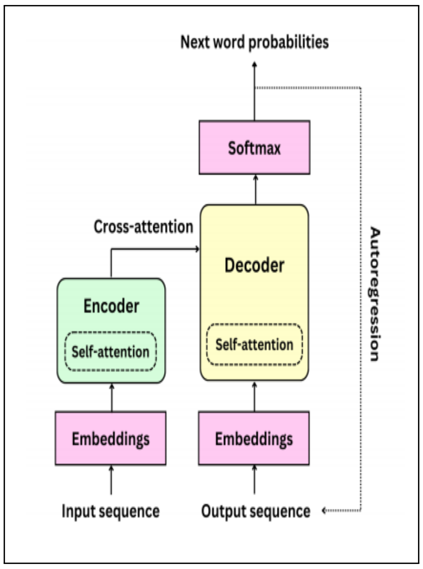


Figure 2.3: A high-level view of the transformer architecture

The transformer, as depicted in Figure 2.3, is designed to predict a sequence of the most probable next words given an input sequence. Overall,this process involves two main components, the encoder and the decoder.

Encoder

- **Purpose:** The encoder's job is to process the input sequence and convert it into a condensed representation, or vector. This vector captures the essence of the input data.
- **How It Works:** It uses a mechanism called self-attention to analyze each word in the input sequence in relation to every other word. This helps the encoder understand the context and significance of each word in the sequence.

Decoder

- **Purpose:** The decoder uses the vector from the encoder to generate the output sequence, one word at a time.
- **How It Works:** It starts with a special token that signals the beginning of the output sequence. At each step, the decoder predicts the next word by considering the words it has already generated and the context provided by the encoder. This is done through a process called autoregression, where each new word depends on the previous words and the context vector.

Key Mechanisms

- **Self-Attention:** Both the encoder and decoder use self-attention to evaluate each word in relation to others in the sequence. This helps the model focus on important words and their relationships. For example, in the phrase "The cat sat on the mat," self-attention helps the model understand that "cat" and "sat" are closely related.
- **Cross-Attention:** In addition to self-attention, the decoder uses cross-attention to access information from the encoder's output. This allows the decoder to combine the context from the encoder with the words it has generated so far.

Additional Techniques

Positional Encoding: Since the transformer model doesn't process words sequentially like older models, it uses positional encoding to keep track of the order of words. This helps the model understand the sequence of words in a sentence.

Layer Normalization and Residual Connections: These techniques help stabilize the learning process and make the model more robust. Layer normalization ensures that the model's output remains consistent, while residual connections help in managing the flow of information during training.

2.3.1.2 Application Ability of LLMs

Employing the transformer architecture, LLMs can understand the statistical patterns of the language they are trained on. This allows them to comprehend highly complex textual data as input while generating natural sounding and relevant responses. With this ability, LLMs can reach a general-purpose use without requiring task-specific programming, making them adaptable to many natural-language-related tasks.

For instance, in text summarization, LLMs can process a long document to generate a concise summary that retains the key points. This is achieved through the self-attention mechanism, which enables LLMs to extract the most important terms and dependencies in the original document to condense them into a summary. For translation tasks, LLMs can transform a sentence from one language to many others. This process involves encoding the features of the source language into context-rich embedding vectors. The decoder can then convert the encoded vectors into text in the target language while maintaining the semantic features.

Additionally, LLMs can be fine-tuned for specific applications by training them on specialized datasets. This approach allows them to adapt to particular domains, such as recruitment, with models like conSultantBERT, which improves accuracy for resume and job description data compared to generic models.

2.3.2 Vector Representation of Natural Language

Generating a vector representation of a text is one of the most common use cases of LLMs. It involves transforming a sequence of text to its numerical vector representation in high dimensional space so that machine learning models can understand. This is usually performed using the encoder part, which typically outputs a set of embedding vectors for a sentence. Each vector in this set carries the semantic information of a token to the overall input sentence. However, in practical usage, using the whole set of vectors for each token can result in great computational loss. Therefore, it is a common practice to transform them into a single vector representation for the whole input sequence. One of the standard methods is to "compress" the granular token-level vectors into one single vector, called pooling.

For example, models like BERT use a special token added at the beginning of a sentence. During training, the model learns to use the vector of this token to capture the overall meaning of the sentence by gathering contextual information from all other tokens. This special token effectively summarizes the sentence.

Sentence transformers are a type of model designed to produce these summary vectors at the sentence level, rather than just for individual tokens. They are trained to ensure that similar sentences have similar vectors, which is useful for tasks like semantic search and similarity. These models have become popular for their ability to capture and represent complex text information effectively.

2.3.3 Natural Language Generation

Text generation is a higher-level use case of LLMs but with no less importance. This task refers to generating responses to user queries about a particular context. This is achieved by processing the query into the encoder to capture the query's intent, then utilizing the decoder to synthesize relevant information from the LLM's pre-trained inherent knowledge base to generate answers. In essence, this process is similar to the generic inference process of an LLM, where the decoder attends to the encoder's representation of the input sequence (query) to output the most probable next words (answer) based on its pre-trained latent space.

Generative Pre-trained Transformer (GPT) is one such model often used for text generation tasks. Instead of focusing on understanding user queries like generic encoder-decoder LLMs, GPT models aim to produce more natural and coherent responses that can effectively carry the main idea of the relevant information. Therefore, the focus is driven towards the decoder part of the LLM, which is responsible for autoregressive next-word prediction, to enhance the generative ability. However, this does not imply a complete lack of attention to understanding the input queries in GPT models. Because the decoder is employed with the powerful self-attention mechanism, it can still effectively attend to the context of the input queries during the next-word generation process. In GPT models, the decoder alone is sufficient to comprehend complex sequences. Therefore, in many cases, the encoder block can be completely absent from the architecture of a GPT model. This allows more computational resource allocation to improve the decoder's generative ability.

GPT models often do not include an encoder, which is common in other models. This allows them to allocate more resources to improving text generation. However, GPT models have some limitations. They can struggle with accuracy and factual correctness because they rely on patterns learned during training rather than having complete knowledge. This can lead to "hallucinations," where the model produces incorrect or vague information. Additionally, without an encoder, GPT models may find it challenging to fully grasp complex or ambiguous queries from users.

2.3.4 AI Agent

While there isn't a widely accepted definition for LLM-powered agents, they can be described as a system that can use an LLM to reason through a problem, create a plan to solve the problem, and execute the plan with the help of a set of tools.

In short, agents are a system with complex reasoning capabilities, memory, and the means to execute tasks.This capability was first observed in projects like AutoGPT or BabyAGI, where complex problems were solved without much intervention.To describe agents a bit more, here's the general architecture of an LLM-powered agent application (Figure 2.4).

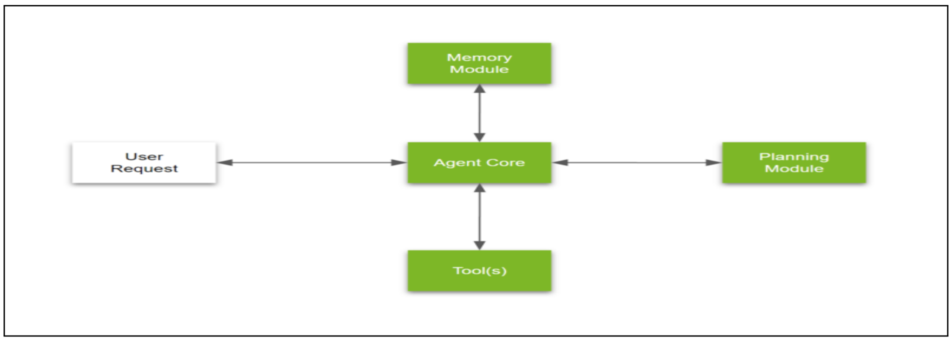


Figure 2.4: General components of an agent

An agent is made up of the following key components:

Agent core

The agent core is the central coordination module that manages the core logic and behavioral characteristics of an Agent. Think of it as the "key decision making module" of the agent. It is also where we define:

- **General goals of the agent:** Contains overall goals and objectives for the agent.
- **Tools for execution:** Essentially a short list or a "user manual" for all the tools to which the agent has access
- **Explanation for how to make use of different planning modules:** Details about the utility of different planning modules and which to use in what situation.
- **Relevant Memory:** This is a dynamic section which fills the most relevant memory items from past conversations with the user at inference time. The "relevance" is determined using the question user asks.
- **Persona of the Agent (optional):** This persona description is typically used to either bias the model to prefer using certain types of tools or to imbue typical idiosyncrasies in the agent's final response.

Memory module

Memory modules play a critical role in AI agents. A memory module can essentially be thought of as a store of the agent's internal logs as well as interactions with a user. There are two types of memory modules:

- **Short-term memory:** A ledger of actions and thoughts that an agent goes through to attempt to answer a single question from a user: the agent's "train of thought."

- **Long-term memory:** A ledger of actions and thoughts about events that happen between the user and agent. It is a log book that contains a conversation history stretching across weeks or months.

It typically requires a composite score that combines various metrics such as semantic similarity, importance and other application-specific factor. This composite score helps in retrieving specific information more effectively.

Tools

Tools are well-defined executable workflows that agents can use to execute tasks.

Planning module

Complex problems, such as analyzing a set of financial reports to answer a layered business question, often require nuanced approaches. With an LLM–powered agent, this complexity can be dealt with by using a combination of two techniques:

- **Task and question decomposition:** Compound questions or inferred information require some form of decomposition.
- **Reflection or critic:**Techniques like ReAct, Reflexion, Chain of Thought, and Graph of thought have served as critic– or evidence-based prompting frameworks. They have been widely used to improve the reasoning capabilities and responses of LLMs.

2.4 Retrival Augmented Generation

2.4.1 Overview of a RAG Framework

The Original RAG Model:

Retrieval Augmented Generation (RAG) was proposed by Lewis et al [7] as a method to address hallucination issues caused by the lack of knowledge of LLMs. In essence, RAG manipulates and augments an LLM's knowledge base with relevant context to enhance answer quality. This allows the model to adapt to a specific domain while mitigating the need for specialized training, which is especially useful in knowledge-intensive environments.

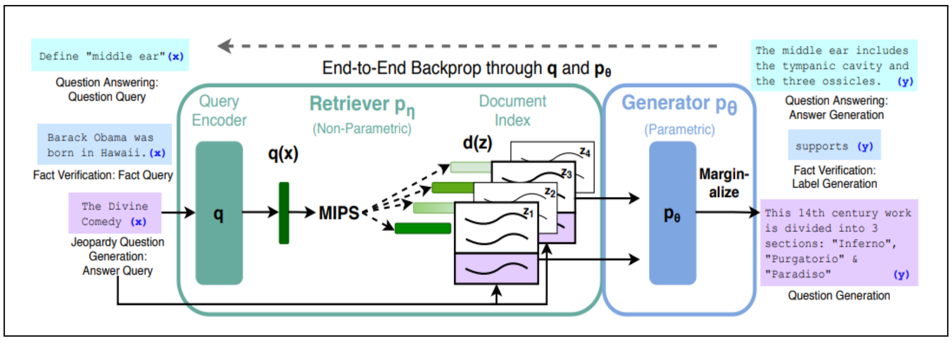

Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and pθ
Query Encoder
Retriever pη
(Non-Parametric)
Document Index
q
q(x)
MIPS
d(z)
z4
z3
z2
z1
Generator pθ
(Parametric)
pθ
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Figure 2.5: The original RAG system

As illustrated in Figure 2.5, the original system contains two main components, the retriever and the generator. The base of the system is a BERT encoder built with BART as the LLM generator. Overall, it behaves similarly to the encoder-decoder model; upon receiving an input sequence x, the encoder transforms it into a dense embedding vector, which is later fed to the decoder to generate an output sequence.

into the decoder, the retriever would search for K most similar documents to the query x. This is achieved using a vector search algorithm that maximizes the inner product between the query and document embedding vectors. The process continues by augmenting the most relevant documents z into the generator, allowing it to be conditioned on a specific context and produce responses with factual accuracy and relevance. For a less abstract explanation,This a formal description of the retriever and generator:

- The retriever $p_\eta(z \mid x)$ returns the distribution of the retrieved content given a query. Using the normalized cosine similarity measure, this distribution describes the relevance of each document z with respect to x among the top K most similar documents. The retriever is referred to as non-parametric memory as it represents the external knowledge of the RAG system.

- The generator $p_\theta(y_i \mid x, z, y_{1:i-1})$ generates a current token based on a context of the previous $i-1$ tokens $y_{1:i-1}$, the original input x, and the retrieved passage z. Since the neural network structures of the generator serve as the inherent memory of the RAG model, it is referred to as parametric memory.

A Higher-level RAG Framework:

The original RAG system was built as an end-to-end trainable model, allowing the retriever

integrated with the system to be tuned during training. Nevertheless, this process can be challenging to reproduce because of its time consumption and computational costs.As such, nowadays, RAG is instead viewed as a framework to combine with LLM systems to enhance knowledge base management.However, most RAG systems are constructed based on the Naive RAG architecture, which is illustrated in Figure 2.6.Naive RAG is a high-level framework with the most fundamental components of a RAG system: the retriever and the generator.
The retriever consists of an embedding model to encode queries and documents intovector embeddings, along with a vector database storing document embeddings and performing similarity-based document retrieval. On the other hand, the generator can be any LLM agent suitable for text generation such as GPT models. The overall process is similar: the retriever component

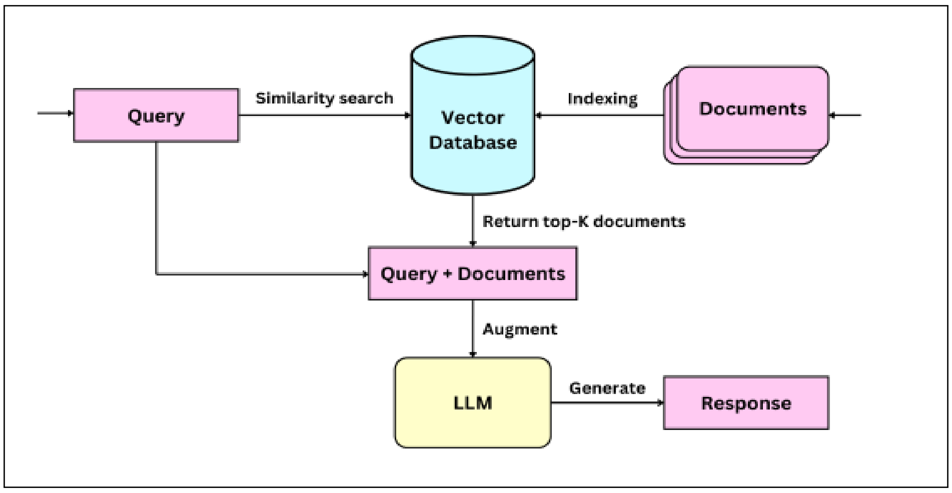


Figure 2.6: Naive RAG framework

2.4.2 An Advanced Framework: RAG Fusion

In many cases, humans are not efficient and accurate at writing prompts to LLMs. This may result in multifaceted and ambiguous queries to the retriever and the LLM generator. This issue is especially common in the recruitment context where job requirements are often composed of many different aspects. Simple RAG systems such as Naive RAG may further amplify this problem if the retriever searches for documents similar to the poor query to feed the LLM, leading to a less accurate and relevant response.

RAG Fusion is a novel technique designed to address this limitation. The key idea is to first deconstruct the original input queries into smaller but more focused sub-queries. For each of these queries (including the original one), the retriever then performs similarity-based retrieval to obtain a relevant document list. A specialized algorithm is then applied to combine and re-rank these lists based on their ranking across multiple lists. The final result is a new ranked list representing the most relevant documents to the original complex query. Figure 2.47illustrates an overview of this process.

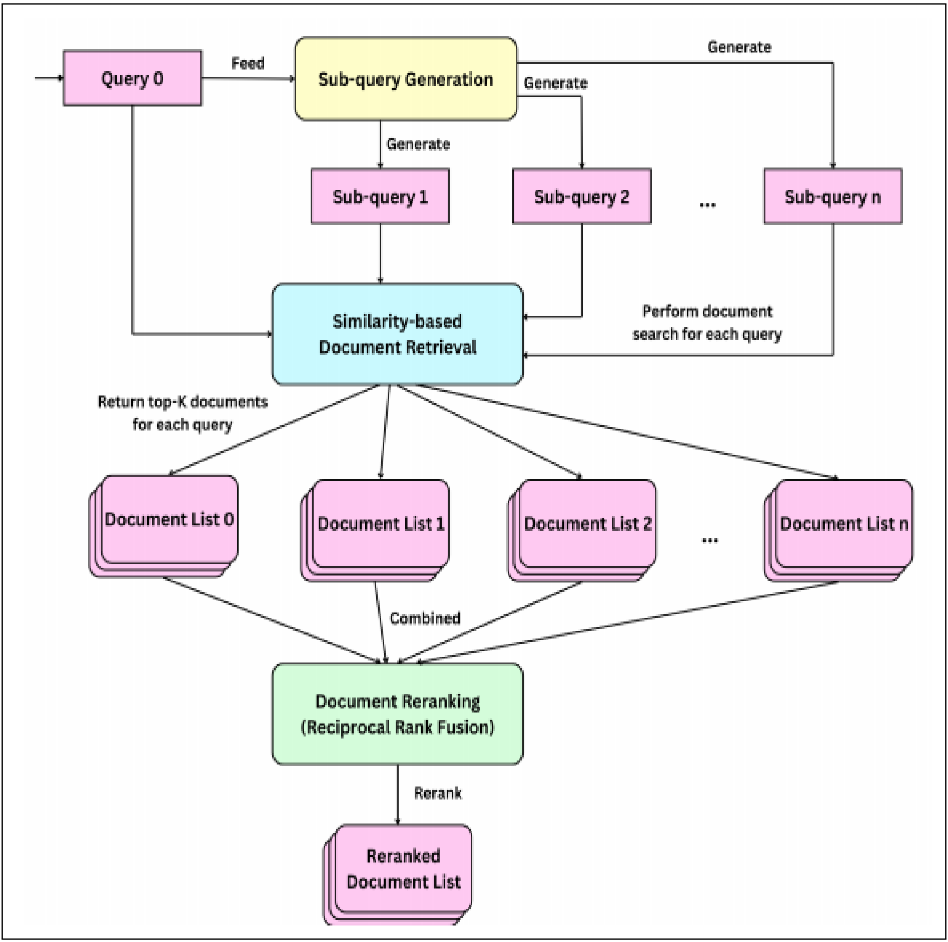


Figure 2.7: Overview of the RAG Fusion framework

The central process of the RAG Fusion framework is its re-ranking mechanism. This re-ranker is responsible for fusing multiple lists of documents into one ranked list so that every new rank reflects the original rankings of documents. To achieve this, the re-ranker utilizes an algorithm called Reciprocal Rank Fusion (RRF), an unsupervised way to effectively re-rank the results of multiple ranked lists with only a few simple initial conditions. Given a set D of documents to be ranked, a set of rankings R, and a constant k to indicate the influence of the current rank on the final score, RRF sorts each document d according to the scoring formula: Pour comparer objectivement les performances des modèles RNN, GRU et LSTM, nous avons utilisé les métriques d'erreur standard suivantes :

- **Mean Absolute Error :** Mesure la moyenne des erreurs absolues entre les valeurs prédites et réelles.
$$MAE = \frac{1}{n}\sum |y_{pred} - y_{true}|$$
- **Root Mean Squared Error :** Similaire au MAE, mais pénalise davantage les grosses erreurs en mettant les erreurs au carré avant de prendre la moyenne et la racine carrée.
$$RMSE = \sqrt{\frac{1}{n}\sum (y_{pred} - y_{true})^2}$$
- **Mean Absolute Percentage Error :** Le MAPE est une mesure d'erreur qui indique, en pourcentage, à quel point les prédictions d'un modèle s'écartent en moyenne des vraies valeurs.
$$MAPE = \frac{1}{n}\sum \left|\frac{y_{pred} - y_{true}}{y_{true}}\right| \times 100$$

The main objective of RRF is to give a higher final ranking to those near the top of more lists. It is clear to observe that the scoring formula closely reflects this goal. For each item, the algorithm computes the inverse rank scores obtained from each list, so that a lower r(d) value, which represents a higher rank, results in a higher individual score. By summing all of the scores from each list, it is possible to produce a combined score for the final ranking that reflects every individual ranking score.
Regardless of the relatively simple approach, RRF was shown to outperform various baseline models in re-ranking. Leveraging this method, RAG Fusion can effectively merge results from multiple separated sub-queries into a single document set, ensuring a comprehensive and relevant context. This framework is especially effective in scenarios where human queries are overly complicated such as the recruitment domain.

2.5 Conclusion

This chapter covers the essential concepts for our project. We start by introducing the methodologies of work, followed by an exploration of Large Language Models (LLMs). We explain their core architecture and fundamental principles, including how they process and generate natural language. We then discuss vector representation, a key component in how LLMs understand and handle text.

Next, we examine natural language generation, a critical capability of LLMs that allows them to produce coherent and contextually relevant content. Following this, we delve into Retrieval-Augmented Generation (RAG), a powerful framework that combines information retrieval with text generation to enhance the performance of LLMs. Special attention is given to RAG Fusion, an advanced variant that optimizes text generation through innovative fusion techniques.

In the following chapter, we will delve into the practical realization of this project, discussing the implementation and the methods used to bring these concepts to life.

Chapter
3

Realization

Contents

3.1 INTRODUCTION

This chapter provides an overview of the methodologies and approaches used in our project. We will then detail each step of the methodologies employed.

3.2 Project Approach

For the development of our resume parser and chatbot, we have chosen the CRISP-DM methodology due to its structured yet adaptable approach, which is crucial for managing the complexities of natural language processing and machine learning in our project. CRISP-DM's comprehensive framework provides a clear roadmap through its six phases—Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment—enabling us to iteratively refine our models and solutions.

CRISP-DM's focus on data mining is particularly advantageous for our project. The detailed Data Preparation phase ensures we effectively handle and process resume data, while the rigorous Evaluation phase validates that our models accurately match candidates with job descriptions. This alignment between phases is key to developing a precise and effective solution that meets our business objectives and enhances the user experience.

3.3 Business Understanding

The Business Understanding phase is fundamental to ensuring the success of a data science project. It consists of four key tasks, each designed to clarify objectives and lay a solid foundation for project development [6]. Below is a breakdown of these tasks and their projected outcomes:

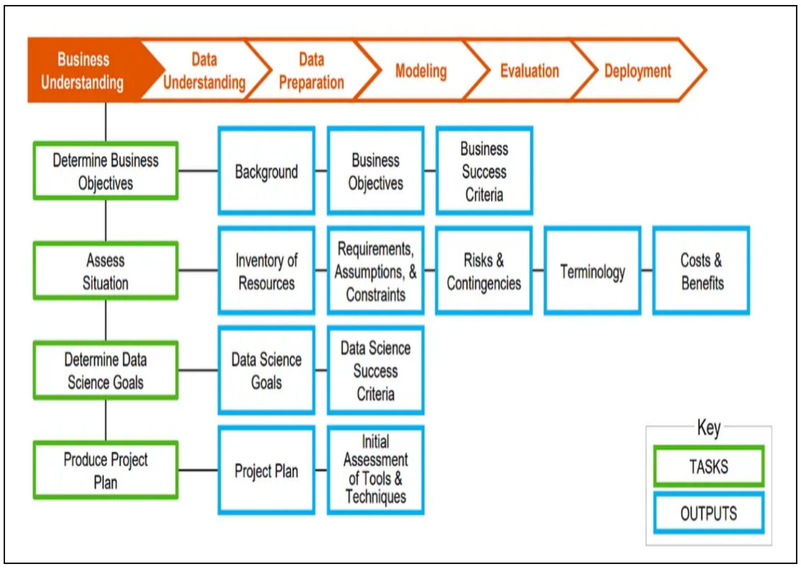


Figure 3.1: Business Understanding phase

3.3.1 Determine Business Objectives

The first step is to clearly define what the company aims to achieve from a business perspective. This involves understanding the specific challenges or inefficiencies that the business is facing. It also includes setting clear objectives that can address these issues and identifying success criteria to measure the effectiveness of the project.

Background:

- The company receives a large number of resumes every month, and the current manual screening process is both time-consuming and inconsistent.

Business Objectives:

- Automate the initial screening of job applicants to streamline the hiring process.
- Reduce the manual effort involved in reviewing resumes.

Business Success Criteria:

- Significantly reduce the time spent on manual resume screening.
- Ensure that the automated system accurately parses and extracts relevant information from resumes to effectively match job requirements.

3.3.2 Assess Situation

This task involves evaluating the current state of resources, identifying requirements, making assumptions, and acknowledging constraints. It includes taking inventory of available resources, outlining project requirements, considering risks, and defining relevant terminology to ensure consistent communication.

Inventory of Resources:

- The current process involves manually reviewing resumes.
- Available resources include a database of historical resumes and a team of HR professionals.

Requirements:

- Resume Parsing Accuracy
- Real-Time Processing
- Scalability ‘
- Security
- Reliability
- Data Availability

Assumptions:

- Sufficient computational resources are available for developing, training, and deploying the model
- The team has access to necessary software tools and libraries (e.g., spaCy, TensorFlow).
- Key stakeholders (e.g., HR department, IT department) are supportive and available for consultations and feedback. ‘
- There is a commitment from the business to implement and use the resume parsing system once developed.
- The environment (e.g., servers, network) can support the additional load from the CV parsing operations.

Constraints:

- Compliance with data protection laws such as GDPR, ensuring that candidate data is handled securely and with consent.
- Limitations of the chosen NLP and machine learning libraries in handling specific CV formats or languages.
- The project must stay within the allocated budget, covering development, deployment, and maintenance costs. ‘
- Cost-effective use of computational resources and storage solutions.
- The project must be completed within a specified timeframe, ensuring timely delivery and implementation.
- Milestones and deadlines for each phase of the project must be met to avoid delays.

Risks and Contingencies:

- Potential issues with data quality, handling different CV formats, and integration challenges are identified. Strategies are in place to mitigate these risks.

Terminology:

- resume Parsing: The process of converting an unstructured resume document into a structured format that can be easily analyzed and processed by software systems.
- NLP: A field of artificial intelligence that focuses on the interaction between computers and human language, enabling them to understand, interpret, and generate human language.
- NER: A subtask of NLP that involves identifying and classifying named entities (such as people, organizations, locations, dates, and other proper nouns) within a text. ‘
- Tokenization: The process of breaking down text into smaller units called tokens, which could be words, phrases, symbols, or other meaningful elements.
- Part-of-Speech Tagging (POS Tagging):The process of marking up a word in a text as corresponding to a particular part of speech, based on its definition and context.
- Deep Learning: A subset of machine learning involving neural networks with many layers (deep neural networks) that can learn from large amounts of data.

- API: A set of protocols and tools that allow different software applications to communicate with each other.
- REST API: A type of API that uses Representational State Transfer (REST) principles to enable communication between clients and servers.
- Structured Data: Data that is organized in a defined manner, often in tables or databases, making it easy to search, analyze, and manipulate.
- Unstructured Data: Data that does not have a predefined format or organization, such as free text in documents, emails, or social media posts.
- Batch Processing: The processing of a large volume of data at once, typically at scheduled times, rather than in real-time.
- Retrieval-Augmented Generation : A technique used in NLP to improve the performance of text generation models by incorporating relevant information retrieved from a large corpus.
- Real-time Processing: The immediate processing of data as it is received, allowing for instant analysis and response.
- Large Language Model: A type of artificial intelligence model that uses deep learning techniques, particularly transformers, to understand, generate, and manipulate human language. LLMs are trained on massive datasets and are capable of performing a wide range of language-related tasks, such as translation, summarization, question answering, and more.

Costs and Benefits:

- Costs include data storage, computational resources, and development time.
- Benefits include significant time savings and improved recruitment outcomes.

3.3.3 Determine Data Science Goals

In this step, the data science goals are transformed into concrete technical objectives. This includes establishing specific targets, such as achieving a minimum of 90% accuracy in extracting critical information like personal details, work experience, education, and skills from resumes.

Success criteria are defined by assessing the model's performance in real-world scenarios and its capacity to meet business objectives.

3.3.4 Produce Project Plan

To achieve the data science goals, a structured project plan is essential. This plan outlines the key phases

Project Plan:

- Data collection and preprocessing (2 weeks)
- Model development and testing the data extracted (5 weeks)
- Model deployment (1 weeks)

Initial Assessment of Tools and Techniques:

- Evaluate NLP libraries (e.g., spaCy, NLTK), machine learning frameworks (e.g., TensorFlow, PyTorch), and Retrieval-Augmented Generation

3.4 Data Understanding

3.4.1 Data Collection

For data collection, a multi-faceted approach was employed to gather a diverse set of resumes. The methods used include:

- Downloading from LinkedIn
- Collecting from Personal Connections
- Web Scraping

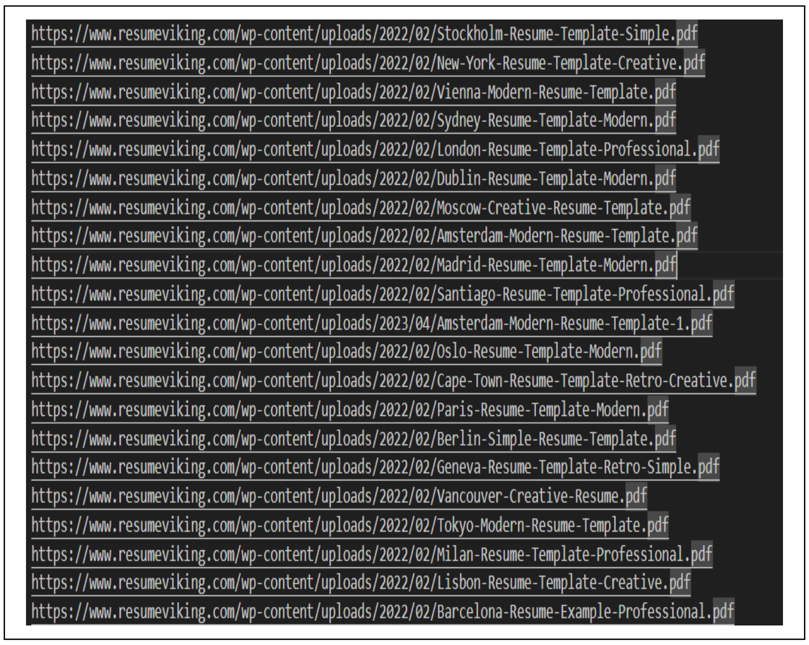
Web scraping was performed using BeautifulSoup and Selenium to extract resumes from the ResumeViking website and other similar sites.

BeautifulSoup was utilized for parsing HTML and XML documents, facilitating the extraction of structured data from web pages. It enabled the navigation of the HTML structure to locate specific elements containing resume information.

Selenium was employed for its capability to interact with dynamic web content. It automated browser interactions, such as clicking buttons and filling out forms, which were essential for accessing and downloading resumes from websites with dynamic content.

The combination of these tools provided an effective means of extracting resume data. The general steps for web scraping included:

- **Setup and Initialization:** Import the necessary libraries. Define the target URL and initialize the web driver to open the browser.
- **Data Retrieval:** Navigate to the specified URL, retrieve the page's HTML content, and parse it to identify relevant sections.
- **Data Extraction:** Extract the resume links from the parsed content. Handle pagination if necessary to gather data across multiple pages.
- **Storage and Cleanup:** Save the extracted data to a file. Close the web browser and release any resources used.



```
https://www.resumeviking.com/wp-content/uploads/2022/02/Stockholm-Resume-Template-Simple.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/New-York-Resume-Template-Creative.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Vienna-Modern-Resume-Template.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Sydney-Resume-Template-Modern.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/London-Resume-Template-Professional.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Dublin-Resume-Template-Modern.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Moscow-Creative-Resume-Template.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Amsterdam-Modern-Resume-Template.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Madrid-Resume-Template-Modern.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Santiago-Resume-Template-Professional.pdf
https://www.resumeviking.com/wp-content/uploads/2023/04/Amsterdam-Modern-Resume-Template-1.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Oslo-Resume-Template-Modern.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Cape-Town-Resume-Template-Retro-Creative.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Paris-Resume-Template-Modern.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Berlin-Simple-Resume-Template.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Geneva-Resume-Template-Retro-Simple.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Vancouver-Creative-Resume.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Tokyo-Modern-Resume-Template.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Milan-Resume-Template-Professional.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Lisbon-Resume-Template-Creative.pdf
https://www.resumeviking.com/wp-content/uploads/2022/02/Barcelona-Resume-Example-Professional.pdf
```

Figure 3.2: Example of the links extracted

This structured approach to data collection ensured the creation of a comprehensive and diverse dataset containing 300 resumes.

3.4.2 Feature Selection

After discussions with the stakeholders and a thorough analysis of the resumes I collected, we finalized the features that need to be extracted. The selected features are as follows:

- **First and Last Name:** The candidate's full name.
- **Contact Information:** Includes the candidate's email, phone number, location, LinkedIn profile, and GitHub profile.
- **Summary:** A brief overview of the candidate's profile, highlighting key strengths and experiences.
- **Languages:** Lists the languages the candidate speaks along with their proficiency level.
- **Skills:** A comprehensive list of all the technical and soft skills mentioned in the resume.
- **Education:** Details of the candidate's educational background, including the institution, duration, and degree obtained.
- **Internships:** Information on internships, including the duration, technologies used, responsibilities, and the name of the organization.
- **Work Experience:** A detailed breakdown of the candidate's professional experience, including the duration, technologies used, responsibilities, and the organization name.
- **Projects:** A summary of the projects undertaken, with details on the title, technologies used, description, and duration of each project.
- **Volunteer Work:** Information about volunteer experiences, specifying the organization name, responsibilities, and position held.
- **Certifications:** A list of certifications earned by the candidate.
- **Interests and Hobbies:** A description of the candidate's personal interests and hobbies.

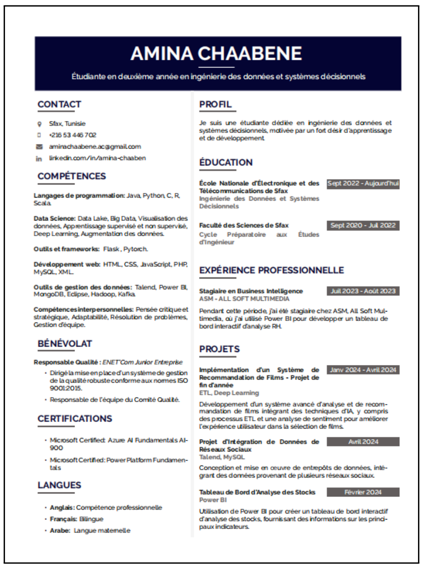
- **Image:** The candidate's image, if available.
- **Links:** URLs for the candidate's GitHub, Medium, Codeforces, and other relevant project links.

This comprehensive set of features ensures that we capture all the critical information required for effective candidate evaluation.

3.5 data Preparation

3.5.1 Text Extraction

For extracting text from resumes, we have two primary solutions. The results are demonstrated using the resume shown in Figure 3.3.



AMINA CHAABENE

Étudiante en deuxième année en ingénierie des données et systèmes décisionnels

CONTACT

Sfax, Tunisie

+216 53 446 702

aminachaabene.ac@gmail.com

in linkedin.com/in/amina-chaaben

COMPÉTENCES

Langages de programmation: Java, Python, C, R, Scala.

Data Science: Data Lake, Big Data, Visualisation des données, Apprentissage supervisé et non supervisé, Deep Learning, Augmentation des données.

Outils et frameworks: Flask , Pytorch.

Développement web: HTML, CSS, JavaScript, PHP, MySQL, XML.

Outils de gestion des données: Talend, Power BI, MongoDB, Eclipse, Hadoop, Kafka.

Compétences interpersonnelles: Pensée critique et stratégique, Adaptabilité, Résolution de problèmes, Gestion d'équipe.

BÉNÉVOLAT

Responsable Qualité : *ENET'Com Junior Entreprise*

- Dirigé la mise en place d'un système de gestion de la qualité robuste conforme aux normes ISO 9001:2015.
- Responsable de l'équipe du Comité Qualité.

CERTIFICATIONS

- Microsoft Certified: Azure AI Fundamentals AI-900
- Microsoft Certified: Power Platform Fundamentals

LANGUES

- **Anglais :** Compétence professionnelle
- **Français:** Bilingue
- **Arabe:** Langue maternelle

PROFIL

Je suis une étudiante dédiée en ingénierie des données et systèmes décisionnels, motivée par un fort désir d'apprentissage et de développement.

ÉDUCATION

École Nationale d'Électronique et des Télécommunications de Sfax Sept 2022 - Aujourd'hui
Ingénierie des Données et Systèmes Décisionnels

Faculté des Sciences de Sfax Sept 2020 - Juil 2022
Cycle Préparatoire aux Études d'Ingénieur

EXPÉRIENCE PROFESSIONNELLE

Stagiaire en Business Intelligence Juil 2023 - Août 2023
ASM - ALL SOFT MULTIMEDIA

Pendant cette période, j'ai été stagiaire chez ASM, All Soft Multimedia, où j'ai utilisé Power BI pour développer un tableau de bord interactif d'analyse RH.

PROJETS

Implémentation d'un Système de Recommandation de Films - Projet de fin d'année Janv 2024 - Avril 2024
ETL, Deep Learning

Développement d'un système avancé d'analyse et de recommandation de films intégrant des techniques d'IA, y compris des processus ETL et une analyse de sentiment pour améliorer l'expérience utilisateur dans la sélection de films.

Projet d'Intégration de Données de Réseaux Sociaux Avril 2024
Talend, MySQL

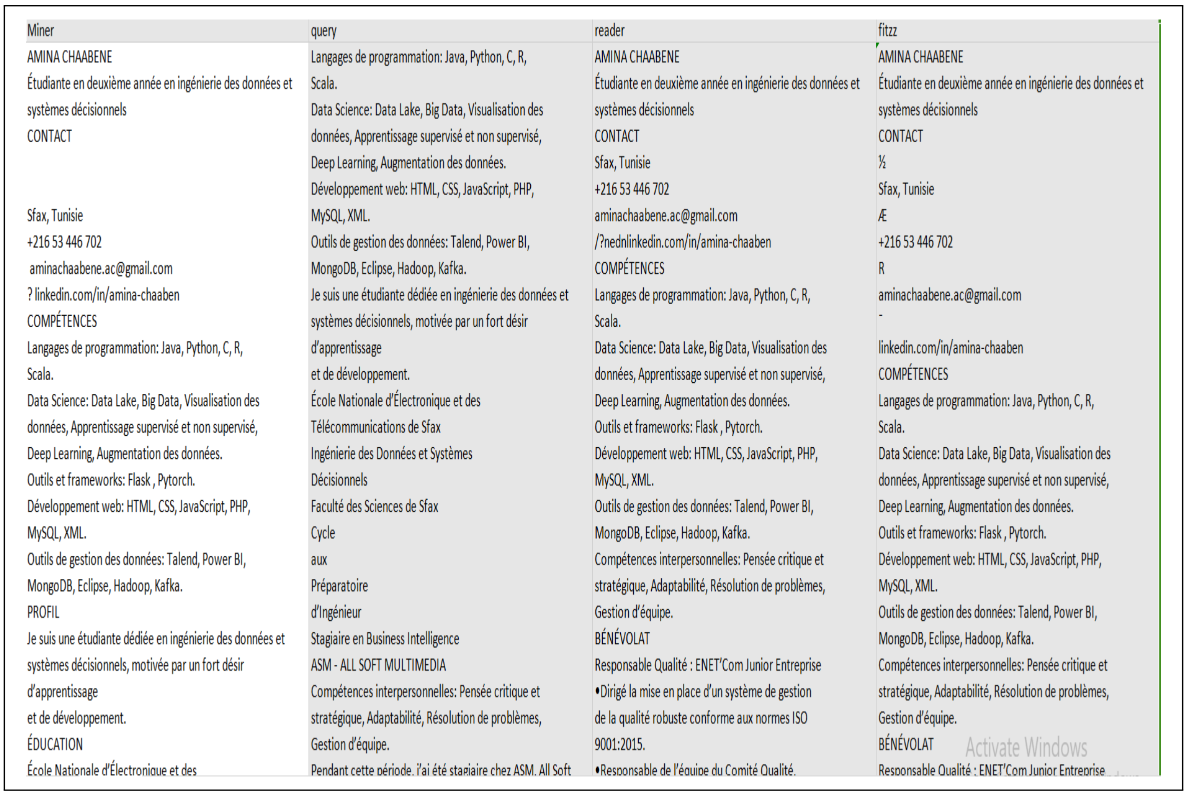
Conception et mise en oeuvre de entrepôts de données, intégrant des données provenant de plusieurs réseaux sociaux.

Tableau de Bord d'Analyse des Stocks Février 2024
Power BI

Utilisation de Power BI pour créer un tableau de bord interactif d'analyse des stocks, fournissant des informations sur les principaux indicateurs.

Figure 3.3: Resume example

Python Libraries: Our initial approach involved using a combination of Python libraries, including PDFMiner, Fitzz, PDFQuery, and PDFReader. These tools were selected for their text parsing and extraction capabilities from PDF documents.



Miner	query	reader	fitzz
AMINA CHAABENE	Langages de programmation: Java, Python, C, R,	AMINA CHAABENE	AMINA CHAABENE
Étudiante en deuxième année en ingénierie des données et	Scala.	Étudiante en deuxième année en ingénierie des données et	Étudiante en deuxième année en ingénierie des données et
systèmes décisionnels	Data Science: Data Lake, Big Data, Visualisation des	systèmes décisionnels	systèmes décisionnels
CONTACT	données, Apprentissage supervisé et non supervisé,	CONTACT	CONTACT
	Deep Learning, Augmentation des données.	Sfax, Tunisie	½
	Développement web: HTML, CSS, JavaScript, PHP,	+216 53 446 702	Sfax, Tunisie
Sfax, Tunisie	MySQL, XML.	aminachaabene.ac@gmail.com	Æ
+216 53 446 702	Outils de gestion des données: Talend, Power BI,	/?nednlinkedin.com/in/amina-chaaben	+216 53 446 702
aminachaabene.ac@gmail.com	MongoDB, Eclipse, Hadoop, Kafka.	COMPÉTENCES	R
? linkedin.com/in/amina-chaaben	Je suis une étudiante dédiée en ingénierie des données et	Langages de programmation: Java, Python, C, R,	aminachaabene.ac@gmail.com
COMPÉTENCES	systèmes décisionnels, motivée par un fort désir	Scala.	-
Langages de programmation: Java, Python, C, R,	d'apprentissage	Data Science: Data Lake, Big Data, Visualisation des	linkedin.com/in/amina-chaaben
Scala.	et de développement.	données, Apprentissage supervisé et non supervisé,	COMPÉTENCES
Data Science: Data Lake, Big Data, Visualisation des	École Nationale d'Électronique et des	Deep Learning, Augmentation des données.	Langages de programmation: Java, Python, C, R,
données, Apprentissage supervisé et non supervisé,	Télécommunications de Sfax	Outils et frameworks: Flask , Pytorch.	Scala.
Deep Learning, Augmentation des données.	Ingénierie des Données et Systèmes	Développement web: HTML, CSS, JavaScript, PHP,	Data Science: Data Lake, Big Data, Visualisation des
Outils et frameworks: Flask , Pytorch.	Décisionnels	MySQL, XML.	données, Apprentissage supervisé et non supervisé,
Développement web: HTML, CSS, JavaScript, PHP,	Faculté des Sciences de Sfax	Outils de gestion des données: Talend, Power BI,	Deep Learning, Augmentation des données.
MySQL, XML.	Cycle	MongoDB, Eclipse, Hadoop, Kafka.	Outils et frameworks: Flask , Pytorch.
Outils de gestion des données: Talend, Power BI,	aux	Compétences interpersonnelles: Pensée critique et	Développement web: HTML, CSS, JavaScript, PHP,
MongoDB, Eclipse, Hadoop, Kafka.	Préparatoire	stratégique, Adaptabilité, Résolution de problèmes,	MySQL, XML.
PROFIL	d'Ingénieur	Gestion d'équipe.	Outils de gestion des données: Talend, Power BI,
Je suis une étudiante dédiée en ingénierie des données et	Stagiaire en Business Intelligence	BÉNÉVOLAT	MongoDB, Eclipse, Hadoop, Kafka.
systèmes décisionnels, motivée par un fort désir	ASM - ALL SOFT MULTIMEDIA	Responsable Qualité : ENET'Com Junior Entreprise	Compétences interpersonnelles: Pensée critique et
d'apprentissage	Compétences interpersonnelles: Pensée critique et	•Dirigé la mise en place d'un système de gestion	stratégique, Adaptabilité, Résolution de problèmes,
et de développement.	stratégique, Adaptabilité, Résolution de problèmes,	de la qualité robuste conforme aux normes ISO	Gestion d'équipe.
ÉDUCATION	Gestion d'équipe.	9001:2015.	BÉNÉVOLAT
École Nationale d'Électronique et des	Pendant cette période, j'ai été stagiaire chez ASM. All Soft	•Responsable de l'équipe du Comité Qualité.	Responsable Qualité : ENET'Com Junior Entreprise

Activate Windows

Figure 3.4: the Result of Python library

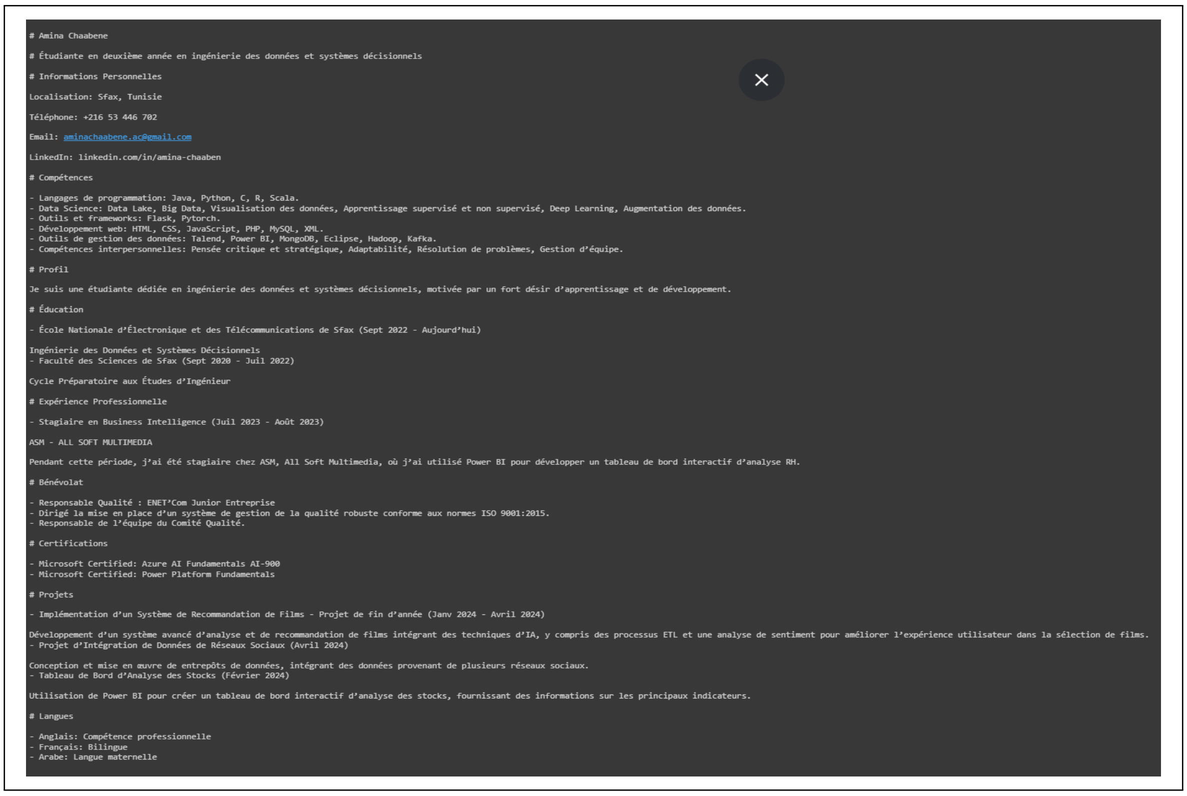
LlamaParse:

LlamaParse, a GenAI-native document parsing platform specifically designed for integration with large language models (LLMs). LlamaParse delivered significantly more consistent and reliable text extraction results compared to the traditional Python libraries.

Key advantages of LlamaParse include:

- **Versatile File Format Support:** LlamaParse effectively handled a wide range of file formats beyond PDFs, making it a versatile tool for diverse document types.
- **Foreign Language Support:** The platform offered robust support for multiple languages, ensuring accurate text extraction across various linguistic contexts.
- **Natural Language Instructions:** LlamaParse allowed us to use natural language instructions to customize the output format, providing greater control and precision over the extracted data.

These features made LlamaParse the superior choice for our project, enabling us to overcome the limitations encountered with Python libraries.



```
# Amina Chaabene

# Étudiante en deuxième année en ingénierie des données et systèmes décisionnels

# Informations Personnelles

Localisation: Sfax, Tunisie

Téléphone: +216 53 446 702

Email: aminachaabene.ac@gmail.com

LinkedIn: linkedin.com/in/amina-chaaben

# Compétences

- Langages de programmation: Java, Python, C, R, Scala.
- Data Science: Data Lake, Big Data, Visualisation des données, Apprentissage supervisé et non supervisé, Deep Learning, Augmentation des données.
- Outils et frameworks: Flask, Pytorch.
- Développement web: HTML, CSS, JavaScript, PHP, MySQL, XML.
- Outils de gestion des données: Talend, Power BI, MongoDB, Eclipse, Hadoop, Kafka.
- Compétences interpersonnelles: Pensée critique et stratégique, Adaptabilité, Résolution de problèmes, Gestion d'équipe.

# Profil

Je suis une étudiante dédiée en ingénierie des données et systèmes décisionnels, motivée par un fort désir d'apprentissage et de développement.

# Éducation

- École Nationale d'Électronique et des Télécommunications de Sfax (Sept 2022 - Aujourd'hui)

Ingénierie des Données et Systèmes Décisionnels
- Faculté des Sciences de Sfax (Sept 2020 - Juil 2022)

Cycle Préparatoire aux Études d'Ingénieur

# Expérience Professionnelle

- Stagiaire en Business Intelligence (Juil 2023 - Août 2023)

ASM - ALL SOFT MULTIMEDIA

Pendant cette période, j'ai été stagiaire chez ASM, All Soft Multimedia, où j'ai utilisé Power BI pour développer un tableau de bord interactif d'analyse RH.

# Bénévolat

- Responsable Qualité : ENET'Com Junior Entreprise
- Dirigé la mise en place d'un système de gestion de la qualité robuste conforme aux normes ISO 9001:2015.
- Responsable de l'équipe du Comité Qualité.

# Certifications

- Microsoft Certified: Azure AI Fundamentals AI-900
- Microsoft Certified: Power Platform Fundamentals

# Projets

- Implémentation d'un Système de Recommandation de Films - Projet de fin d'année (Janv 2024 - Avril 2024)

Développement d'un système avancé d'analyse et de recommandation de films intégrant des techniques d'IA, y compris des processus ETL et une analyse de sentiment pour améliorer l'expérience utilisateur dans la sélection de films.
- Projet d'Intégration de Données de Réseaux Sociaux (Avril 2024)

Conception et mise en œuvre de entrepôts de données, intégrant des données provenant de plusieurs réseaux sociaux.
- Tableau de Bord d'Analyse des Stocks (Février 2024)

Utilisation de Power BI pour créer un tableau de bord interactif d'analyse des stocks, fournissant des informations sur les principaux indicateurs.

# Langues

- Anglais: Compétence professionnelle
- Français: Bilingue
- Arabe: Langue maternelle
```

Figure 3.5: Result of Text Extraction Using LlamaParse

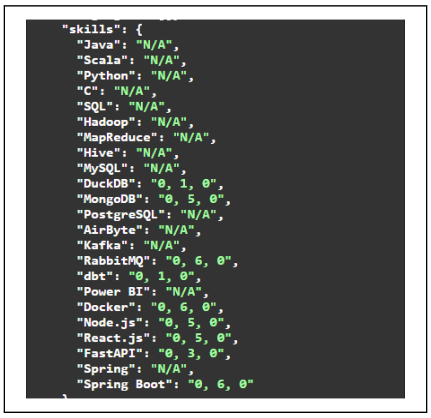
3.5.2 Period Calculation

The primary goal of this section is to extract the duration of internships, education, or work experiences from the text and associate this information with the skills section of a resume. For each skill listed, we aim to add the corresponding relevant experience period, providing a clear link between the candidate's skills and their practical application over time.

Libraries Used:

- **dateparser:** This library is employed to parse and extract dates from text, even when the formats may vary. It handles natural language input to detect and interpret date ranges accurately.
- **python-Levenshtein:** Used for calculating string similarity, this library helps in matching skills listed in a resume to relevant keywords or phrases with high precision. By comparing the textual similarity between the extracted skills and predefined skill sets, it ensures accurate and reliable skill identification.

By combining these libraries, We can effectively bridge the gap between skills and the timeline of experience, enhancing the value of resume parsing and improving skill relevance in the final output.



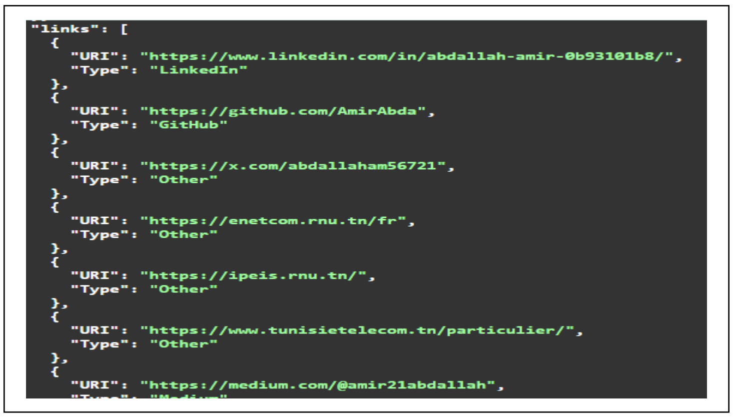
```
  "skills": {
    "Java": "N/A",
    "Scala": "N/A",
    "Python": "N/A",
    "C": "N/A",
    "SQL": "N/A",
    "Hadoop": "N/A",
    "MapReduce": "N/A",
    "Hive": "N/A",
    "MySQL": "N/A",
    "DuckDB": "0, 1, 0",
    "MongoDB": "0, 5, 0",
    "PostgreSQL": "N/A",
    "AirByte": "N/A",
    "Kafka": "N/A",
    "RabbitMQ": "0, 6, 0",
    "dbt": "0, 1, 0",
    "Power BI": "N/A",
    "Docker": "0, 6, 0",
    "Node.js": "0, 5, 0",
    "React.js": "0, 5, 0",
    "FastAPI": "0, 3, 0",
    "Spring": "N/A",
    "Spring Boot": "0, 6, 0"
```

Figure 3.6: Example of the period calculated

Figure 3.7 shows an example of the result, where we obtain a tuple representing the number of years, months, and days.

3.5.3 Link Extraction

The primary purpose of this section is to extract links and their associated text from PDF resumes. It uses the ‘fitz‘ (PyMuPDF) library to extract both the links and their corresponding text, while the ‘urllib.parse‘ module is employed to parse the URLs and extract the domains. The goal is to specifically identify and extract links to platforms such as GitHub, LinkedIn, or personal projects that can provide HR with insights into the candidate’s work and online presence.



```
"links": [
  {
    "URI": "https://www.linkedin.com/in/abdallah-amir-0b93101b8/",
    "Type": "LinkedIn"
  },
  {
    "URI": "https://github.com/AmirAbda",
    "Type": "GitHub"
  },
  {
    "URI": "https://x.com/abdallaham56721",
    "Type": "Other"
  },
  {
    "URI": "https://enetcom.rnu.tn/fr",
    "Type": "Other"
  },
  {
    "URI": "https://ipeis.rnu.tn/",
    "Type": "Other"
  },
  {
    "URI": "https://www.tunisietelecom.tn/particulier/",
    "Type": "Other"
  },
  {
    "URI": "https://medium.com/@amir21abdallah",
```

Figure 3.7: Example of links extracted

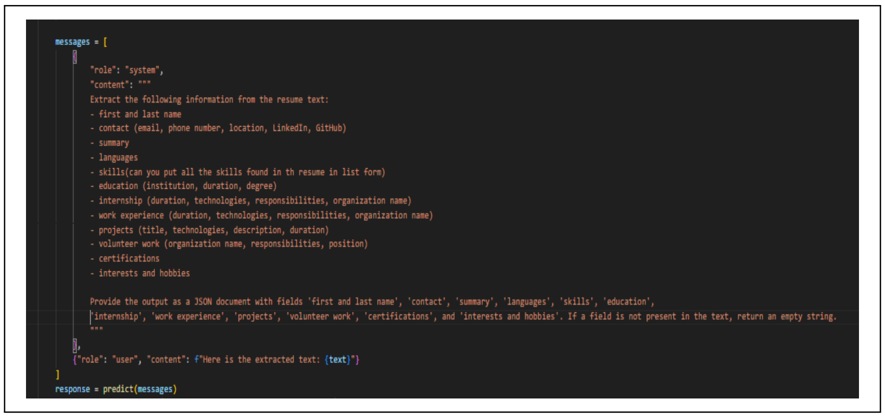
3.6 Modeling

3.6.1 Features extraction

In this section, we present the development of an LLM-based agent designed to automate the extraction of key information from resumes. This agent aims to streamline the resume parsing process while enhancing accuracy. It specifically targets essential details, including the candidate's name, contact information, skills, education, work experience, and more.

The process begins by extracting text from PDF files. If the resume is in a language other than English, the content is automatically translated into English using the Google Translator API. This step ensures consistency and facilitates subsequent processing by standardizing the text.

To achieve accurate extraction results, we crafted a detailed message that guides the LLM in identifying and structuring the required information. The messages variable consists of a list of dictionaries representing the conversation between the system and the user. The system message outlines the extraction task, specifying the types of information to be retrieved from the resume text, such as the candidate's name, contact details, skills, and education. The user message contains the actual resume text for analysis. This approach ensures that the LLM receives clear instructions and relevant data for effective processing.



```
messages = [
    {
        "role": "system",
        "content": """
        Extract the following information from the resume text:
        - first and last name
        - contact (email, phone number, location, LinkedIn, GitHub)
        - summary
        - languages
        - skills(can you put all the skills found in th resume in list form)
        - education (institution, duration, degree)
        - internship (duration, technologies, responsibilities, organization name)
        - work experience (duration, technologies, responsibilities, organization name)
        - projects (title, technologies, description, duration)
        - volunteer work (organization name, responsibilities, position)
        - certifications
        - interests and hobbies

        Provide the output as a JSON document with fields 'first and last name', 'contact', 'summary', 'languages', 'skills', 'education',
        'internship', 'work experience', 'projects', 'volunteer work', 'certifications', and 'interests and hobbies'. If a field is not present in the text, return an empty string.
        """
    },
    {"role": "user", "content": f"Here is the extracted text: {text}"}
]
response = predict(messages)
```

Figure 3.8: LLM Instructions

For managing extraction tasks, we utilized the Groq API, which is optimized for handling large-scale language models efficiently. The chosen model for this task is LLaMA 3-8B-8192, a lightweight model recognized for its capability to process extensive contexts and perform complex extraction tasks with high precision. Its lightweight nature facilitates efficient deployment and faster processing, making it particularly effective for large-scale resume analysis. The LLaMA 3-8B-8192 model analyzes the text and identifies structured information such as personal details, professional experience, education history, skills, and relevant projects.

3.6.2 Person Detection

MobileNet is a class of convolutional neural networks (CNNs) optimized for mobile and resource-constrained environments. Unlike traditional deep learning models that require substantial computing resources, MobileNet achieves efficient performance through techniques like depthwise separable convolutions, which reduce the number of parameters and computations required.

The Single Shot MultiBox Detector (SSD) is an object detection framework that allows for real-time object detection. It is called "single-shot" because it predicts bounding boxes and class scores for multiple objects in an image in a single forward pass of the network, making it extremely fast. SSD divides an input image into a grid and predicts both bounding boxes and probabilities for each object at different scales.

For person detection, we used the MobileNetSSD_deploy.caffemodel, a pre-trained model specifically designed for object detection tasks within the MobileNet SSD framework. We chose this model because it combines the lightweight MobileNet architecture with the SSD framework, offering an excellent balance between speed and accuracy. This makes it ideal for our use case, where efficiency is crucial. The model processes all images within the resume and returns only the images where a person is detected, ensuring accurate person detection with minimal computational overhead.

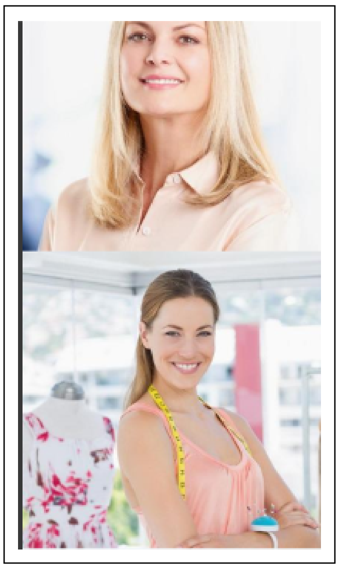


Figure 3.9: Example of person image returned.

3.6.3 Chatbot

The system employs RAG Fusion, an advanced RAG framework that combines generative agents and similarity-based retrieval methods to enhance answer quality. Moreover, RAG Fusion is notable in effectively addressing complex and ambiguous human queries. This can be particularly useful in recruitment-related scenarios, where job descriptions often contain complex and multifaceted requirements.

Figure 3.10 illustrates the high-level view of the RAG pipeline. The process begins by processing resumes into a vector storage. Upon receiving a job description query, the LLM agent is prompted to generate corresponding sub-queries. The vector storage performs a retrieval process that returns the top-K most similar documents for each sub-query. These documents are then combined and re-ranked into a new document list, serving as external context to be augmented into the LLM. This enables it to generate accurate, relevant, and informative responses that assist in matching resumes with complex job description queries.

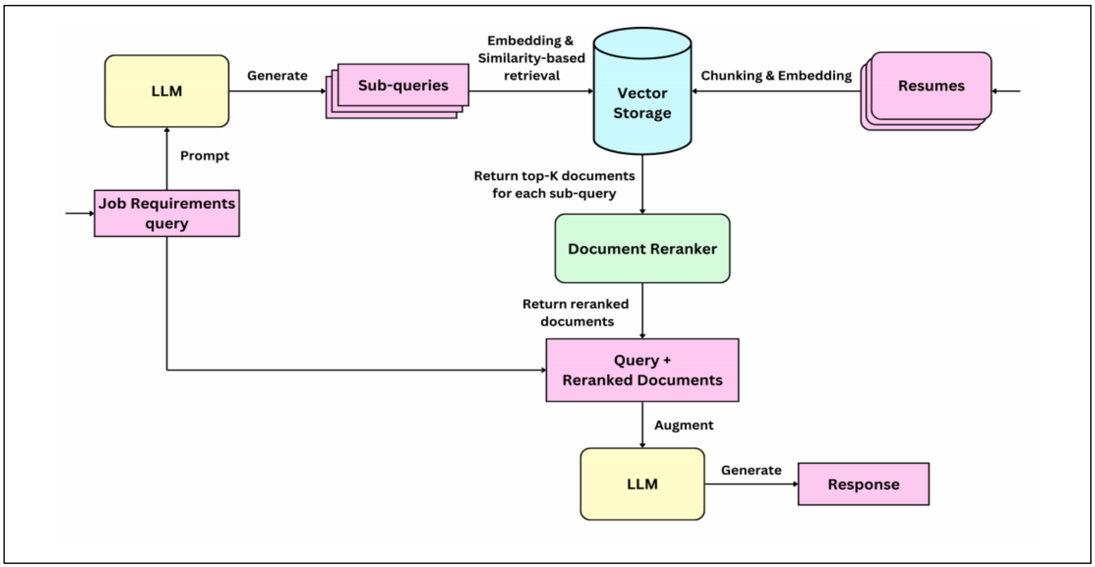


Figure 3.10: High-level view of system flow.

3.6.3.1 Preprocessing

Sub-Queries Generation

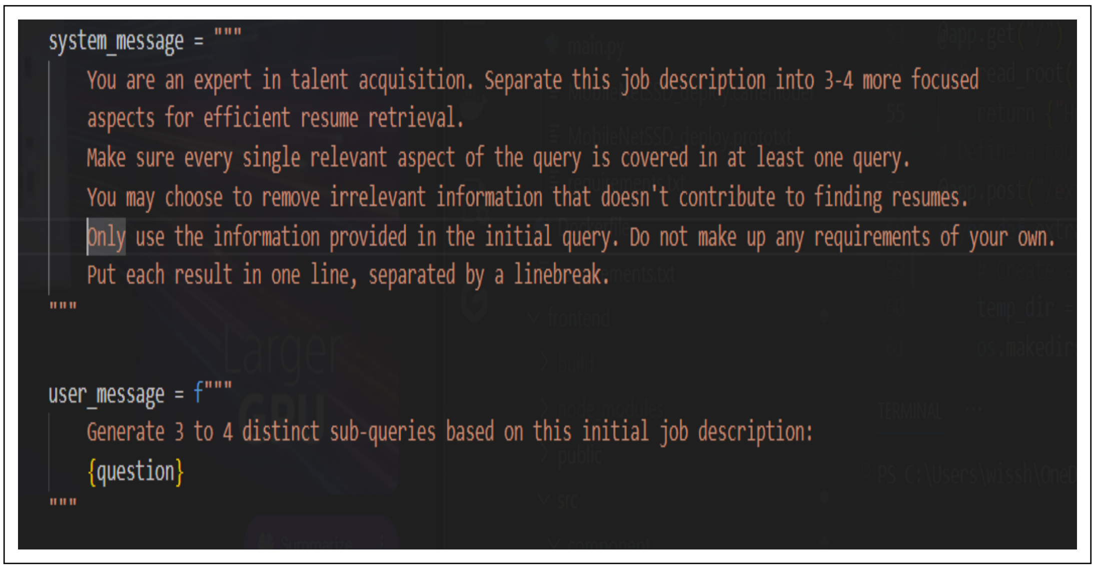
The sub-queries generation process starts by loading the user input into the LLM (Large Language Model) agent. This allows the complex job query to be analyzed and broken down into multiple sub-queries, each addressing different aspects of the job requirements. For example, a job requirement such as "Data Scientist with R/Python skills, teamwork ability, and at least 3 years of experience" can be decomposed into sub-queries like:

- "Data Scientist with R/Python skills"
- "Data Scientist with teamwork ability"
- "Data Scientist with at least 3 years of experience"

The primary goal of this framework is to extract relevant information from job postings to generate 3 to 4 sub-queries, which typically correspond to the sections found in job descriptions.

This process is facilitated by crafting a prompt that provides specific instructions and context to the LLM agent. The prompt, detailed in Listing 3.11, includes two components:

- **System Prompt:** A fixed message that offers consistent instructions and context to the LLM agent about the task at hand.
- **User Prompt:** A dynamic query submitted by the user, tailored to the particular job description or requirement.

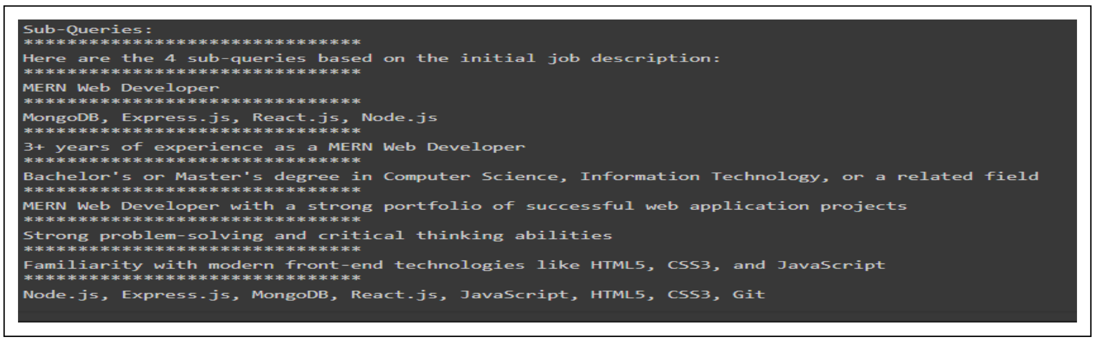


```
system_message = """
    You are an expert in talent acquisition. Separate this job description into 3-4 more focused
    aspects for efficient resume retrieval.
    Make sure every single relevant aspect of the query is covered in at least one query.
    You may choose to remove irrelevant information that doesn't contribute to finding resumes.
    Only use the information provided in the initial query. Do not make up any requirements of your own.
    Put each result in one line, separated by a linebreak.
"""

user_message = f"""
    Generate 3 to 4 distinct sub-queries based on this initial job description:
    {question}
"""
```

Figure 3.11: Prompt template for sub-queries generatio.

The overall prompt instructs the LLM agent to analyze job requirements as an expert recruiter would ("You are an expert in talent acquisition..."). It emphasizes using only the information provided in the initial query to minimize inaccuracies or "hallucinations" from the model. Additionally, the prompt can be customized to adjust parameters, such as the number of sub-queries generated, making it adaptable to specific system needs.

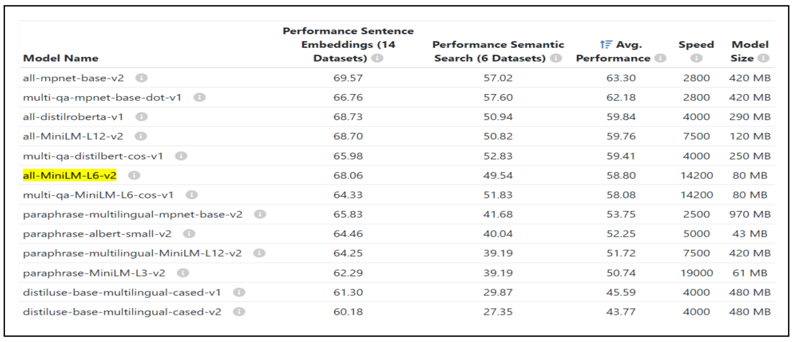


```
Sub-Queries:
*******************************
Here are the 4 sub-queries based on the initial job description:
*******************************
MERN Web Developer
*******************************
MongoDB, Express.js, React.js, Node.js
*******************************
3+ years of experience as a MERN Web Developer
*******************************
Bachelor's or Master's degree in Computer Science, Information Technology, or a related field
*******************************
MERN Web Developer with a strong portfolio of successful web application projects
*******************************
Strong problem-solving and critical thinking abilities
*******************************
Familiarity with modern front-end technologies like HTML5, CSS3, and JavaScript
*******************************
Node.js, Express.js, MongoDB, React.js, JavaScript, HTML5, CSS3, Git
```

Figure 3.12: Example of subquestions.

Embedding Model

For the text representation task, the system utilizes all-MiniLM-L6-v2, a popular lightweight sentence transformer for semantic search tasks. Reimers and Gurevych provided the Table 4.1 to compare the performance of all-MiniLM-L6-v2 among some other popular sentence transformer models. As defined by the authors, higher performance metrics reflect better performance of the model in a task, while higher speed metric denotes a higher number of sentences embedded per second. It is possible to observe that all-MiniLM-L6-v2 achieves comparable performance with other strong models in sentence embeddings and semantic search tasks while offering significantly higher speed, even with a substantially smaller model size.



Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2	60.18	27.35	43.77	4000	480 MB

Figure 3.13: Models performance comparison.

The all-MiniLM-L6-v2 model has a token limit of 256 words, meaning that an input sequence of 256 words or higher will be truncated before being embedded. The output embedding is a 384-dimensional dense vector representation of the text.

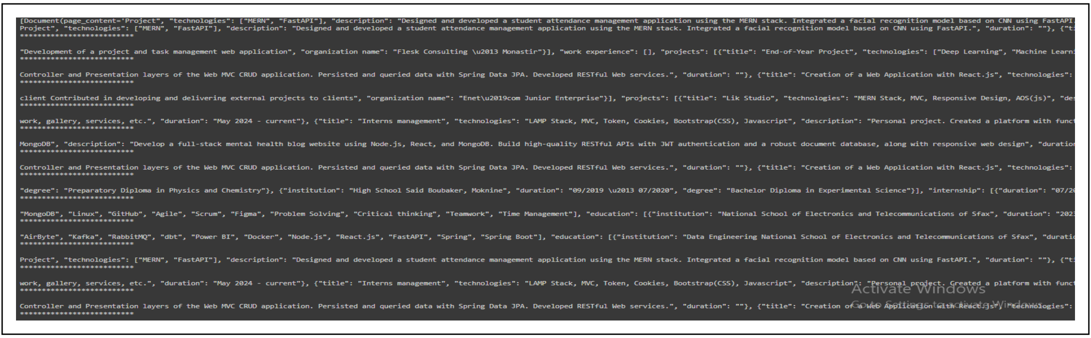
Document Chunking

Document chunking involves breaking down information-heavy documents into smaller, manageable segments. This approach enables more focused retrieval of relevant information and reduces noise by preventing individual chunks from being overloaded. In the case of resumes, which are often long and detailed, chunking can greatly enhance retrieval efficiency.

Advanced techniques like content-based chunking divide resumes based on section content, but these methods can be complex and computationally intensive. Therefore, recursive chunking is preferred for its simplicity and effectiveness. This strategy aims to keep chunks of similar size while preserving their structure and coherence, avoiding abrupt breaks in sentences or paragraphs.

Recursive chunking works by hierarchically and iteratively segmenting the text using predefined separators like periods or line breaks. If the chunks do not meet the desired size or structure, the process is repeated with different separators until the chunks achieve the appropriate size and structure.

For this system, the chunk size is set to approximately 1024 characters, balancing the needs of the embedding model with practical text length considerations. This chunk size is about half a page of text. To maintain context across chunks, an overlapping size of 50% (around 500 characters) is used. This overlap ensures that chunks retain information from adjacent sections, increasing the likelihood of capturing relevant content during retrieval.



```
[Document(page_content='Project", "technologies": ["MERN", "FastAPI"], "description": "Designed and developed a student attendance management application using the MERN stack. Integrated a facial recognition model based on CNN using FastAPI.
Project", "technologies": ["MERN", "FastAPI"], "description": "Designed and developed a student attendance management application using the MERN stack. Integrated a facial recognition model based on CNN using FastAPI.", "duration": ""}, {"ti
**************************

"Development of a project and task management web application", "organization name": "Flesk Consulting \u2013 Monastir"}], "work experience": [], "projects": [{"title": "End-of-Year Project", "technologies": ["Deep Learning", "Machine Learni
**************************

Controller and Presentation layers of the Web MVC CRUD application. Persisted and queried data with Spring Data JPA. Developed RESTful Web services.", "duration": ""}, {"title": "Creation of a Web Application with React.js", "technologies":
**************************

client Contributed in developing and delivering external projects to clients", "organization name": "Enet\u2019com Junior Enterprise"}], "projects": [{"title": "Lik Studio", "technologies": "MERN Stack, MVC, Responsive Design, AOS(js)", "des
**************************

work, gallery, services, etc.", "duration": "May 2024 - current"}, {"title": "Interns management", "technologies": "LAMP Stack, MVC, Token, Cookies, Bootstrap(CSS), Javascript", "description": "Personal project. Created a platform with funct
**************************

MongoDB", "description": "Develop a full-stack mental health blog website using Node.js, React, and MongoDB. Build high-quality RESTful APIs with JWT authentication and a robust document database, along with responsive web design", "duratio
**************************

Controller and Presentation layers of the Web MVC CRUD application. Persisted and queried data with Spring Data JPA. Developed RESTful Web services.", "duration": ""}, {"title": "Creation of a Web Application with React.js", "technologies":
**************************

"degree": "Preparatory Diploma in Physics and Chemistry"}, {"institution": "High School Said Boubaker, Moknine", "duration": "09/2019 \u2013 07/2020", "degree": "Bachelor Diploma in Experimental Science"}], "internship": [{"duration": "07/20
**************************

"MongoDB", "Linux", "GitHub", "Agile", "Scrum", "Figma", "Problem Solving", "Critical thinking", "Teamwork", "Time Management"], "education": [{"institution": "National School of Electronics and Telecommunications of Sfax", "duration": "2023
**************************

"AirByte", "Kafka", "RabbitMQ", "dbt", "Power BI", "Docker", "Node.js", "React.js", "FastAPI", "Spring", "Spring Boot"], "education": [{"institution": "Data Engineering National School of Electronics and Telecommunications of Sfax", "duratio
**************************

Project", "technologies": ["MERN", "FastAPI"], "description": "Designed and developed a student attendance management application using the MERN stack. Integrated a facial recognition model based on CNN using FastAPI.", "duration": ""}, {"ti
**************************

work, gallery, services, etc.", "duration": "May 2024 - current"}, {"title": "Interns management", "technologies": "LAMP Stack, MVC, Token, Cookies, Bootstrap(CSS), Javascript", "description": "Personal project. Created a platform with funct
**************************

Controller and Presentation layers of the Web MVC CRUD application. Persisted and queried data with Spring Data JPA. Developed RESTful Web services.", "duration": ""}, {"title": "Creation of a Web Application with React.js", "technologies":
**************************
```

Figure 3.14: Example of the text chunks.

3.6.3.2 Retrieval

Vector Store

We use a vector store in this case, which is a lightweight storage system designed for efficient management and retrieval of dense vectors. Facebook AI Similarity Search (FAISS), a library developed by Meta, is employed for this purpose. FAISS provides fundamental vector store functionalities, including local vector storage and highly efficient in-memory similarity searching. It achieves this through various specialized index structures that enable rapid retrieval of similar vectors.

Data Ingestion

The resume embeddings and original text chunks from the previous stage are indexed into a vector space using FAISS. A flat index is used for indexing, prioritizing search accuracy and precision over speed. Ensuring accurate and fair retrieval of resumes is more crucial than achieving the fastest search speed. Despite this, retrieval speed remains efficient, as the number of vectors to be processed is manageable (not in the billions). The indexed data is saved to the local disk, serving as the knowledge base for the LLM agent.

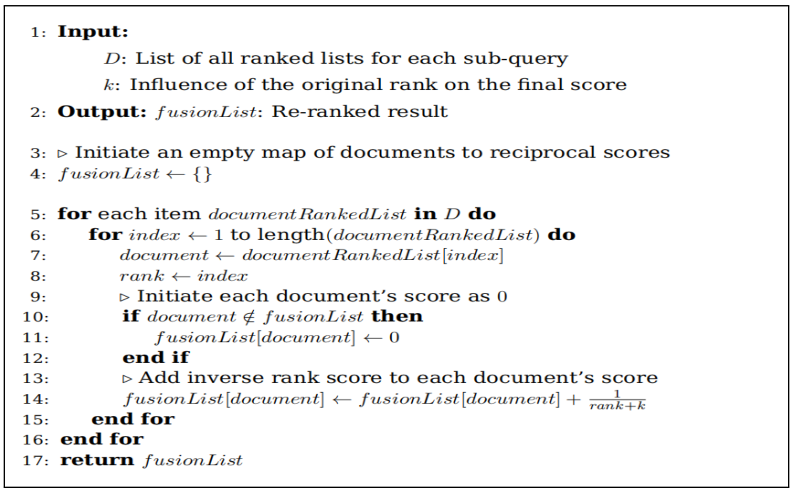
Similarity-based Retrieval

The similarity measure used is the cosine distance, which is the complement of the cosine similarity measure. This choice is due to the library's support for cosine distance as a metric. Cosine distance measures the angular difference between vectors, focusing on their semantic similarity rather than their magnitudes.

FAISS searches the vector space using cosine distance to compare query embeddings with document embeddings. It retrieves the most similar resume chunks for each sub-query, ranking them accordingly. Finally, the results from all sub-queries are combined, maintaining the original ranking of the documents.

Reciprocal Rank Fusion (RRF)

Reciprocal Rank Fusion (RRF) aims to compute a new rank score for documents that reflects their original rankings. This is achieved by aggregating the inverse ranks from multiple ranked lists into a final score for each document. Algorithm 1 provides a pseudocode illustration of this process.



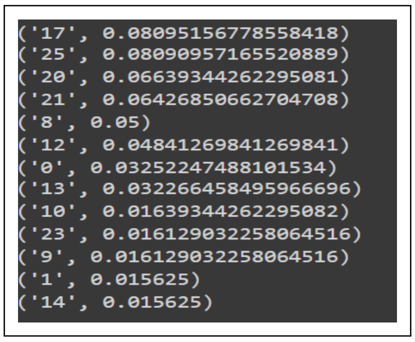
```
Input:
    $D$: List of all ranked lists for each sub-query
    $k$: Influence of the original rank on the final score
Output: $fusionList$: Re-ranked result

▷ Initiate an empty map of documents to reciprocal scores
$fusionList \leftarrow \{\}$

for each item $documentRankedList$ in $D$ do
    for $index \leftarrow 1$ to length($documentRankedList$) do
        $document \leftarrow documentRankedList[index]$
        $rank \leftarrow index$
        ▷ Initiate each document's score as $0$
        if $document \notin fusionList$ then
            $fusionList[document] \leftarrow 0$
        end if
        ▷ Add inverse rank score to each document's score
        $fusionList[document] \leftarrow fusionList[document] + \frac{1}{rank+k}$
    end for
end for
return $fusionList$
```

Figure 3.15: Algorithm of Reciprocal Rank Fusion implementation.

The influence coefficient k is set to 60, as recommended by Cormack et al. (2009). Although this value is not critical to the final re-ranking results, it helps balance the influence of the original rankings. With these parameters, RRF processes the combined ranked list from the previous stage to produce a re-ranked list of documents, each with its reciprocal score.



```
('17', 0.08095156778558418)
('25', 0.08090957165520889)
('20', 0.06639344262295081)
('21', 0.06426850662704708)
('8', 0.05)
('12', 0.04841269841269841)
('0', 0.03252247488101534)
('13', 0.032266458495966696)
('10', 0.01639344262295082)
('23', 0.016129032258064516)
('9', 0.016129032258064516)
('1', 0.015625)
('14', 0.015625)
```

Figure 3.16: Result of the re-ranking process.

3.6.3.3 Generation

LLM Agent Details

The LLaMA 3-8B-8192 model is employed for generating final responses and creating sub-queries. It is a high-performance model with a substantial context window, processing up to 30,000 tokens per minute (approximately 40 pages of text).

A critical parameter is the temperature, which influences the model's creativity. A lower temperature, such as 0.1, enhances output determinism and consistency by making predictions more predictable and reducing randomness. This setting helps ensure that the responses align closely with the provided resumes and queries, minimizing deviations and hallucinations.

Small-to-Big Retrieval

To avoid the limitations of processing only text chunks, the system retrieves the full resumes using metadata. This method, known as small-to-big retrieval, first fetches smaller chunks and then uses metadata to access the complete documents. This approach improves retrieval efficiency and ensures that the LLM has full contextual information for accurate assessments.

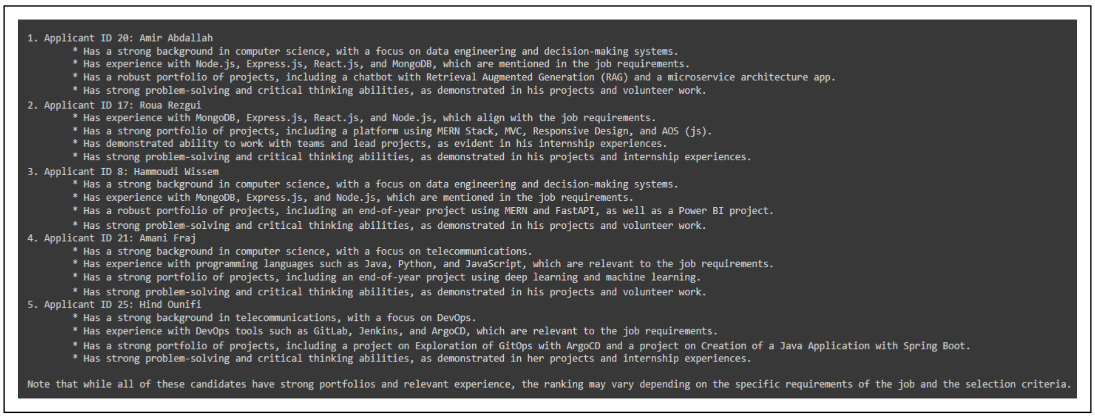
Prompt Design

The system uses a specialized prompt that integrates job descriptions and re-ranked resumes.

This prompt directs the model to summarize suitable resumes and explain its choices. The prompt template is tailored for job-resume matching but can be adjusted for other objectives in real screening processes.

The System Prompt provides general instructions and assigns the model a specific role, such as "You are an expert in talent acquisition..." to ensure relevant responses. It includes instructions to base responses solely on the given context ("Use only the following pieces of context...") and to generate "I don’t know" when necessary. This helps mitigate hallucinations and maintains response accuracy.

The User Prompt combines re-ranked resumes into a single text input, separated by line breaks, and uses job descriptions as the query. This setup ensures that the model has both the context and the specific requirements needed for generating accurate responses.



```
1. Applicant ID 20: Amir Abdallah
        * Has a strong background in computer science, with a focus on data engineering and decision-making systems.
        * Has experience with Node.js, Express.js, React.js, and MongoDB, which are mentioned in the job requirements.
        * Has a robust portfolio of projects, including a chatbot with Retrieval Augmented Generation (RAG) and a microservice architecture app.
        * Has strong problem-solving and critical thinking abilities, as demonstrated in his projects and volunteer work.
2. Applicant ID 17: Roua Rezgui
        * Has experience with MongoDB, Express.js, React.js, and Node.js, which align with the job requirements.
        * Has a strong portfolio of projects, including a platform using MERN Stack, MVC, Responsive Design, and AOS (js).
        * Has demonstrated ability to work with teams and lead projects, as evident in his internship experiences.
        * Has strong problem-solving and critical thinking abilities, as demonstrated in his projects and internship experiences.
3. Applicant ID 8: Hammoudi Wissem
        * Has a strong background in computer science, with a focus on data engineering and decision-making systems.
        * Has experience with MongoDB, Express.js, and Node.js, which are mentioned in the job requirements.
        * Has a robust portfolio of projects, including an end-of-year project using MERN and FastAPI, as well as a Power BI project.
        * Has strong problem-solving and critical thinking abilities, as demonstrated in his projects and volunteer work.
4. Applicant ID 21: Amani Fraj
        * Has a strong background in computer science, with a focus on telecommunications.
        * Has experience with programming languages such as Java, Python, and JavaScript, which are relevant to the job requirements.
        * Has a strong portfolio of projects, including an end-of-year project using deep learning and machine learning.
        * Has strong problem-solving and critical thinking abilities, as demonstrated in his projects and volunteer work.
5. Applicant ID 25: Hind Ounifi
        * Has a strong background in telecommunications, with a focus on DevOps.
        * Has experience with DevOps tools such as GitLab, Jenkins, and ArgoCD, which are relevant to the job requirements.
        * Has a strong portfolio of projects, including a project on Exploration of GitOps with ArgoCD and a project on Creation of a Java Application with Spring Boot.
        * Has strong problem-solving and critical thinking abilities, as demonstrated in her projects and internship experiences.

Note that while all of these candidates have strong portfolios and relevant experience, the ranking may vary depending on the specific requirements of the job and the selection criteria.
```

Figure 3.17: Response to the query: "The best 5 resumes that fit this job description."

3.7 Evaluation

Text Extraction

When using Python libraries for text extraction, we encountered several challenges:

- **Inconsistent Extraction Results:** The libraries often produced inconsistent outputs, particularly with complex layouts or closely related sections within the document.
- **Handling Complex Structures:** These tools struggled with documents featuring non-standard formatting or intricate structures, leading to difficulties in accurately capturing and organizing the extracted text.

In contrast, using Llama Parse for text extraction did not present any such problems; it effectively extracted all the text. **Person Detection**
For person detection, we passed 100 resumes through this pre-trained model. Out of these, 80 resumes contained candidate images, and the model successfully extracted 78 images, achieving an accuracy of 97.5%.

Links Extraction

For link extraction, we did not encounter any issues.

Features Extraction

We did encounter some issues with feature extraction:

- **Education Section:** The tool sometimes struggled to accurately identify the degree obtained, particularly if it was not explicitly mentioned in the resume. Additionally, it often inappropriately repeated the school name.
- **Internship and Project Sections:** The tool had difficulty extracting all mentioned technologies when they were not explicitly listed in the descriptions of internships or projects.
- **Volunteer Work Section:** When responsibilities were mentioned, the tool occasionally misinterpreted them as job titles, leading to inaccurate rewriting of position descriptions.

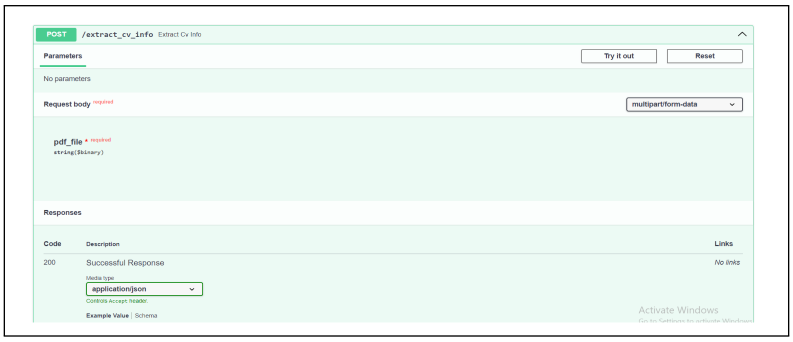
Chatbot

In developing the chatbot, we encountered several issues:

-
- **Inaccurate Information:** There were instances where the large language model (LLM) provided incorrect or misleading information about the resumes. This could be due to limitations in the LLM's understanding of context.

3.8 Deployment

First, we developed two server-side applications using FastAPI. In the resume parser application, we created an endpoint that accepts resumes and returns a JSON file containing all the relevant extracted information.


POST /extract_cv_info Extract Cv Info

Parameters

No parameters

Request body required — multipart/form-data

pdf_file * required
string($binary)

Responses

Code	Description	Links
200	Successful Response Media type: application/json Controls Accept header. Example Value | Schema	No links

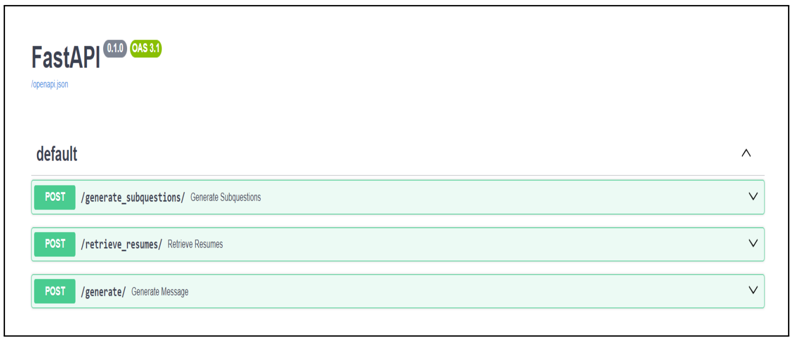
Figure 3.18: Reume parser API Endpoint Documentation.

In the chatbot application, we implemented three endpoints: Here's a detailed completion for each endpoint:

- **Subquery Generation Endpoint:** This endpoint processes a job description to generate subqueries that focus on specific skills, qualifications, and experiences mentioned in the job description. It breaks down the job requirements into smaller, actionable queries to enhance the accuracy of resume matching. The request includes the full job description, and the response is a list of subqueries relevant to different aspects of the job.

- **Resume Retrieval Endpoint:** This endpoint retrieves resumes based on the subqueries generated from the job description. It searches the resume database for candidates whose resumes match the criteria specified in the subqueries. The request includes a list of subqueries, and the response is a ranked list of resumes.

- **Chatbot Interaction Endpoint:** This endpoint allows users to interact with the chatbot to ask questions, request clarifications, or get more information about the resumes and job descriptions. Users can engage in a conversation with the chatbot to discuss specific candidates, seek further insights, or clarify job-related details. The request includes the user's message or query, and the response is the chatbot's reply.


FastAPI 0.1.0 OAS 3.1
/openapi.json

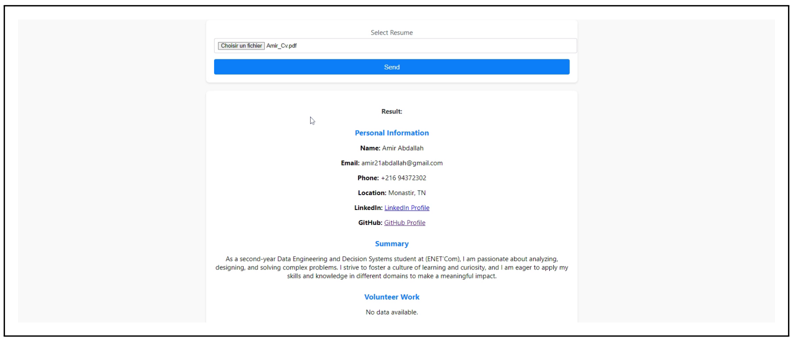
default

POST /generate_subquestions/ Generate Subquestions
POST /retrieve_resumes/ Retrieve Resumes
POST /generate/ Generate Message

Figure 3.19: Chatbot API Endpoint Documentation.

For security, the backend only accepts requests from the frontend, ensuring that no external sources can access the API. We also secured sensitive API keys by storing them in environment variables, hidden from the frontend, and enforced strict CORS policies to allow only specific frontend origins. Furthermore, we restricted the backend to accept only POST requests to enhance security and prevent unauthorized access.

Next, we created a React.js page where users can upload a resume and receive the extracted information.


Select Resume
Choisir un fichier Amir_Cv.pdf
Send

Result:

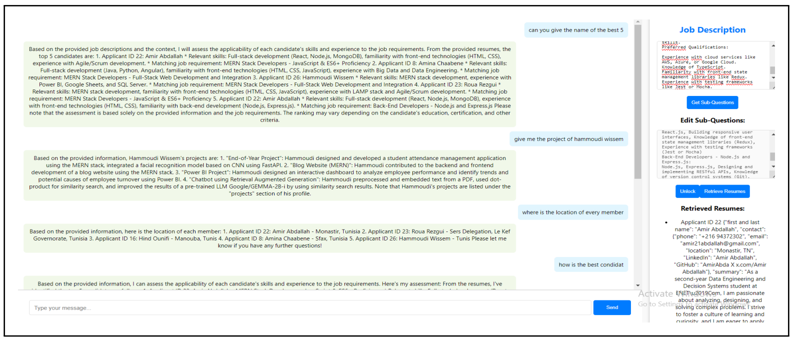
Personal Information
Name: Amir Abdallah
Email: amir21abdallah@gmail.com
Phone: +216 94372302
Location: Monastir, TN
LinkedIn: LinkedIn Profile
GitHub: GitHub Profile

Summary
As a second-year Data Engineering and Decision Systems student at (ENET'Com), I am passionate about analyzing, designing, and solving complex problems. I strive to foster a culture of learning and curiosity, and I am eager to apply my skills and knowledge in different domains to make a meaningful impact.

Volunteer Work
No data available.

Figure 3.20: Resume Parser Web Page.

Additionally, we created another React.js page where users can input a job description, generate subqueries, and modify them as needed. After making adjustments, the user can press a button to retrieve relevant resumes. Furthermore, the user can interact with an LLM (Large Language Model) to discuss the resumes and job description.



can you give the name of the best 5

Based on the provided job descriptions and the context, I will assess the applicability of each candidate's skills and experience to the job requirements. From the provided resumes, the top 5 candidates are: 1. Applicant ID 22: Amir Abdallah * Relevant skills: Full-stack development (React, Node.js, MongoDB), familiarity with front-end technologies (HTML, CSS), experience with Agile/Scrum development. * Matching job requirement: MERN Stack Developers - JavaScript & ES6+ Proficiency 2. Applicant ID 8: Amina Chaabene * Relevant skills: Full-stack development (Java, Python, Angular), familiarity with front-end technologies (HTML, CSS, JavaScript), experience with Big Data and Data Engineering. * Matching job requirement: MERN Stack Developers - Full-Stack Web Development and Integration 3. Applicant ID 26: Hammoudi Wissem * Relevant skills: MERN stack development, experience with Power BI, Google Sheets, and SQL Server. * Matching job requirement: MERN Stack Developers - Full-Stack Web Development and Integration 4. Applicant ID 23: Roua Rezgui * Relevant skills: MERN stack development, familiarity with front-end technologies (HTML, CSS, JavaScript), experience with LAMP stack and Agile/Scrum development. * Matching job requirement: MERN Stack Developers - JavaScript & ES6+ Proficiency 5. Applicant ID 22: Amir Abdallah * Relevant skills: Full-stack development (React, Node.js, MongoDB), experience with front-end technologies (HTML, CSS), familiarity with back-end development (Node.js, Express.js). * Matching job requirement: Back-End Developers - Node.js and Express.js Please note that the assessment is based solely on the provided information and the job requirements. The ranking may vary depending on the candidate's education, certification, and other criteria.

give me the project of hammoudi wissem

Based on the provided information, Hammoudi Wissem's projects are: 1. "End-of-Year Project": Hammoudi designed and developed a student attendance management application using the MERN stack, integrated a facial recognition model based on CNN using FastAPI. 2. "Blog Website (MERN)": Hammoudi contributed to the backend and frontend development of a blog website using the MERN stack. 3. "Power BI Project": Hammoudi designed an interactive dashboard to analyze employee performance and identify trends and potential causes of employee turnover using Power BI. 4. "Chatbot using Retrieval Augmented Generation": Hammoudi preprocessed and embedded text from a PDF, used dot-product for similarity search, and improved the results of a pre-trained LLM Google/GEMMA-2B-i by using similarity search results. Note that Hammoudi's projects are listed under the "projects" section of his profile.

where is the location of every member

Based on the provided information, here is the location of each member: 1. Applicant ID 22: Amir Abdallah - Monastir, Tunisia 2. Applicant ID 23: Roua Rezgui - Sers Delegation, Le Kef Governorate, Tunisia 3. Applicant ID 16: Hind Ounifi - Manouba, Tunis 4. Applicant ID 8: Amina Chaabene - Sfax, Tunisia 5. Applicant ID 26: Hammoudi Wissem - Tunis Please let me know if you have any further questions!

how is the best condidat

Based on the provided information, I can assess the applicability of each candidate's skills and experience to the job requirements. Here's my assessment: From the resumes, I've

```
Preferred Qualifications:

Experience with cloud services like
AWS, Azure, or Google Cloud.
Knowledge of TypeScript.
Familiarity with front-end state
management libraries like Redux.
Experience with testing frameworks
like Jest or Mocha.
```

```
React.js, Building responsive user
interfaces, Knowledge of front-end
state management libraries (Redux),
Experience with testing frameworks
(Jest or Mocha)
Back-End Developers - Node.js and
Express.js:
Node.js, Express.js, Designing and
implementing RESTful APIs, Knowledge
of version control systems (Git).
```

- Applicant ID 22 {"first and last name": "Amir Abdallah", "contact": {"phone": "+216 94372302", "email": "amir21abdallah@gmail.com", "location": "Monastir, TN", "LinkedIn": "Amir Abdallah", "GitHub": "AmirAbda X x.com/Amir Abdallah"}, "summary": "As a second-year Data Engineering and Decision Systems student at ENET\u2019Com, I am passionate about analyzing, designing, and solving complex problems. I strive to foster a culture of learning and curiosity, and I am eager to apply

Figure 3.21: Chatbot Web Page.

Finally, we crafted Dockerfiles for each of the four services: two frontends and two backends.These Dockerfiles encapsulate the environment and configuration settings required for each service. To efficiently manage and deploy these services, we utilized Docker Compose. Docker Compose is a powerful tool that simplifies the orchestration of multicontainer Docker applications. It allows us to define and coordinate all components of our application including services, networks, and volumes in a single ‘docker-compose.yml‘ file.

By using Docker Compose, we can manage our entire application setup with ease. This unified configuration enables us to start, stop, and coordinate all four services both frontends and backends with a single command, ensuring a streamlined and efficient deployment process.

3.9 CONCLUSION

In this chapter, we used the CRISP-DM framework to guide the development of our resume screening platform. We integrated LlamaParse for accurate text extraction, implemented an LLM agent to automate key information retrieval, and added features like person detection and period calculation. The system was deployed using FastAPI and a React.js frontend, with RAG Fusion improving the chatbot's ability to match resumes with job descriptions, resulting in a fully functional and scalable platform.

CONCLUSION GÉNÉRALE

This internship project focused on the development of a resume screening platform. Conducted in collaboration with the company DNEXT Intelligence SA, the main goal was to design a solution that enables recruitment teams to efficiently screen resumes and automate parts of the hiring process.

The platform offers a range of features that help recruiters in several ways. These include automating the initial screening of resumes, extracting essential candidate details, and generating structured data that can be used to assess applicant qualifications. The system uses large language models and RAG Fusion for high-quality data retrieval and processing, ensuring accurate matching between candidates' resumes and complex job descriptions.

We utilized the CRISP-DM framework to guide the project in a structured and iterative manner, ensuring alignment with the company's goals. By following its key phases—business understanding, data preparation, modeling, evaluation, and deployment—we continuously improved the platform, meeting DNEXT Intelligence SA's needs throughout the project.

Throughout the internship, we gained valuable skills in web application development,web scrapping,data extraction, and the use of large language models.

In conclusion, the resume screening platform represents a comprehensive and efficient solution for DNEXT Intelligence SA. It significantly improves the recruitment process by automating resume management, streamlining candidate evaluations, and ensuring better matches between job requirements and applicants. This project has been an enriching experience, allowing us to apply theoretical knowledge while developing practical skills that will be crucial for our future careers in data science and web development.

BIBLIOGRAPHIE

[1] **How many applications does it take to get a job?** Kolmar, C. (2023) **[en ligne]**. Disponible sur:
https://www.zippia.com/advice/how-many-applications-does-it-take-to-get-a-job/

[2] **. Attention is all you need.**. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Format PDF. Disponible sur:
https://arxiv.org/pdf/1706.03762

[3] **conSultantBERT: Fine-tuned Siamese Sentence-BERT for Matching Jobs and Job Seekers**.Lavi, D., Medentsiy, V., and Graus, D. (2021). Format PDF. Disponible sur:
https://arxiv.org/pdf/2109.06501

[4] **Bert: Pre-training of deep bidirectional transformers for language understanding.**. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Format PDF. Disponible sur:
https://arxiv.org/pdf/2109.06501

[5] **The cost of training nlp models: A concise overview.**. Sharir, O., Peleg, B., and Shoham, Y. (2020). Format PDF. Disponible sur:
https://arxiv.org/pdf/2004.08900

[6] **CRISP-DM Phase 1: Business Understanding**. Zipporah Luna (2021).**[en ligne]**.Disponible sur:
https://medium.com/analytics-vidhya/crisp-dm-phase-1-business-understanding-255b47adf90a

[7] **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**.Patrick Lewis. Format PDF. Disponible sur:
https://arxiv.org/pdf/2005.11401

[8] **Introduction to llm agents**.Tanay Varshney. **[en ligne]**.Disponible sur: https://developer.nvidia.com/blog/introduction-to-llm-agents/

[9] **Reciprocal rank fusion outperforms condorcet and individual rank learning method**.Cormack, G., Clarke, C., Büttcher, S. (2009). **[en ligne]**.Disponible sur: https://plg.uwaterloo.ca/ gvcormac/cormacksigir09-rrf.pdf

Resume Screening Application

Hammoudi Wissem

Résumé:

Ce projet de filtrage de CV vise à automatiser le processus de sélection des candidats en utilisant des techniques de RAG Fusion et des agents d'intelligence artificielle. L'application analyse les CV en identifiant les compétences, expériences professionnelles, formations, et projets, afin de les comparer aux exigences des descriptions de poste. En réduisant le temps de traitement et en augmentant la précision, cette solution permet d'améliorer la qualité du recrutement tout en minimisant les biais humains.

Mots clés: Filtrage de CV, RAG Fusion, LlamaParse, recrutement.

Abstract:

This resume screening project aims to automate the candidate selection process by utilizing RAG Fusion techniques and AI agents. The application analyzes resumes by identifying skills, professional experiences, education, and projects, and matches them against job descriptions. By reducing processing time and increasing accuracy, this solution improves recruitment quality while minimizing human bias.

Key-words: Resume screening, RAG Fusion, LlamaParse, recruitment.