Reykjavík University

Department of Computer Science

University of Camerino

School of Science and Technology

Master of Science in Computer Science

Enhancing Documents Review through Knowledge Graphs and Large Language Models

Candidate

Pietro Angelici

Student ID: 119982

Supervisors

Prof. Dr. Marco Piangerelli

Prof. Dr. Stefán Ólafsson

Ing. Emanuele Bezzecchi

A.Y. 2022/2023

I dedicate this work to those who, like me, see in the science of computing not just formulas and algorithms, but a passport to explore new horizons.
Every line of code, every problem solved, has been one more step in my personal journey, a journey that, just like my actual travels, has enriched me and prepared me for future adventures.
May this milestone not be a conclusion, but an invitation to travel further, to discover new lands in the infinite universe of computer science and beyond.

Contents

List of Figures

CHAPTER 1

Introduction

In a time marked by a continuous and rapid increase in data volumes, the ability to extract meaningful insights and navigate effectively through intricate document collections emerges as fundamental pillars for advancement in various sectors. Acquiring knowledge from extensive and complex data sources is not just a challenge, but also an unmissable opportunity to push the boundaries of innovation and progress [74].

The advent of digital transformation [115] has inaugurated a new era in data analysis, introducing, however, significant challenges in managing and interpreting voluminous sets of documents. This thesis work proposes to explore how an interactive and visual approach to document analysis can radically innovate the perception and understanding of information. The designed system, developed during this study, stands as a bridge between the accessibility of information and the human ability to interpret it, providing a revolutionary solution for the retrieval, visualization, and analysis of data in an intuitive and detailed manner.

The continuously increasing availability of data does not automatically translate into effective access and interpretation of information. Traditional methodologies of document analysis clash with their static and non-interactive nature, hindering effective data exploration and limiting individuals' ability to discover meaningful insights and make data-driven decisions [79].

The ineffectiveness of traditional methods of data analysis and visualization manifests in multiple contexts, impairing users' ability to navigate, interpret, and fully leverage complex information. This limitation constitutes a significant obstacle not only in the academic and research world but also in critical sectors like business, medicine, and public administration, where the ability to make informed and timely decisions is of vital importance.

Existing solutions for document analysis present significant limitations, including the lack

of interactivity, difficulty in the semantic interpretation of data, and the absence of an integrated solution that harmonizes interactivity, intuitive visualization, and semantic analysis [123]. In response to these gaps, the project leverages emerging technologies and cutting-edge approaches, such as interactive knowledge graphs, advanced Natural Language Processing (NLP) [58], and Large Language Models (LLM) [86], to offer a revolutionary user experience in the field of document analysis and overcome current limitations.

The designed system marks a significant advancement in this field. The platform not only transforms static documents into dynamic structures but also enables users to interact with data through natural language queries, offering an intuitive visualization based on knowledge graphs [57] that facilitates the identification of patterns and correlations. The project transcends the boundaries of traditional methods, facilitating an intuitive understanding and in-depth semantic analysis of data, and inaugurating new modes of interaction and analysis. Sources of inspiration for this development include innovative technologies and approaches as illustrated in the Graphlogue [99] and ChatPDF [13] papers.

In the system evaluation, a detailed analysis of its performance and usability was conducted, highlighting strengths and areas for improvement. The creation of navigable Knowledge Graphs (KG) [52] has proven to provide a comprehensive view of the information contained in the documents.

A Knowledge Graph represents a structured way of organizing knowledge and the connections between various concepts. It serves to structure information such that the interrelations among different entities can be comprehended, facilitating a more interconnected and comprehensive understanding of a particular knowledge domain.

Within a Knowledge Graph, information is structured as nodes and edges in a graph format. Nodes symbolize entities like individuals, locations, concepts, or occurrences, whereas edges depict the relationships among these entities. This structuring enables a more precise and understandable depiction of the interactions and connections between various entities.

The results of the analysis confirmed the system's excellence in delivering a satisfactory user experience. However, opportunities have emerged to further optimize performance and user interface to ensure smoother navigation and deeper analysis of information. The innovative approach adopted in the project has opened new perspectives in data exploration and analysis, suggesting potential future developments to further enhance users' ability to access and understand information.

The PDF (Portable Document Format) document [120], despite its universality, presents significant challenges in information extraction and structuring, making the evolution of document processing technologies crucial. This project emerges in this context as an innovative solution, leveraging and integrating the latest technologies to overcome existing limitations. Information retrieval is highly important in various sectors.

In academia, efficient information retrieval from PDF documents is paramount. Researchers and students rely on this process to access critical data contained in publications, articles, and books. Effective data extraction and structuring can significantly accelerate literature review, data analysis, and knowledge synthesis.

In the business world, the ability to quickly extract information from PDF documents is essential for making informed decisions. Effective structuring of this data fuels market analysis, financial reporting, and operational strategies, transforming raw data into valuable insights.

In the public sector, the digitization of services has led to an increase in the use of PDF documents. The ability to efficiently retrieve and structure information from these documents is crucial for improving operational efficiency, transparency, and citizen access to services.

This research lays the foundation for future developments in the field of data analysis. The achieved results highlight the potential of systems like the project to revolutionize the way we interact with information, outlining new directions for innovation and system improvement.

CHAPTER 2

Scientific and Technological Context

In this chapter, we will delve into the scientific and technological landscape associated with constructing knowledge graphs from PDF documents [11], utilizing NLP methodologies, focusing on recent studies and research of this new technology. We will analyze traditional approaches, highlighting their main limitations, and then outline the latest innovations in terms of tools, technologies, algorithms, languages, and standards. This analysis will allow us to deeply understand how the new solutions overcome the difficulties encountered by previous methodologies.

The state of the art in constructing knowledge graphs from PDF documents, particularly through the use of NLP, has evolved significantly [82]. Recent advancements focus on overcoming traditional challenges such as ambiguity and impreciseness in natural language, employing sophisticated entity and relation extraction techniques to build comprehensive knowledge graphs from vast amounts of textual data.

A notable approach is the end-to-end construction of an NLP KG from scientific papers, as discussed in a study by Mondal, Hou, and Jochim [83]. This research introduces novel methods for extracting relations between entities within scientific papers, applying the developed framework (SciNLP-KG) to a large dataset of NLP papers to generate a knowledge graph that facilitates the construction of scientific leader-boards for the NLP community. The framework focuses on extracting four types of relations—evaluated On between tasks and datasets, evaluated By between tasks and evaluation metrics, as well as co-referent and related relations between the same type of entities—demonstrating the ability to produce a high-quality knowledge graph. However, the study encounters limitations such as semantic and contex-

tual ambiguity, which complicate the precise identification of relations, and a limited generalizability across different scientific domains. Additionally, the quality of the generated knowledge graph heavily depends on the completeness and accuracy of the extracted data, which poses a challenge for ensuring comprehensive and accurate knowledge representation.

Furthermore, a tutorial overview by Aggarwal, Bhatia, Shekarpour, and Sheth provides a comprehensive introduction to knowledge graphs, detailing curated and automatic construction approaches, including fact extraction methods and the use of semantics and structure for enhancing cognitive capabilities in various applications [2]. This tutorial emphasizes the significance of knowledge graphs in text analytics, highlighting their ability to recognize entities based on underlying background knowledge, thereby providing more abstract semantics beyond statistical text analysis. It also discusses traditional problems such as Entity Recognition (NER) and Disambiguation, alongside emerging problems in NLP tasks, question-answering, and machine learning, using features extracted from the semantics and structure of knowledge graphs. The integration of curated and automatic methods presents a complex challenge, balancing the accuracy of manual approaches with the scalability of automatic processes. Additionally, the tutorial acknowledges the ongoing struggle with natural language ambiguity, which significantly impacts the effectiveness of entity recognition and disambiguation techniques.

Moreover, IBM Research's publication underscores the importance of knowledge graphs in modern applications, emphasizing the challenge of automatically constructing KGs from natural language texts due to language ambiguity [17]. The presentation summarized research progress over KG construction from text, focusing on information acquisition, entity and relation extraction, and the current state of the art methods, techniques, and tools for constructing knowledge graphs from text. This work aims to provide an overview of the field, discussing capabilities, limitations, and current challenges. It highlights the difficulties in automatic KG construction, including the accurate interpretation of linguistic ambiguity and the extraction of entities and relations. The limitations of existing tools in handling linguistic diversity and understanding the necessary context for accurate entity and relation extraction are also emphasized. Moreover, balancing scalability with the precision of extraction processes remains a significant hurdle, indicating a need for further research and development in this area.

In the paper titled "A comprehensive survey on automatic knowledge graph construction" by Zhong, Lingfeng et al., published in ACM Computing Surveys in 2023 [126], the authors delve into the landscape of techniques and methodologies for the automatic construction of knowledge graphs. This study provides a comprehensive overview of the various phases of the process, from data collection and entity extraction to relationship extraction, graph construction, and population. Among the strengths of this work is the extensive analysis of various

approaches to the problem, including machine learning techniques, rule-based methods, and the utilization of advanced NLP frameworks. The authors highlight how integrating these technologies can overcome some of the most common limitations, such as natural language ambiguity and the challenge of handling unstructured data at scale.
However, the work also underscores significant drawbacks and challenges. Among the main limitations is the difficulty in ensuring the accuracy and completeness of knowledge graphs in the presence of heterogeneous and large volumes of data. Furthermore, issues related to semantic interpretation and context remain a substantial challenge, especially when dealing with complex texts or domain-specific content. The generalizability of the proposed techniques is another critical point, as many solutions demonstrate optimal performance only in limited contexts or with certain types of data.

These advancements demonstrate the dynamic and evolving nature of knowledge graph construction from PDF documents using NLP, highlighting a transition towards more sophisticated, automated methods that promise enhanced accuracy, efficiency, and utility in various domains.

2.1 Traditional Approaches and Their Limitations

Let's examine in detail the limitations of traditional approaches in the field of text extraction and knowledge graph construction, focusing particularly on text extraction based on fixed rules, context-free syntactic analysis, and the lack of scalability and adaptability. [117]

Text Extraction Based on Fixed Rules

Fixed-rule approaches utilize a predefined set of instructions to identify and extract information from documents. These can include regular expressions, text patterns, or specific layout indicators (such as headings or bullet points).

Inability to Handle Variability: Real-world documents often exhibit significant variability in terms of layout and structure. This includes variations in formatting, font usage, and the arrangement of elements on the pages. Fixed rules are incapable of adapting to these variations, resulting in imprecise or incomplete extractions.
Difficulty with Complex Document Structure: Documents with complex structures, such as those containing multiple columns, footnotes, or graphic inserts, can easily confuse tools based on fixed rules, leading to extraction errors or the loss of critical information [114].

Maintenance and Scalability: The creation and maintenance of fixed rules require significant manual effort and a deep understanding of the target documents. In environments where doc-

uments change frequently or where various types of documents need to be handled, keeping the rules updated becomes impractical.

Context-Free Syntactic Analysis

These approaches focus on analyzing the grammatical structure of the text, identifying parts of speech, subjects, verbs, objects, etc., without considering the broader context in which the words are used.

Issues with Polysemy and Disambiguation: Many terms in natural language have more than one meaning (polysemy). Without contextual analysis, it is impossible to determine which meaning is the correct one in a given instance, leading to misinterpretations. Lack of Contextual Understanding: Context-free syntactic analysis fails to capture the nuances of language that are determined by the broader context. This includes the inability to correctly interpret sarcasm, metaphors, or implicit language, which are all crucial elements for human understanding of the text [22].

Lack of Scalability and Adaptability

Traditional approaches are often tailor-made for specific tasks or domains and are not easily adaptable to new contexts or to increasing data volumes.

System Rigidity: Systems designed for specific tasks or domains often assume certain constants regarding the data structure or the nature of information requests. This rigidity makes them ill-suited to adapt to new types of documents or evolving extraction requirements.
Difficulty in Managing Large or Variable Data: With the increase in data volume and the variety of document formats, maintaining the system's effectiveness and efficiency becomes an increasingly complex and costly task [84].

2.2 New Tools and Technologies

Advancements in the field of Natural Language Processing (NLP) and related technologies have led to the development of new solutions capable of overcoming the limitations of traditional approaches. [12]

AI-Based Language Models

AI-based language models, such as GPT (Generative Pre-trained Transformer) [51] and BERT

(Bidirectional Encoder Representations from Transformers) [20], represent a revolution in NLP. They are based on deep neural networks [112] and are trained on vast datasets, learning to understand and generate natural language. These models excel in analyzing context and grasping semantic nuances, far outperforming older systems based on rigid rules.

These models have a wide range of applications, from automatic translation to voice recognition, from text generation to the creation of advanced chatbots [1]. Their flexibility and learning capability make them versatile tools, capable of offering innovative solutions in various fields, improving the understanding of natural language and facilitating more natural human-machine interactions.

Semantic Extraction Frameworks

Frameworks like SpaCy [1] or NLTK [2] provide advanced tools for the semantic analysis of texts. These tools can identify entities (such as people, places, or things), relationships between entities, and understand the contextual meaning of words in sentences or paragraphs.

The use of these frameworks allows for accurately interpreting complex texts, building detailed knowledge graphs, and analyzing large volumes of text efficiently. They are particularly useful in fields like sentiment analysis or data-driven decision support, where a deep understanding of content is crucial.

Advanced Visualization and Interaction Technologies

Tools like D3.js [3] or Sigma.js [4] enable interactive visualization of knowledge graphs, facilitating the understanding of complex relationships between data. These tools make data not just visually appealing but also interactive, allowing users to explore and manipulate visualizations for deeper insights.

These technologies enhance the usability of knowledge graphs, making them more accessible and intuitive. They allow users to directly interact with the data, exploring and understanding it more effectively, which is particularly useful in areas like scientific research, business intelligence, and strategic planning.

Open Standards and Interoperability

[1] SpaCy: [24]
[2] NLTK: [88]
[3] D3.js: [18]
[4] Sigma.js: [108]

Cite	NLP Technologies	Pros	Cons
[83]	Entity Extraction (EE) and Relation Extraction (RE), SciNLP-KG framework	High-quality Knowledge Graph generation; Focus on specific relations	Semantic and contextual ambiguity Limited generalizability; Data dependency
[2]	Fact Extraction (FE), Semantic (SA), and Structural (ST) Analysis	Enhancement of cognitive capabilities; Entity recognition based on background knowledge	Balance between manual accuracy and automatic scalability; Language ambiguity issues
[17]	Information Acquisition (IA), Entity Extraction (EE), and Relation Extraction (RE)	Discussion on current methods; techniques and tools for KG construction	Challenges with linguistic ambiguity; Limitations of existing tools
[126]	Machine Learning (ML), Rule-Based Methods (RBM), Advanced NLP frameworks (ANLP)	Extensive analysis of techniques and methodologies; Integration of technologies to overcome common limitations	Challenges in KG accuracy and completeness; Semantic interpretation and contextual issues

The adoption of open standards like RDF (Resource Description Framework) [98] and OWL (Web Ontology Language) [8] promotes interoperability and data sharing. They provide a framework for representing information in a structured and semantic way, facilitating integration between different systems and applications.

Thanks to these standards, it is possible to integrate and reuse knowledge graphs across different applications, greatly expanding their value. They enable linking data from diverse sources, gaining new insights, and supporting complex decision-making processes. The evolution of tools and methodologies in the field of NLP and information management is opening new and exciting prospects. These advancements promise to revolutionize the way we access, interpret, and use information, making decision-making processes more informed and human-machine interactions more natural and intuitive.

2.3 Our Utilized Tools

In light of the various constraints encountered, the need to create more refined approaches for analyzing PDF documents became evident. The primary goal was not limited to mere text extraction but aimed at accurately preserving and deciphering the structure, layout, and

graphical elements of the document, significantly improving search precision. This need has led to the evolution of more sophisticated methodologies, including the use of Optical Character Recognition (OCR)[61] and the application of advanced artificial intelligence models, in order to obtain a more detailed and comprehensive semantic understanding of the content.

Text Extraction with OCR

OCR (Optical Character Recognition) represents a significant leap in document processing, overcoming many of the limitations associated with basic text extraction. This technology transforms physical documents and non-interactive PDF files, such as text scans, into editable and searchable digital formats, opening up new possibilities for data storage, retrieval, and analysis.

OCR operates through a process of recognizing and digitizing characters present in images of documents. This process can be divided into several stages:

- **Pre-processing of the Image:** Before recognizing the text, the image is optimized to improve the quality and readability of characters. This may include orientation correction, background noise removal, and contrast adjustment.
- **Character Recognition:** Using advanced algorithms, OCR analyzes the image and identifies the characters present. This process can be supported by machine learning [7] and artificial intelligence [55] techniques to enhance recognition accuracy, especially in the presence of different fonts or variable image quality.
- **Text Conversion and Formatting:** The recognized characters are converted into digital text. OCR strives to preserve the original formatting of the document, including spaces, paragraph breaks, and font styles, to maintain fidelity to the original document.

One of the advantages of OCR is content accessibility. This is where converting an image into digital text format proves invaluable. By doing so, OCR enables the content to become searchable and analyzable, overcoming the limitations posed by physical formats or low-quality scans. This accessibility opens up numerous possibilities for leveraging the content in various ways, such as research, data analysis, and information retrieval.

Another significant advantage of OCR is its contribution to efficiency in storage and retrieval processes. Once text is digitized, it becomes much easier to store, organize, and retrieve. This digitization eliminates the need for physical storage space and reduces the time and effort required to sift through documents manually. With OCR, document management becomes streamlined, leading to increased productivity and cost savings for businesses and organizations.

Additionally, OCR integrates seamlessly with NLP technologies, offering further advantages. The text extracted through OCR can be subjected to various NLP techniques for semantic analysis, information extraction, automatic translation, and more. This integration enhances the capabilities of both OCR and NLP, enabling sophisticated processing of textual data for a wide range of applications, including text summarization, sentiment analysis, and language understanding.

Syntactic and Semantic Analysis using NLP

Natural Language Processing (NLP) represents an advanced field of artificial intelligence and computational linguistics focused on the interaction between computers and human language. NLP employs a range of techniques and models to analyze, understand, and generate natural language, paving the way for a wide range of applications, from sentiment analysis [109] to document categorization, named entity recognition (NER) [68] to text generation. NLP relies on simple linguistic models and statistical approaches to analyze the structure and meaning of text. These include methods such as tokenization [96], morphological analysis [5], POS tagging [14], and syntactic parsing [89].

NLP Fundamentals

Tokenization: Tokenization is the process of dividing text into smaller units called tokens, which can be words, phrases, or symbols. This step is essential to simplify analysis and prepare the text for further processing.
Morphological Analysis: Morphological analysis involves examining the structure of words, identifying roots, suffixes, prefixes, and inflections. This process helps understand the role and meaning of words in context, contributing to the semantic understanding of the text.
Part-of-Speech (POS) Tagging: POS tagging involves assigning a grammatical category, such as noun, verb, adjective, etc., to each token. This classification is crucial for analyzing the grammatical structure of sentences and understanding relationships between words.
Syntactic Parsing: Syntactic parsing focuses on analyzing the grammatical structure of sentences by building a syntactic tree that represents the grammatical hierarchy and dependencies between words. This allows for understanding the logical structure of the text and identifying subjects, predicates, objects, and other syntactic elements.

Named Entity Recognition (NER)

NER is one of the techniques in NLP that aims to identify and classify specific entities mentioned in a text. Entities can include people, organizations, locations, dates, numbers, currencies, and other relevant information.

The fundamental goal of NER is to extract organized data from the text by recognizing entities and assigning precise classifications to each one. For example, in a sentence like 'Microsoft

has its headquarters in Redmond,' NER can identify 'Microsoft' as an entity belonging to the organizations category and 'Redmond' as an entity related to a location.

This type of technology is widely used in applications such as sentiment analysis, information extraction, document indexing, and other NLP tasks that require understanding and organizing information present in texts.

Large Language Models (LLMs)

LLMs represent a significant breakthrough in the field of artificial intelligence and NLP. Thanks to their ability to understand and generate natural language, these models offer innovative solutions to overcome the limitations of traditional methods for querying and analyzing PDF files.

An LLM is an artificial intelligence model based on deep neural networks, specifically on Transformer architectures [48], trained on vast amounts of text. Thanks to its extensive knowledge base [127] and advanced attention mechanism, an LLM can understand and generate natural language with an unprecedented level of sophistication.

A notable example of an LLM is GPT (Generative Pre-trained Transformer) [122], developed by OpenAI. GPT is among the most advanced language models and has an open-source version (GPT-3.5-turbo) and a paid version (GPT-4-turbo) at the time of writing this thesis [94].

Training and Operation:

LLMs are trained on vast textual datasets, learning linguistic patterns, relationships between concepts, and various language structures. Once trained, through input text (prompts), they are capable of generating text, answering questions, translating languages, synthesizing information, and performing many other language-related tasks.

A prompt refers to a text string or input provided to a model in order to obtain a desired response or text generation. In language models like GPT, the prompt is the initial input that is presented to the model to generate a response.

The choice of the prompt is often crucial in determining the quality and relevance of the response obtained from the model. By varying the prompt, it is possible to obtain different responses. The ability to formulate effective prompts is an important part in interacting with prompt-based language models.

When it comes to understanding the context, LLMs offer significant advantages. Unlike traditional keyword-based search engines, LLMs possess semantic depth, enabling them to grasp the context in which words or phrases are utilized within PDF documents. This capability leads to a more precise interpretation of content, enhancing the accuracy of searches and analyses conducted using LLMs.

Furthermore, LLMs excel in handling ambiguity and polysemy, thanks to their contextual understanding abilities. They can discern between various meanings of words or phrases based on the context provided within the document. This skill is crucial for accurate comprehension and interpretation, especially when dealing with content that may contain multiple interpretations or ambiguous language. By effectively navigating ambiguity, LLMs contribute to improved understanding and analysis of textual data, making them valuable tools for various applications, including information retrieval and natural language processing tasks.

Complex Queries and Answer Generation:

Natural Language Queries: LLMs can handle queries posed in natural language, understanding the user's intention and providing relevant and contextualized answers. **Synthesis and Summary:** In addition to answering specific questions, LLMs can synthesize and summarize information, extracting key concepts and presenting the information in a concise and comprehensible manner [125].

Adaptability and Continuous Learning:

Learning from New Data: LLMs can continue to learn and adapt to new data and information, improving their performance over time and adapting to new domains and language styles. **Personalization:** LLMs can be tailored or "fine-tuned" [64] on specific datasets or tasks, allowing for high customization to meet specific needs for data querying and analysis.

LLMs have revolutionized the field of NLP and document querying, offering powerful and versatile tools for information retrieval and language understanding. However, it's also important to consider the challenges associated with LLMs, such as the need for large amounts of data for training, the risk of data bias, and computational complexity. Despite these challenges, the advantages offered by LLMs in overcoming the limitations of traditional methods make them a promising and continually evolving frontier in the field of artificial intelligence and information retrieval.

Retrieval-Augmented Generation (RAG)

RAG [67] is an advanced approach in the field of artificial intelligence and NLP that combines two fundamental components: information retrieval [116] and text generation. The goal of RAG is to enhance the quality and relevance of generated text by incorporating contextual information extracted from an external knowledge base or a database of documents. Here is a detailed overview of the main characteristics of RAG:

The RAG method consists of two main steps:

Retrieval (Information Retrieval): This component is responsible for identifying and re-

trieving relevant parts of text or documents from a vast data collection. Retrieval is based on semantic search techniques that can include keyword matching, text embedding [4], or other methods of semantic understanding [73]. The goal is to find information that is relevant to the given request or context.

Generation (Text Generation): Once relevant information is retrieved, the generation component uses advanced language models (like GPT or similar LLMs) to create responses or texts that are coherent, informative, and well-formulated. Text generation is based not only on the context provided by the user but also on the additional information retrieved in the previous phase.

Operation of RAG

User Input: The system receives a query or a request from the user. **Information Retrieval:** The retrieval module searches its resources (like a database of documents) to find content relevant to the user's input. **Processing and Selection:** The retrieved information is processed, and the most pertinent parts are selected to be used in text generation. **Text Generation:** The text generation model uses both the initial user input and the retrieved information to generate a coherent and contextually rich response.

Advantages of RAG

More Informed Responses: By incorporating specific information retrieved from external sources, RAG is capable of generating responses that are not only linguistically correct but also content-rich and specific to the given context.

Flexibility and Breadth: RAG can draw from a wide range of information and sources, making it possible to generate text on a variety of detailed and diverse topics.

Learning and Adaptability: With the continuous updating of the database or source documents, the RAG system can stay up-to-date with the most recent and relevant information.

RAG represents a significant innovation in the field of NLP and is finding application in various fields, from the creation of intelligent virtual assistants to data-driven decision support, demonstrating its effectiveness in providing responses that are both contextually relevant and rich in information.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a transformative language model that has revolutionized the field of Natural Language Processing (NLP) and represents a significant evolution in text representation, allowing for the capture of the complex-

ity and subtlety of human language [21].

BERT excels in analyzing the context in which words appear, enabling it to interpret the meaning of words based on their actual use in sentences. Words or phrases that share similar meanings are represented by vectors that are located close together in the vector space, thereby facilitating the task of identifying and grouping similar semantic content.

SBERT: Optimizing BERT for Sentence Comparison

Sentence-BERT (SBERT) [105] is a variant of BERT optimized for directly comparing sentences, making operations like semantic search and text clustering more efficient. SBERT modifies the architecture of BERT to produce vector representations of entire sentences, rather than individual words. This allows for the direct calculation of semantic similarity between two sentences without having to compare each word individually.

Thanks to SBERT, it's possible to obtain vector representations that preserve the context and overall meaning of sentences. This paves the way for a wide range of applications, such as searching for similar documents, classifying texts, and other activities that require a deep understanding of text semantics.

Measuring Similarity with Cosine Similarity

Cosine similarity [72] is a crucial metric in the context of BERT and SBERT for evaluating how close two vectors (and therefore two sentences) are to each other in the vector space. This metric measures the cosine of the angle between two vectors, providing an assessment of their directional similarity regardless of their magnitude.

The calculation of cosine similarity is particularly useful in comparing sentences because it allows for the capture of semantic similarity, ignoring the length of the sentences or the frequency of words. Sentences with similar meanings will have vectors with a similar direction, resulting in a cosine similarity closer to 1.

The calculation is the dot product of the vectors divided by the product of the Euclidean norm of each vector. The formula is as follows:

$$\text{Cosine Similarity } (\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

where:

- $\mathbf{A} \cdot \mathbf{B}$ is the dot product between the vectors $\mathbf{A}$ end $\mathbf{B}$,
- $\|\mathbf{A}\| \cdot \|\mathbf{B}\|$ are the Euclidean norms of the vectors $\mathbf{A}$ end $\mathbf{B}$.

The result of cosine similarity ranges from -1 to 1. A value of 1 indicates that the vectors

are identical, 0 indicates

Data Visualization: Knowledge Graph

Knowledge Graphs were created by Google and first announced on May 16, 2012. A Knowledge Graph is a structured representation of knowledge and relationships between different concepts. It is a way of organizing information so that it is possible to understand the connections between different entities and gain a more comprehensive and interconnected view of a specific domain of knowledge.

In a Knowledge Graph, information is organized as nodes and edges in a graph. Nodes represent entities such as people, places, concepts, or events, while edges indicate the relationships between them. This approach enables a more accurate and understandable modeling of how different entities are connected and interact.

Knowledge Graphs are widely used in various sectors, including search engines, artificial intelligence, virtual assistants, and data analysis applications, as they provide a powerful structure for organizing and understanding large amounts of complex information.

Data Visualization: UMAP

Uniform Manifold Approximation and Projection (UMAP) [110] represents a significant advancement in the field of data analysis and visualization. This algorithm, introduced by McInnes, Healy, and Melville in 2018, is based on a mathematical conception and an intuitive understanding of data structure. UMAP is known for its versatility and processing speed. It can be applied to a wide range of data types and easily adapts to different contexts and requirements, ensuring fast and reliable results. Below are some advantages of using UMAP:

High Dimensional Representation: UMAP starts from the assumption that high-dimensional data can be faithfully represented in lower-dimensional spaces, preserving both local and global topological structure. **Topological Approach:** Unlike other dimensionality reduction methods, UMAP leverages concepts from topology, a branch of mathematics that studies properties of spaces preserved through continuous transformations. This allows UMAP to better capture the intrinsic structure of the data.

CHAPTER 3

High-Level Contribution

The developed system represents a significant advancement in PDF document processing and analysis, surpassing the limitations of traditional solutions and providing an intuitive and interactive user interface. Below, a detailed overview of the system's features, its applicability, how it addresses existing challenges, usage modes, and some practical examples are provided.

3.1 System Features

The system is designed to transform the way we interact with PDF documents, offering the following key features:

PDF Document Analysis and Processing

The system takes a PDF file as input and subjects it to in-depth analysis. This analysis goes beyond text extraction and includes understanding the document's structure, identifying chapters, paragraphs, and images, as well as extracting meaningful entities and relationships. All of this is made possible by the presence of an index in the PDF document, as information about chapters and reference pages is extracted from the index.

Interactive Knowledge Graph Generation

Based on the document analysis, the system generates a dynamic and interactive knowledge graph. This graph not only visually represents the document's structure and content but also allows the user to explore relationships between entities, navigate between different chapters

and paragraphs, and view information in an intuitive and accessible format.

Natural Language Querying of the Document

One of the most innovative features of the system is its ability to query the document using natural language. Users can ask specific questions about the document's content, and the system, leveraging advanced natural language processing techniques, provides precise answers, indicating the sections of the document from which the information was extracted.

Intuitive and Interactive Visualization

The system features a graphical user interface that facilitates interaction with the knowledge graph and the results of document analysis, including details about the paragraphs from which the answer was derived, such as the paragraph, chapter, and page. Users can view the graph, explore relationships between entities, navigate through document sections, and see the answers to their queries in a clear and organized format.

3.2 Overcoming Existing Challenges

The introduced system represents an innovative solution to challenges traditionally associated with PDF document processing and analysis. It distinguishes itself by its ability to address and overcome two main problems that have limited known open-source solutions:

Detailed Reconstruction of PDF Structure

One of the most significant challenges in processing PDF documents has been the difficulty of accurately reconstructing their internal structure. The developed system overcomes this limitation by providing a detailed and faithful representation of the document's structure. Unlike other open-source solutions, the system does not limit itself to a superficial text analysis but identifies and organizes chapters, paragraphs, and images, offering a comprehensive understanding and intuitive navigation of the document.

Intuitive Natural Language Querying of the Document

Another limitation of previous solutions was the inability to effectively and naturally query the contents of PDF documents. The developed system introduces a feature that allows users to ask questions in natural language and receive precise answers based on the document's content. This not only enhances the accessibility and interactivity of the system but also pro-

vides detailed information about the location of the answers within the document, such as the page, chapter, and reference paragraph. The ability to provide contextualized and well-located answers within the document represents a significant advancement compared to traditional document search and analysis capabilities.

3.3 User Guide for System Usage

The system is designed to offer an intuitive and interactive user experience through a single-page application [38]. The interface is organized to facilitate access to various features and ensure smooth and consistent navigation of the content. Below, the structure of the interface and its key functionalities are described.

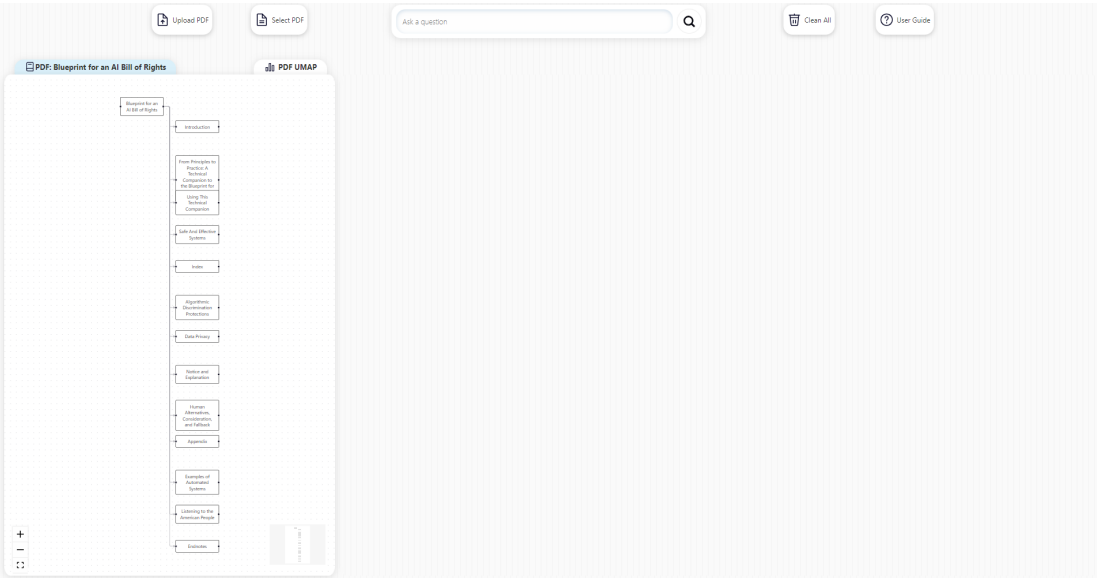

Figure 3.1: Initial view of the system.

User Interface Elements

Search Bar and Query Functionality:

At the top of the interface (Figure 3.1), you will find the search bar where you can write questions related to the selected PDF document. Next to the search bar, there is a magnifying glass icon that, when clicked, submits the question to QuestGraph (alternatively, you can use the Enter key).

Document Management

To the left of the search bar, two buttons are placed offering the following functionalities:

- Uploading a new PDF file.

- Selecting a previously uploaded PDF document in QuestGraph through a drop-down.

Application Cleanup and Guidance

To the right of the search bar, two buttons are placed offering the following functionalities:

- Clearing the graphical interface to remove displayed data.
- Viewing the application's functionality documentation.

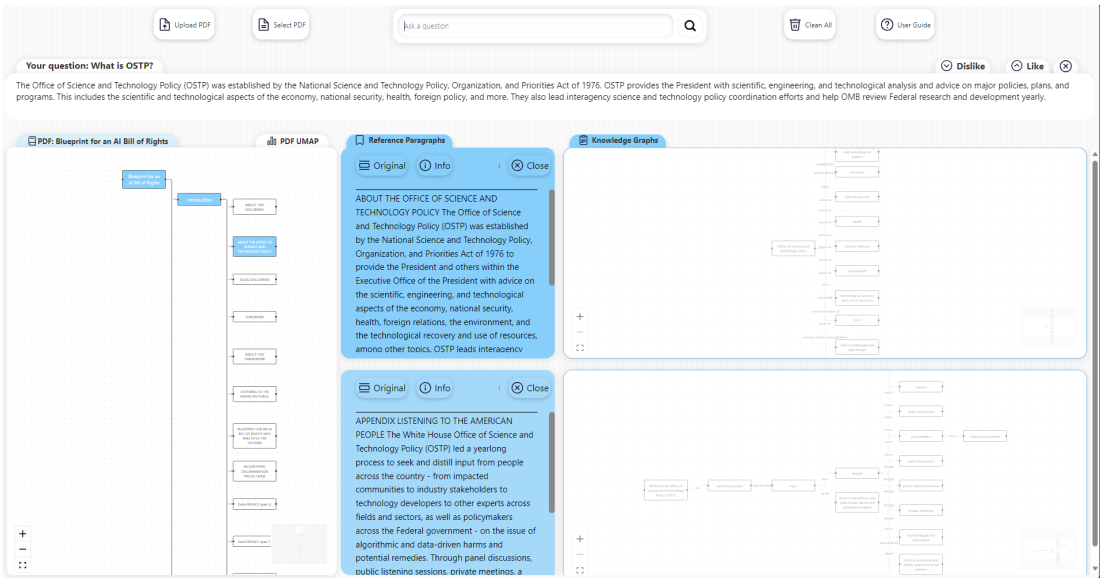

Figure 3.2: System View after Asking a Question.

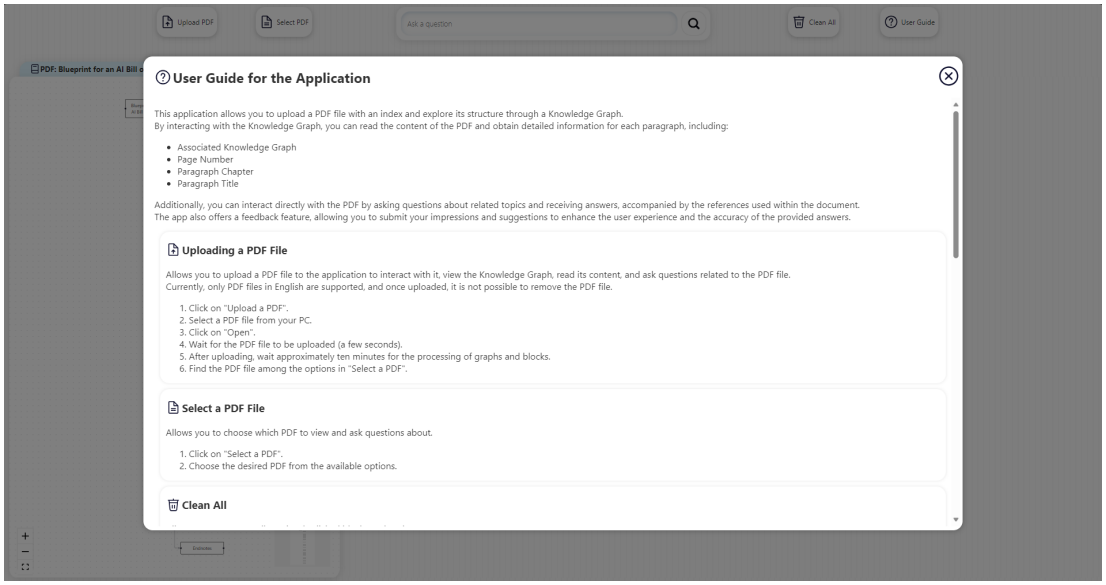

Figure 3.3: System View after Clicking on the Introductory Guide Button

Knowledge Graph Display

On the left side of the interface (Figure 3.4), you will find the knowledge graph representing the structure of the PDF document. This graph can be navigated and expanded to view details of chapters, paragraphs, and images. By clicking on "PDF UMAP," you can access the UMAP (Uniform Manifold Approximation and Projection) view of the document, which provides a compact and informative overview. After asking a question, the system will display the updated UMAP with embeddings of vectors of the nearest similarity responses to the question asked. With the red color, we can see the vector of the question positioned within the UMAP, while with the green color, we can see the vectors of the responses. In the figure (Figure 3.5), we can see the question displayed at the top left and the answer to the question.

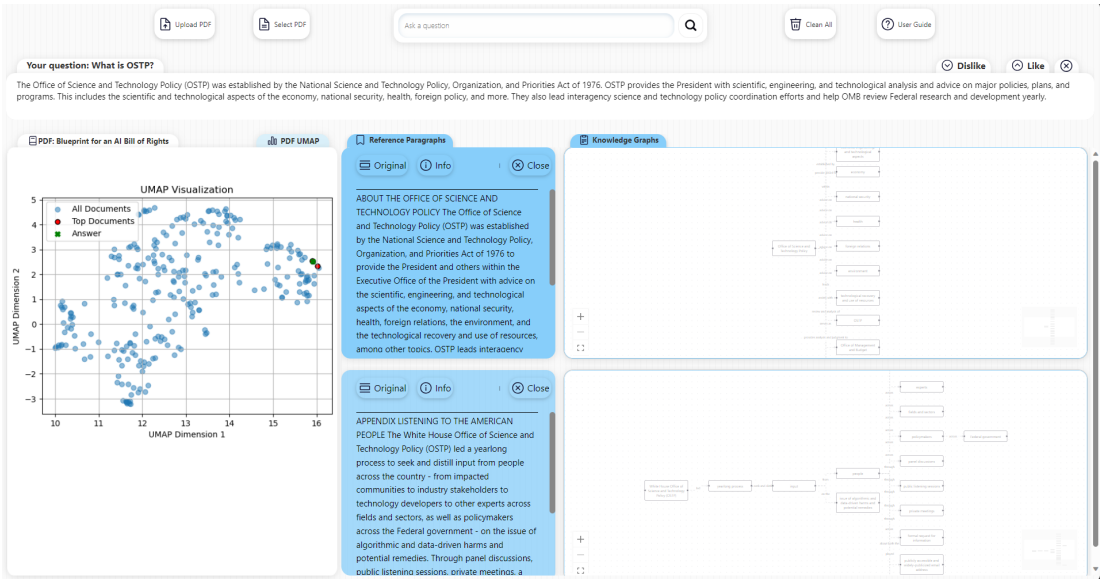

Figure 3.4: The system's display after asking a question and viewing the UMAP.

Figure 3.5: Questions and Answers to the PDF Document

Content Display Area

On the right-hand side, the initially empty area populates in response to user interactions. If a question is asked or if a paragraph or image is clicked, this section displays the corresponding content, including the paragraph with its specific knowledge graph or the selected image. In the case of a paragraph, there are two sections at the top of the corresponding block: "Original," which is the standard display of the text contained in the paragraph, and "Info," where the page number in the PDF document, the title of the chapter, and the corresponding paragraph's title are displayed. To close the individual paragraph or image, you can click on "Close" or click again on the corresponding node. (Figure 3.6).

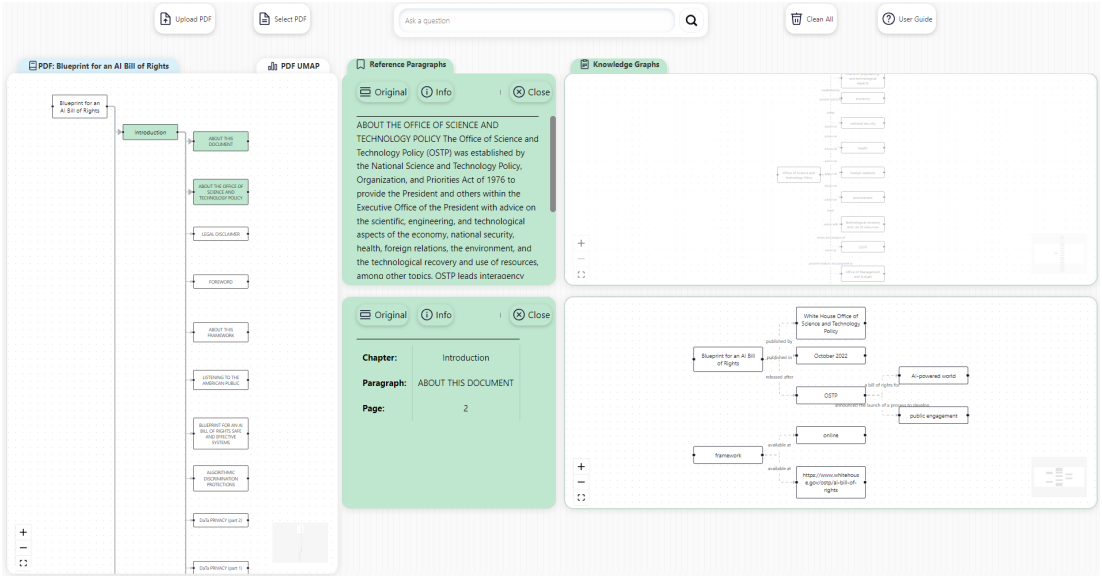

Figure 3.6: Display after selecting two paragraphs, the first in the 'Original' section and the second in the 'Info' section, both with their Knowledge Graphs..

Feedback and Response Improvement

At the Top of the interface, after asking a question and receiving a response, users can view the question they asked and the answer provided by the system (Figure 3.5). Additionally, there is an option to provide feedback on the received response, both positive and negative. In the case of negative feedback, a form is presented, allowing users to provide useful details for enhancing future system responses (Figure 3.7).

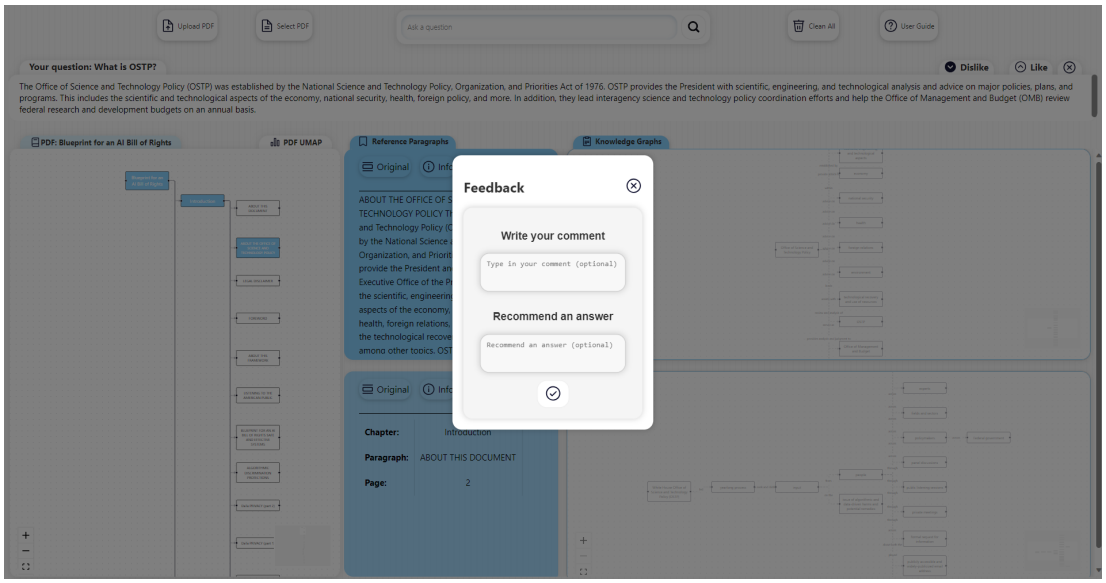

Figure 3.7: The system's display after clicking on 'dislike'.

3.4 Potential use cases

The versatility and competencies of the system are highlighted through the analysis of different contexts of use:

Refinement of Document Management in Business Contexts

In the business context, the system emerges as a fundamental tool for optimizing the management of reports, operational manuals, and technical documentation. Users benefit from the platform to promptly identify essential information, outline connections between fundamental concepts, and obtain immediate answers to specific questions regarding the content of the documents.

Enhancement of Knowledge Management

Functioning as a sophisticated tool for knowledge management, the system enables organizations to classify, review, and consult a wide variety of documents. Such usage transforms cumbersome textual archives into an interactive and easily navigable collection of information, encouraging access and use of accumulated knowledge to facilitate decision-making and stimulate innovation.

Simplification of Access to Legal and Regulatory Information

The system adapts effectively to the examination of legal and regulatory documentation, granting users the ability to navigate agilely through complex sets of legal texts. Legal professionals and consultants are able to swiftly locate specific clauses, interpret legal terminology,

and understand the implications of certain regulations.

Customization of Educational and Training Pathways

Educators and trainers can exploit the system to organize educational materials, stimulate interaction with content, and tailor the learning experience to individual needs. Students have the opportunity to interact directly with the system, investigate related topics, and obtain in-depth explanations, making the learning process a dynamic and engaging conversation.

These contexts highlight the role of the system not only as a technological advancement but also as a catalyst for the expansion of efficiency, the enrichment of knowledge, and the deepening of understanding in various sectors, promoting a more direct and interactive way of accessing information.

CHAPTER 4

Low-Level Contribution

This chapter aims to thoroughly examine the technical infrastructure and internal mechanisms that constitute the core of the system developed for the application. The focus is directed towards a detailed analysis of architectural and technological choices, as well as the analysis of algorithms that enable data processing and user interaction with the application. The fundamental software components will be presented, describing their role within the application ecosystem and justifying the design choices made.

4.1 Conceptual and Concrete Structure of the Application

The architecture of the application developed for the management and analysis of PDF documents is based on an integrated workflow that combines OCR technologies with advanced natural language processing and visual user interaction. Below, the end-to-end process and key components are described

Uploading the PDF Document: The user begins by uploading a PDF document through the application's user interface; the PDF must contain a well-structured Table of Contents.

Extraction and Conversion of Images: The system reads the PDF file and utilizes a conversion module to transform the pages into PNG images, preparing them for OCR processing. Additionally, any images contained within the pages of the PDF file are also saved as PNG files.

Identification of the Table of Contents: In parallel, the system extracts the text from the PDF and identifies the document's Table of Contents using techniques such as vectorization,

cosine similarity, and the BERT model.

Storage: The PDF document and the extracted images from the pages are stored in the storage section of the Firebase database [19] for quick and secure access.

Optical Character Recognition (OCR): The images undergo OCR to extract text, utilizing advanced algorithms to ensure accuracy.

Bounding Box-based Chunking: The text is divided into logical segments based on bounding boxes detected during OCR for effective structuring. To address the challenge of handling sentences that span multiple pages, chunking techniques were applied, adapting the text to the detected blocks and ensuring content continuity. Additionally, SpaCy was used for further sentence segmentation within each block, enhancing the structure and accessibility of the extracted text.

Table of Contents Identification: OpenAI's ChatGPT-4-Vision processes the PNG files of the table of contents and, through a specific prompt, generates a JSON representation of the table of contents. This process provides a detailed mapping of pages corresponding to individual chapters, facilitating the segmentation of text into distinct paragraphs.

Paragraph Headings with GPT-4-Turbo: To further enrich the structure, ChatGPT-4-Turbo is used to assign relevant headings to individual paragraphs within the document, enhancing the readability and accessibility of the content.

Knowledge Graph with GPT-3.5-Turbo: A detailed Knowledge Graph is constructed for each paragraph, identifying relationships and key concepts, which are then represented in JSON [43] format. It takes a specific prompt as input, containing the paragraph text, chapter name, PDF name, request with output examples, and the system's role. As output, it provides a JSON response containing nodes and edges.

JSON Serialization: The processed data is converted into JSON, standardizing the format for subsequent storage and retrieval.

Database: The data contained in the structured JSON is stored in MongoDB [85], which provides a flexible query interface for data retrieval.

Flask Middleware: Flask [97] acts as a bridge between the back end and front end, managing the data transmission logic.

User Interaction: The user interface, built with React [78], enables an interactive visualization of the Knowledge Graph, enhancing the user experience. Additionally, users can ask questions and receive answers through the interface.

Retrieval-Assisted Generation (RAG): This technology is employed to answer specific questions by providing context from the document. It runs in parallel with the application, allowing all necessary steps to be completed and uploading the UMAP images generated from embeddings to Firebase storage.

User Feedback: The system collects anonymous feedback through Firestore to guide future updates and improvements of the application.

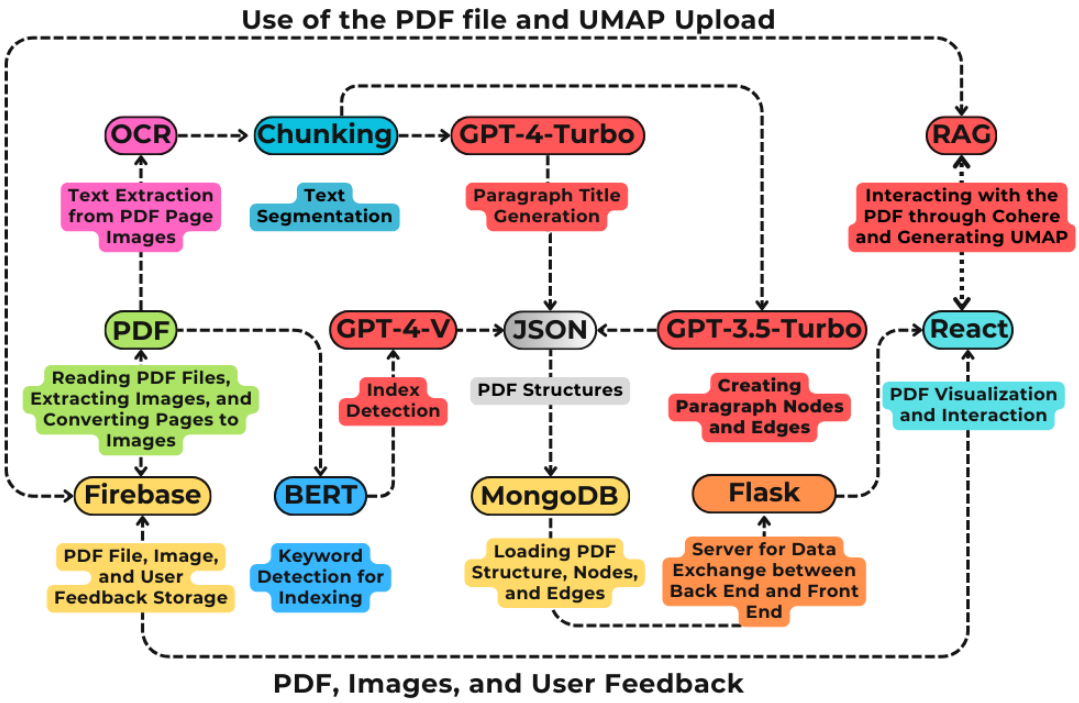

Figure 4.1: The diagram presents a workflow for processing PDF files, extracting information, and interacting with the data.

Through this integrated architecture, the system not only enables a detailed analysis of PDF documents but also provides an intuitive interface for users to interact with the extracted data, significantly enhancing the accessibility and utility of the information contained within PDF (Figure 4.1).

4.2 Key Technologies Used

The development of the application required the integration of several advanced technologies, each chosen for its specific features and performance within the workflow context. Below, I provide a summary of the main technologies employed and their contributions to the project.

Front-end

I chose React to develop the front end of the application due to its ability to build rich and interactive user interfaces with a component-based approach. Using React simplified the management of the application's state and the responsiveness of the user interface, resulting in a smooth and engaging user experience (Figure 4.1).

React Libraries and Plugins

The application utilizes a wide range of React libraries, including:

axios [124] is used to handle HTTP requests, simplifying the interaction with the Flask backend and the Firebase database.

@tanstack/react-query [70]: Implemented to manage server state in React, optimizing performance through intelligent request caching and state synchronization.

framer-motion [101]: Chosen to animate React components with advanced animations, enhancing the user experience with intuitive visual feedback.

React Flow [62]: Integrated to visualize and manage the knowledge graph, allowing users to interact with a graphical representation of the extracted data. Initially, I used the free version of React Flow, but later opted for the paid version to access advanced graphical features that proved crucial for my project. React Flow works with data in JSON format to generate a graphical representation of nodes and edges, making it easier to visualize relationships and connections within the graph.

Lordicon [71]: Used to include animated and interactive icons, increasing the visual appeal of the interface.

react-hot-toast [69]: Employed to provide immediate notifications and feedback to the user, enriching the interaction with clear and timely messages.

Firebase SDK: Integrated for user authentication, real-time database, and image/document

storage, establishing seamless communication between the front end and back end.

dagre [87]: Adopted for the automatic positioning of nodes in the knowledge graph, ensuring a clear and organized visualization.

react-dom/client e ./reportWebVitals [75]: Used for rendering the application in the DOM and monitoring website performance metrics.

Thanks to this collection of technologies, the front end not only reflects the modernity and scalability required by the project's requirements but also provides a level of interactivity and usability that elevates the overall user experience.

Back-end

The back end of the application was developed in Python [44], a choice motivated by its versatility, the rich offering of third-party libraries, and the ease of integration with machine learning systems. Below, I list and describe the main Python libraries that played a crucial role in the development of the back end.

Flask: Flask (Figure 4.1) was chosen for its simplicity and flexibility as a web micro-framework. In particular, the request functions were utilized to handle HTTP requests, and jsonify was employed to send JSON responses to clients. Flask-CORS [106] was integrated to manage Cross-Origin Resource Sharing (CORS), enabling the back-end to securely communicate with the front-end hosted on a different domain.

Flask is a lightweight and powerful framework for web application development in Python, following the Web Server Gateway Interface (WSGI) software design pattern. This pattern is essential for facilitating interaction between web applications and web servers.

The Flask server I implemented works by listening to HTTP requests sent by the Front End. These requests are routed to the appropriate routes defined in the server. The server then performs the necessary logic, processes the requested data, and sends it back to the client as an HTTP response.

One of the major advantages of Flask is its intuitive API, which simplifies the process of defining routes and handling requests. Additionally, its flexible development environment allowed me to easily tailor the server to the specific needs of my project.

Data processing and analysis libraries: I used re for regular expressions [45], uuid [47] for generating unique identifiers, json for JSON data manipulation, and typing [46] for annotating types in functions, improving code readability and maintainability.

Environment variables and interaction with the operating system: The dotenv [63] library allowed for safe loading of environment variables, and os enabled interaction with the operating system.

Database: pyrebase [53] was used to interface with Firebase for storage operations, and pymongo facilitated interaction with the MongoDB database (Figure 4.1).

Machine Learning and NLP: I used torch [102] and numpy [90] for numerical computations and for running deep learning models. The transformers [35] libraries provided pre-trained models like BertTokenizer and BertModel [26] for Natural Language Processing (NLP) 4.1.

Logging and document management: logging was used for event tracking and troubleshooting, while fitz (part of PyMuPDF) [66] was used for PDF document processing (Figure 4.1).

Integration with OpenAI: The openai library was used to integrate OpenAI's APIs, allowing the system to utilize advanced natural language processing models. I used models like GPT-4-Vision, GPT-4-Turbo, and GPT-3.5-Turbo. These models were used to interpret and structure extracted text, generate titles, and construct components of a knowledge graph. Additionally, tiktoken [59] was used for counting tokens in input and output (Figure 4.1).

Text processing libraries: shutil [42] and io [41] are used for file management. base64 [40] is employed for encoding and decoding binary data into strings.

Image processing libraries: cv2 (OpenCV, Open Source Computer Vision Library) is a widely-used open-source library for image processing and computer vision, used in this case to read an image from a file. pytesseract [65] is used for optical character recognition (OCR), as I employed an open-source OCR software called Tesseract OCR [118] to convert images and scanned documents into editable text.

Data Analysis and Visualization: For dimensionality reduction and visualization, I used umap and matplotlib [113], while spacy was employed for advanced text preprocessing.

Specific Libraries: ast [39] was used for converting strings into executable Python code, and hnswlib for building proximity graphs for fast information retrieval.

RAG (Retrieval-Augmented Generation): The 'cohere' [15] library allows communication with Cohere's APIs for RAG (Retrieval-Augmented Generation). This choice enabled the uti-

lization of a highly advanced search and response generation mechanism. The process begins by uploading the PDF document to Cohere, where the text is embedded, creating a vector representation that captures the semantic meaning of the content. Subsequently, document indexing is performed, organizing the data for efficient searching.

When a user asks a question, the system employs the RAG model to retrieve relevant documents based on the pre-processed embeddings. This step allows for the identification and retrieval of the most relevant portions of text from within the extensive dataset. Finally, a response is generated that not only relies on the context of the question but is also enriched with specific information extracted from the documents. This approach ensures that the provided answers are not only accurate but also deeply rooted in the context of the original PDF document, greatly enhancing the user's information experience.

The choice of Cohere as a partner for implementing RAG technology was driven by the accuracy of their embedding system and their powerful capabilities in generating contextualized text. The integration of their APIs proved to be a critical asset for the project, providing users with immediate access to detailed and specific information, thereby facilitating a better understanding of the document's content.

This set of libraries formed the technological foundation of our back-end, ensuring that the system is scalable, maintainable, and performant. Each library was carefully chosen for its specific functionalities, enabling me to build a robust back-end architecture that supports the complex operations required by the project.

Database

At the core of the application, I chose to use two distinct but complementary database systems, Firebase and MongoDB, to manage different aspects of the application's data.

Firebase

Firebase, a web and mobile application development platform managed by Google, was chosen for its real-time data management capabilities and serverless architecture that facilitates rapid development. In the application, Firebase Storage was used to store and serve PDF documents and processed images. Its high availability and data durability ensure that files are always accessible when needed. Additionally, Firebase Firestore, a document-based NoSQL database, was adopted to collect and store user feedback.

MongoDB

MongoDB was selected as the primary NoSQL database due to its excellent flexibility in han-

dling JSON documents, which perfectly suits the dynamic nature of the data manipulated by the application. Its distinctive features, such as support for complex ad-hoc queries, efficient indexing, and the ability to handle high workloads, made it the ideal choice for operations that require fast reads and writes and efficient transaction management. MongoDB hosts the various structured knowledge graphs.

With the combination of Firebase and MongoDB, a robust and scalable database ecosystem was created that supports both static storage operations and the complex requirements of dynamic data processing. This hybrid architecture allowed me to leverage the strengths of both systems to provide a comprehensive and efficient data management solution.

4.3 System Design Iterations

Initial Experiments on Creating a Knowledge Graph

Initially, I embarked on a series of practical experiments using the Python programming language and the Pyvis visualization library [100], with the goal of exploring and understanding the mechanisms behind creating a Knowledge Graph.

To implement the creation of nodes and edges, I considered using a Language Model (LM). My initial choice was the ChatGPT-3.5-turbo model [95], known for its advanced natural language processing capabilities.

I designed a specific prompt that, when provided with input text, initiated the process of generating nodes and edges for the Knowledge Graph. The ChatGPT-3.5-turbo model was used to interpret the provided text and identify semantic relationships, thus creating the corresponding nodes and edges.

Subsequently, the results obtained from the model were processed and passed as input to the Pyvis library. Pyvis, a powerful network visualization library, allowed for a clear and interactive representation of the created Knowledge Graph.

During this experimental phase, I conducted numerous tests to evaluate the effectiveness of the Knowledge Graph creation process and optimize the involved parameters. The selection of the LLM model, the design of the prompt, and the interpretation of the results were key points of reflection and adaptation.

Input from PDF Files

Subsequently, I continued with the goal of taking input from a PDF document containing text. This phase of the process required a series of experiments and tests with different Python libraries available.

Initially, I used the pdfplumber library [56] to read the PDF file and extract text from the

pages. However, during the course of testing, I explored several alternatives such as PyPDF2 [37] and pdfminer [107], which proved to be more effective in certain aspects. Despite these improvements, I later encountered the need to perform text "chunking," which involves breaking the text into more manageable blocks. To address this specific phase, I decided to change libraries once again and adopted PyMuPDF.

This iterative process of exploring and adapting libraries played a key role in refining the approach to handling PDF files and helped establish a more robust methodology for processing the extracted text.

Vector Representation

After acquiring the text from the input PDF file, I deemed it necessary to enable document-wide searching by transforming sentences into vectors.

To implement this transformation, I explored various Python libraries, including Word2Vec [104], FastText [76], and more. While using these libraries, I proceeded to create specific models and vocabularies for vector generation.

Obtaining a vector representation of sentences allowed me to facilitate text manipulation and analysis. Additionally, each tested library required defining specific parameters and creating models suitable for the context of the PDF document.

Vector Database

At this point, I needed to load the vectors into a specialized database. After exploring various options, I chose Pinecone [111], a vector database that proved to be suitable for the requirements at that time.

Search and Metadata Association

In the search phase, the primary goal was to implement a system to efficiently retrieve relevant vectors from the vector database. To achieve this, I devised a method to associate vectors with their respective metadata. This approach involved matching each vector with a corresponding sentence based on textual content. Consequently, when performing a search query, the system could return the associated sentence, selected based on the relevance of the words it contained.

This strategy became necessary due to an issue that arose in the conversion of textual data into vectors and vice versa. During this process, I observed that the resulting vectors varied significantly, making it challenging to obtain accurate search results through sentence encoding and decoding methods.

This obstacle led me to reconsider and modify my approach, ultimately allowing me to iden-

tify and correct the underlying error, avoiding the use of vectors altogether.

Application of Named Entity Recognition (NER)

In the advanced phase of the project, I focused on improving the quality and accuracy of nodes, edges, and their relationships in the knowledge graph, which until that point had primarily relied on sentences extracted from PDFs. A fundamental technique in the field of NLP for identifying significant entities in text is NER.

To implement NER, I chose to use SpaCy, a widely recognized Python library for effective natural language processing. I applied NER to the sentences extracted from the PDF documents, experimenting with various SpaCy models, all in the English language. I started with the smallest model ("en-core-web-sm") [23], explored intermediate options ("md" and "lg"), and finally tested the most advanced model ("en-core-web-trf").

Simultaneously, I evaluated other approaches for NER and tokenization, including the use of a Hugging Face model called Babelscape/rebel-large [25]. However, due to limitations related to computational power, I decided to proceed with the SpaCy-based implementation.

Later, I discovered that by properly optimizing the prompt for ChatGPT, it was possible to achieve superior results without the use of NER. This realization led to a revision of the initial approach, removing the use of NER and demonstrating the importance of accurate prompt configuration in the context of natural language processing.

Implementation of Chunking

During the development of the project, I identified the need to implement a chunking technique. This process was crucial for managing the size of text extracted from PDFs: I needed to work with text portions larger than a single sentence but significantly smaller than the entire document.

Initially, I opted for a page-based chunking approach, meaning I extracted the text of one entire page at a time. However, this method raised several issues, with the most relevant ones being:

1. Handling sentences that spanned multiple pages.
2. Distinguishing between various chapters within the document.

To address the first problem, I implemented a solution that involved concatenating the initial sentences of the next page with the text of the current page if the last sentence of the current page did not end with a period. This approach allowed for a more coherent and complete reading of the text.

The second problem, however, proved to be more complex. After several attempts and tests, I concluded that the page-based chunking approach was not sufficient to effectively distinguish between the various chapters of the document. This realization led me to rethink and modify my chunking strategy.

Extraction and Handling of Images from PDF Documents

In the image saving process, I explored various options to optimize the quality and usability of the extracted images. One of the tested solutions was converting images to the SVG format, a vector format that promised high quality and scalability. I experimented with various Python libraries, including Potrace, which, however, limited the images to grayscale, a compromise that did not fully meet the project's needs.

Despite the existence of paid solutions offering advanced image conversion capabilities, I preferred to stick to open-source tools. After various trials and evaluations, I decided to adopt a more direct and reliable approach: saving images in high-quality PNG format using PyMuPDF directly. This choice ensured a balance between image quality, fidelity to the original document, and ease of management, making the PNG format the ideal solution for the project's requirements.

Improvement of Similarity Search with SBERT

During the project, I focused on optimizing similarity search. The initial system, based on vectorization with FastText or Word2Vec, had some limitations, especially in handling typographical errors or small variations in search words, which compromised the ability to retrieve relevant sentences as the logic was implemented.

To overcome these challenges, I decided to implement SBERT (Sentence-BERT), a variant of the BERT model optimized for embedding entire sentences. The use of SBERT marked a significant improvement in similarity search. Unlike previous methods, SBERT offers a richer and more detailed semantic representation of sentences, enabling more robust and variation-tolerant search.

In particular, I used SBERT to generate vector embeddings of sentences, which were then compared using cosine similarity to identify the most relevant sentences in response to a search query. This approach allowed for retrieving sentences with the highest similarity scores, greatly improving the accuracy and reliability of search results.

Despite the improvements achieved with SBERT, I continued to explore further enhancements and ultimately opted for the integration of a chatbot. This solution led to another leap in quality, providing even more detailed and contextualized responses to user queries, transforming the search experience into an interactive and informative dialogue.

Integration of MongoDB for Embedding Management

With the introduction of SBERT into the project, I was able to generate high-quality vector embeddings for sentences extracted from PDF documents. The next step was to find an efficient storage system for this data. Despite the initial use of Pinecone, I recognized the need for a solution that better suited the project's requirements and was more intuitive to manage. Consequently, I opted for the integration of MongoDB, a database known for its flexibility and ease of use.

The decision to switch to MongoDB was motivated by several factors. First, MongoDB offered me a more straightforward and accessible structure for storing and managing sentence embeddings alongside their corresponding texts. Additionally, my familiarity with MongoDB and its widespread adoption in the industry assured me that I could make the most of its features.

Subsequently, due to the integration of the chatbot, I stopped saving the embeddings, and therefore MongoDB remained the most effective choice.

Exploration and Evaluation of Open Source LLM Models

In pursuit of the goal to further optimize the project, I embarked on an extensive phase of research and experimentation with various open-source Large Language Models (LLMs), aiming to assess alternatives to ChatGPT. This process involved testing a wide range of models, each with its own specific features and capabilities. Some of the explored models include:

- **meta-llama/LLaMA [77]:** LLaMA is an open-source LLM model that has been developed for research purposes and NLP applications. It is designed to be modular and customizable for a variety of tasks.
- **princeton-nlp/Sheared-LLaMA-2.7B [34]:** This is a variant of LLaMA developed by the Princeton research group. It is trained on a large corpus of texts and offers natural language generation capabilities.
- **mistralai/Mistral [3]:** Mistral is a GPT-3-based LLM model trained on a wide range of multilingual data. It is known for its natural language generation performance and can be used for various NLP applications.
- **Falcon-7B-Instruct [29] and Falcon-40B-Instruct [28]:** These are LLM models developed for machine learning and language understanding purposes. They have been trained on a large dataset of instructions, thus specializing in interpreting and generating instructions.

- **HuggingFaceH4/zephyr-7b-alpha [36]:** An LLM model developed by Hugging Face, known for the community support and wide range of pre-trained models offered. This specific variant is designed for text generation applications.
- **bigscience/bloom [27]:** An LLM model developed as part of the BigScience project, aiming to create large-scale LLM models for scientific research purposes. This model is known for its size and computational power.
- **GPT and variants [93]:** These models are part of OpenAI's Generative Pre-trained Transformer (GPT) series, known for their ability to generate high-quality natural language. They are used in various applications including chatbots, automatic translation, and text generation.
- **Guanaco [32]:** An open-source LLM model that can be specialized for specific applications. Its adaptability and customization make it useful for a wide variety of tasks.
- **adept/fuyu-8b [30] and adept/persimmon-8b-base [33]:** These are variants of LLM models trained for specific tasks or in particular domains. For example, they might be specialized in medicine or law.

To facilitate the use of certain models like LLaMA and Mistral, I leveraged Gradio [31], a Python library developed by Hugging Face, which enables the use of models hosted on Hugging Face without the need to run them locally. Other models were tested using Google Colab, taking advantage of the increased computational power it offers, although I encountered limitations even with the Google Colab Pro plan [50].

The models were evaluated based on their effectiveness in specific tasks such as node, edge, and relationship generation, text chunking, and text segmentation into chapters. This testing phase provided valuable insights into the capabilities and limitations of each model.

After careful evaluation and numerous tests, I concluded that, despite the diversity and potential of the explored models, the most suitable solution for the needs of my project was to continue using ChatGPT. This decision was driven by a combination of factors, including the quality of results, ease of integration, and alignment with the existing workflow.

Exploration of Solutions for Document Structure Generation

One of the critical challenges in my project was to identify an effective method to recognize and reproduce the complex structure of PDF files. The central question was: how can I accurately discern and replicate the organizational structure of a PDF document?

In the search for a solution, I came across a recently published paper that seemed to offer the perfect answer. The paper described a Python program called Document Structure Generator

(DSG) [103], specifically designed to analyze and reconstruct document structures through OCR. Intrigued by the potential of DSG, I conducted a series of tests and implementations, hoping to integrate this technology into my project.

However, despite my efforts and direct communication with the DSG development team, I had to face an unexpected reality: the program relied on proprietary software to which I had no access, making its integration into my workflow impossible.

Faced with this obstacle, I explored alternatives by testing various libraries available on GitHub [49], including Donut [60], LayoutLMv3 [81], PaddleOCR [121], and Deepdoctection [80]. Despite the wide variety of options, the results obtained did not meet expectations. The tested solutions were unable to accurately capture the complexity and variety of document structures I intended to analyze.

In light of these challenges, I had to adopt a more creative and independent approach. Through a process of experimentation and adaptation, I managed to develop a customized solution that, while not perfect, allowed me to progress in the project by providing a functional representation of the PDF document structure.

Exploration of Clustering and Text Analysis Techniques

In an attempt to organize the text extracted from PDF documents into coherent chapters based on context, I explored the use of clustering, a data analysis technique that aims to group similar elements based on specific features or metrics. The goal of clustering is to discover hidden structures in the data by organizing elements so that those within the same group are more similar to each other than those in different groups.

For this phase of the project, I used a dendrogram, a graphical tree-like representation that illustrates hierarchical relationships and similarities among elements. The dendrogram proved particularly useful in visualizing how various text segments clustered together, providing an intuitive view of the hierarchical structure of the data.

In the text analysis process, I adopted various techniques to further refine the understanding and organization of the content:

Context Checking: I examined the context in which words or phrases were used to ensure their meaning was interpreted correctly, recognizing that words can have different meanings depending on the context.

Part of Speech (POS) Tagging: I used POS tagging to identify the grammatical category of words within sentences, helping to understand the grammatical role and function of each word in the context of the sentence.

TF-IDF Analysis: I experimented with TF-IDF analysis to assess the importance of words within documents by comparing the frequency of a word in a document with its inverse frequency in the corpus of documents.

Summarization: I explored summarization techniques to condense the text, extracting the most relevant information and reducing the content to a more concise format without losing essential meaning.

Context Analysis: I conducted context analysis to better understand the circumstances and factors surrounding specific events, situations, or communications within the text.

Despite the in-depth exploration of these techniques, I realized that implementing such a complex clustering and text analysis system was not strictly necessary for the immediate goals of the project. Furthermore, creating such a system would have required a significant investment of time and resources. Therefore, I decided to temporarily set aside these ideas, opting to focus on more direct and manageable alternatives while retaining the knowledge gained for future explorations.

CHAPTER 5

Evaluation

The evaluation of the system is crucial to demonstrate its effectiveness and efficiency. In this chapter, I outline the methodologies and results of the evaluation procedures, dividing the analysis into two main categories: efficiency and effectiveness.

5.1 User Interaction through the Graphical Interface

User interaction with the system was evaluated through a series of usability tests conducted on a sample of testers representative of the application's end users. The testers, with various specializations in the scientific field and familiarity with everyday computer use, performed a series of tasks designed to assess the intuitiveness of the interface, the clarity of presented information, and the system's responsiveness to requests.

Tester Profile
The group of testers consisted of 11 professionals from the IT sector, both male and female, with ages ranging from 25 to 35 years, all with a high level of academic education and significant experience in using computer technologies. This profile ensured that the gathered feedback was informative and representative of the needs of advanced users. None of them had previously used this system or similar systems.

Tasks and Interaction Paths
The testers followed a series of predefined interaction paths (happy paths) to perform specific tasks, including:

- Querying the system using natural language questions.

- Using feedback features to evaluate the system's responses.
- Exploring various data visualization modes, such as the knowledge graph and UMAP.
- Cleaning the interface and managing PDF documents.

The tests were conducted on the PDF named "Blueprint for an AI Bill of Rights." The Blueprint for an AI Bill of Rights is an initiative proposed by the United States administration, under the guidance of the Office of Science and Technology Policy (OSTP), with the aim of guiding and structuring the development and deployment of artificial intelligence (AI) in a manner that respects the fundamental rights of citizens. This document does not have the force of law but serves as a recommendation for public and private organizations developing, acquiring, or using AI systems. You will be asked to find a specific paragraph within the text in order to understand and evaluate the difficulty in the search. The various tasks the testers followed were as follows:

- Ask when the "Blueprint for an AI Bill of Rights" [54] was published and ensure the answer is correct by reading the relevant paragraphs, then clean the screen.
- Ask what "OSTP" is and leave positive feedback for the system's response, then clean the screen.
- Find out which page the paragraph "PRIVACY-PRESERVING SECURITY" is on and leave negative feedback for the system's response, adding a comment, then clean the screen.
- View the UMAP of the PDF.

The happy paths for the various tasks were as follows:

- Click on "Ask a question," write the question, read the response, read the relevant paragraphs provided, click on "Clean All."
- Click on "Ask a question," write the question, read the response, click on "Like," click on "Clean All."
- Click on "Ask a question," write the question, read the response, click on "Info" to find the paragraph and page, click on "Dislike," write a comment, click on "Clean All."
- Click on "PDF UMAP."

Tester Observations and Feedback

During the tests, valuable observations were collected, highlighting both strengths and areas for improvement:

First Task: Testers took varying amounts of time to familiarize themselves with the interface, highlighting the importance of an introductory guide or tutorial feature to facilitate initial user orientation. The first task was completed with a minimum time of 1.28 minutes and a maximum time of 5.29 minutes. Most testers followed the happy path, while one person initially interacted with the knowledge graph of the PDF structure.

Second Task: In the second task, the minimum time was 54 seconds, while the maximum time was 2.43 minutes. Many testers quickly found the solution in this task, as they had become familiar with the application by then.

Third Task: In the third task, the minimum time was 2.10 minutes, while the maximum time was 5.29 minutes. Only a few testers clicked on the paragraph info, while many tried to find the information in the knowledge graphs, had difficulty scrolling through the paragraphs, and attempted to click on various buttons. Many asked the system on which page the paragraph was located but did not receive the correct answer because it was not a predefined action, making it one of the most challenging tasks to tackle.

Fourth Task: In the fourth task, the minimum time was 7 seconds, while the maximum time was 5 minutes. The substantial difference lies in the fact that several testers did not know what a UMAP was. The variation in the time taken to understand the UMAP highlighted how the inclusion of advanced concepts requires a clear and accessible explanation, ensuring that all users can benefit from the offered features.

User Satisfaction Assessment: SUS Results

During the evaluation, I adopted the System Usability Scale (SUS), a standardized tool for measuring user satisfaction with interactive systems. The SUS provides a comprehensive assessment of usability and the overall user experience through a series of questions covering various aspects of the system, from ease of learning to overall satisfaction.

The SUS scores obtained ranged from a minimum of 62 to a maximum of 88, indicating a generally good level of acceptability with room for improvement. This range of scores reflects the diverse experiences and perceptions of users, underscoring the importance of considering both positive aspects and those that require further optimization.

Analysis of SUS Scores and Implications

Scores Above 80: Scores in this range suggest a high level of user satisfaction, indicating that the system is intuitive, efficient, and enjoyable to use. These positive ratings are an en-

couraging signal and indicate that many aspects of the system have been well-designed and effectively meet user needs.

Scores Between 70 and 80: These scores indicate a moderate level of satisfaction. Users in this range have generally had a positive experience but may have encountered some obstacles or limitations. These scores require a more in-depth analysis to identify specific areas that need improvement or clarification.

Scores Below 70: Scores in this range suggest that there are significant aspects of the system that do not satisfy users or hinder their interaction. It is essential to investigate the causes of such ratings, which could result from usability issues, missing functionality, or a steeper learning curve.

Completion Times and Usability

The total time to complete all tasks ranges from a minimum of 6 minutes to a maximum of 17 minutes. The range of time taken to complete the tasks has shown that while the interface is generally intuitive and responsive, there are opportunities to further optimize the user experience, making the system even more accessible and user-friendly.

Implications for Improvement

Usability tests have provided valuable insights into user interaction with the system, highlighting the need to improve initial orientation, navigation within the interface, and the clarity of presented information. Most of the collected feedback has already been used to enhance the system, making the interface even more user-friendly, informative, and in line with user expectations and needs.

5.2 Efficiency

The efficiency of the system was assessed by considering both resource consumption (such as processing time and memory usage) and the system's ability to handle a high volume of requests or data. The test structure for efficiency evaluation includes various aspects such as the testing environment, tasks and workloads, and quantitative metrics.

Testing Environment

The tests were conducted on two different platforms to ensure that the system is robust and

performs well in various operating environments. In the case of Windows, the tests were conducted within virtual environments managed through Anaconda [6], ensuring dependency consistency and environment isolation. Below are the specifications of the machines and development environment used:

Anaconda Virtual Environment

For the Windows 11 (version 23H2) platform, the tests were conducted within virtual environments created and managed through Anaconda. This allowed:

- Maintaining an isolated and consistent development and testing environment, preventing conflicts between dependencies and ensuring result reproducibility.
- Easily managing package versions and dependencies, facilitating the installation and configuration of libraries required for the system.
- Ensuring that the system's performance was not influenced by variations in the execution environment, providing a stable and controlled evaluation context.

Windows Platform

The test was performed on an MSI (Katana B12V) machine within the Anaconda virtual environment, with the following specifications:

- **Processor:** 12th Gen Intel(R) Core(TM) i7-12650H @ 2.30 GHz [16]
- **GPU:** NVIDIA GeForce RTX 4070 Laptop GPU [92]
- **Driver GPU:** Driver NVIDIA Studio 546.33 [91]
- **RAM:** 16,0 GB (15,7 GB utilizzabile)
- **Operating System:** Windows 11 Home, Version 23H2, Operating System Build 22631.3007
- **Other Specifications:** 64-bit operating system on x64-based processor.

Mac Platform

similarly, the tests were also conducted on a MacBook Air (M2, 2022) [9], providing a comparison between different ecosystems and hardware. The specifications of the Mac machine used for testing include:

- **Chip:** Apple M2 [119]
- **RAM:** 8 GB RAM
- **Operating System:** macOS Sonoma Version 14.2.1 [10]

This hardware and software configuration provided a stable and powerful platform for conducting tests on both Windows and Mac, ensuring that the system's performance evaluation reflected its actual capabilities in different operating environments. The inclusion of both platforms in the testing process allowed for assessing the system's portability and compatibility across multiple operating systems, ensuring a reliable and versatile application.

Tasks and Workloads

The tasks and workloads were designed to simulate realistic usage scenarios and evaluate the system's ability to handle various types of requests and data volumes. Details about specific tasks, such as processing and analyzing PDF documents, extracting information, generating responses, as well as real-time user interaction through the graphical interface, were carefully monitored to assess the system's efficiency.

Processing and Analysis of PDF Documents

The phase of processing and analysis of PDF documents was carefully evaluated, taking into account various factors that can influence the system's performance. Documents were selected to represent a variety of use cases, including diversity in terms of length, structure, and visual content. The factors considered in the evaluation process include:

Document Size and Structure: The analysis took into account the number of pages, chapters, words, and images, evaluating how these characteristics influence processing time and the available storage space required.

System Requirements: The need for storage space for temporary files, such as images extracted from documents, was monitored, highlighting the importance of adequate storage capacity.

Network Connectivity: The stability and speed of the internet connection were recognized as critical factors, especially considering interaction with external services like OpenAI for natural language processing.

Hardware Reliability: The importance of reliable and high-performance hardware was emphasized, recognizing that computer processing capability significantly affects execution time, even though many tasks are delegated to external services like OpenAI.

Information Extraction and Response Generation The phase of information extraction and response generation was evaluated with a focus on system responsiveness and the ability

to handle dynamic requests. Key considerations in this phase include:

Request Handling: The system was tested with sequential requests, assessing its ability to process individual requests efficiently and provide timely responses.

Network Dependencies: The impact of connection speed on response latency was examined, recognizing the importance of stable connectivity for real-time interactions.

External Processing: The significance of the external service Cohere for managing computational complexity was acknowledged, highlighting the system's efficiency in delegating complex tasks to external resources, thus reducing the computational load on the user's device.

Quantitative Metrics

The system's performance has been quantified using precise metrics such as response time and memory usage, providing an objective measurement of the system's efficiency in various test scenarios.

Response Time

- **Processing and Analysis of PDF Documents:**
  - *Scenario:* Testing with a 73-page PDF with a size of 11.1 MB.
  - *Results:* The execution time showed significant variability, with a minimum of 1 hour and 30 minutes and a maximum of 3 hours. This variability can be attributed to factors such as content complexity, document size, network congestion, and device processing capacity.
- **Information Extraction, Response Generation, and Real-time Interaction:**
  - *Scenario:* Requests for extraction and response, and user interactions through the graphical interface.
  - *Results:* Overall, the system was able to provide responses and display reference paragraphs in less than 20 seconds, demonstrating high responsiveness and efficient handling of dynamic requests.

Memory Usage

- **Processing and Analysis of PDF Documents:**

  - *Observations:* The system's performance was found to be sensitive to the availability of RAM. In the case of the MacBook, the system optimally utilized the processor, with minimal use of RAM. Conversely, on the Windows PC, almost full utilization of RAM was observed, reflecting different resource management strategies.

- **Information Extraction, Response Generation, and Real-Time Interaction:**

  - *Observations:* The use of RAM in these phases was minimal and primarily related to the operation of the browser. This indicates that most of the computational load for these operations is managed externally, alleviating pressure on local system resources.

This section has provided a detailed evaluation of the system's efficiency, based on concrete tests and quantitative measurements. The analysis highlighted not only the system's performance in terms of speed and resource consumption, but also its ability to handle complex tasks and high workloads, thus underlining the project's success in terms of operational efficiency.

5.3 Effectiveness

The effectiveness of the system was evaluated qualitatively, based on how it has improved or increased the functionalities, usability, flexibility, and ease of maintenance.

Increase in Functionalities

The new functionalities provided by the system represent a significant advancement in the management and analysis of PDF documents, bringing tangible improvements in the user experience, the quality of the results obtained, and the overall ability of the system to effectively meet the project's requirements.

Intuitive Interaction through Natural Language

The ability to interact with PDF documents using natural language is a revolutionary step towards a more intuitive and accessible user experience. Users can now:

Ask Questions Naturally: Instead of manually navigating through pages of documents, users can simply formulate questions in natural language and receive contextualized and precise answers.

Receive Contextualized Responses: The system not only provides answers but also indicates the specific location of the information within the document, significantly improving efficiency in searching and knowledge acquisition.

Structured Visualization through Knowledge Graph

Visualizing PDF documents through a Knowledge Graph offers a visual and structured interpretation of the content, enriching the interaction experience with the document:

Understanding the Structure of the Document: Users can intuitively visualize the overall structure of the document, quickly understanding how information is organized and interconnected.

Detailed Analysis of Paragraphs: Each paragraph is explored through the Knowledge Graph, highlighting key entities and their relationships. This approach not only improves text comprehension but also facilitates the identification of significant patterns and connections within the content.

Direct Access to Relevant Information

The integration of these functionalities leads to a drastic reduction in the time and effort required to find specific information within complex documents:

Precise Information Localization: Users can jump directly to the page or section of the document where the answer to their question was found, optimizing the content review and analysis process.

Evidence-Based Responses: Each answer provided is supported by direct references to the document's content, ensuring that the information presented is not only relevant but also verifiable.

Usability and User Interface

Usability is a fundamental pillar in the architecture of any application, directly influencing the degree of satisfaction and efficiency of users in using the system. Designing a user interface that is intuitive, functional, aesthetically pleasing, and responsive represents a primary goal to ensure an exceptional user experience. In this section, the various aspects of the application's user interface are analyzed in detail, highlighting the design choices and functionalities

that contribute to the system's high usability.

Top Section: Interaction and Functionality

The top section of the interface is designed to facilitate direct and intuitive communication with the system, offering various interactive options:

Natural Language Querying: Users can type questions in natural language and receive contextualized responses based on the content of the uploaded PDF document. This approach promotes natural and direct interaction with the system.

Access to User Manual: An accessible user manual provides clear instructions on the optimal use of the application, enhancing the user experience and reducing the learning curve.

Document Management: Options such as uploading new PDF documents, selecting the current document, and clearing the interface allow users to easily manage their documents and interactions with the application.

Left Section: Content Visualization

This section provides a detailed visual representation of the document, significantly enhancing content comprehension:

Knowledge Graph: Graphically represents the overall structure of the document, highlighting chapters, images, and paragraphs. This synthetic and semantic visualization facilitates the analysis and interpretation of the document.

UMAP: Provides a two-dimensional map of the semantic similarities between sentence embeddings, offering a unique perspective on the semantic relationships within the document.

Right Section: Detail of Reference Paragraphs The right section focuses on reference paragraphs, offering an in-depth view of specific information:

Reference Paragraphs: Displays paragraphs that contain information relevant to the asked questions, enriched with more information about the paragraph (page number, chapter title, and paragraph title) and detailed knowledge graphs that highlight entities, relationships, and properties.

Interaction with Content: Users can explore and modify the visualization of the paragraph knowledge graphs, deepening their understanding of the content and analyzing the relationships between entities.

Images: Users can also view images contained in the various chapters.

Bottom Section: Response and Feedback The bottom section is dedicated to presenting responses and collecting user feedback:

Contextualized Response: Displays the system's response to the user's question, providing context and clear references that increase the user's trust and satisfaction.

Feedback: Allows users to provide their feedback on the received response, promoting continuous improvement of the system based on actual user interactions.

Design and Interaction Principles The design of the user interface is based on established principles of usability and interaction design:

Intuitive Navigation: The interface is organized in a clear and logical manner, allowing for smooth navigation and easy access to functionalities.

Aesthetics and Visibility: Special care has been taken with the aesthetic appearance and visual clarity, ensuring a pleasant and comprehensible environment for users.

Feedback and Interactivity: The interface dynamically responds to user input, providing immediate feedback and making the interaction with the system both satisfying and informative.

Informational Clarity: Information is presented in a clear and comprehensible manner, with the use of supportive visual elements to enrich the user experience and facilitate understanding of the content.

Flexibility

The flexibility of the application is a fundamental aspect that determines its ability to adapt to various contexts and requirements. In this section, the key factors that contribute to the application's flexibility are explored, highlighting both its potential and the challenges it presents.

Type and Format of the PDF Document

Versatility in Processing: The application demonstrates considerable versatility in processing a wide range of PDF documents, including books, theses, and manuals. It does not require a predefined structure, proving capable of adapting to the content and format of the uploaded document.

Challenges with Unstructured Documents: Despite its versatility, the application encounters difficulties with documents lacking a clear index. These obstacles affect the completeness and accuracy of the generated knowledge graph, highlighting the importance of quality input material.

Type and Complexity of the Natural Language Question

Ability to Respond to Diverse Questions: The application can handle a wide variety of questions, adapting to the complexity and diversity of user requests without requiring a particular structure for queries.

Limitations in Responses to Complex Questions: Challenges may arise in responding to overly complex or ambiguous questions. The quality of the response can vary based on the specificity and relevance of the question, highlighting the importance of clear and contextualized formulation.

Challenges in Handling Complex Documents in the Visualization of the Knowledge Graph Despite the flexible options, difficulties may arise in visualizing particularly long or information-rich documents. The density of information can make the knowledge graph difficult to interpret, requiring further development in data management and presentation.

Maintenance Maintainability is a crucial aspect that determines the longevity and adaptability of a software system over time. It concerns the ease with which the system can be updated, modified, or corrected in response to changes in requirements, technologies, or user needs. The system has been designed and implemented following a modular approach, which brings the following benefits:

Separation of Responsibilities: The code is organized into independent modules, each responsible for a specific functionality of the application. This principle of separation of responsibilities ensures that changes in one module do not directly affect the functions of other modules.

Ease of Updating and Extension: Thanks to the modular structure, the system can be easily updated or extended. Modification or addition of new functionalities can occur through intervention on specific modules, reducing the risk of errors and minimizing the impact on existing parts of the system.

CHAPTER 6

Conclusions and future developments

In the research conducted, exploration was made in the analysis and visualization of PDF documents, leveraging the potential of interactive knowledge graphs, Natural Language Processing (NLP), and Large Language Models (LLM). The developed system, the project, has demonstrated that an interactive and visual approach can revolutionize the way large sets of documents are interpreted and understood, leading to a significant increase in the ability to extract relevant insights and navigate through complex data collections. This project represents a significant step forward compared to existing solutions, offering a deep semantic understanding of documents and an interaction and visualization capability never seen before. The system has shown that, by integrating advanced technologies, it is possible to overcome the limitations of traditional methods and provide users with powerful tools to interpret and interact with data in innovative ways. I am particularly proud to having successfully integrated LLM models to provide contextual interpretation and advanced text generation.
Having created an interactive visualization that not only improves data navigation but also provides an intuitive understanding of the relationships and structures within the documents. Managing additional PDF structures that lack an index remains a challenge, requiring further innovation and experimentation for effective integration. The dependence on the specificity of prompts provided to LLMs requires careful balancing to maximize the system's efficiency, accessibility, and cost. The computational power required to run LLMs locally is an obstacle for offline, local use, and optimal customization. My project offers substantial improvements over previous solutions, including a richer and more interactive user experience that transforms static document analysis into a dynamic and engaging process.
The ability to handle and interpret the complexity and variability of natural language, surpassing the limitations of keyword-based search engines and rule-based systems. These features make this project not only a more advanced solution but also a promising platform for further

developments and innovations in the field.

For the future, I identify several promising directions: **Improvement of Table Extraction:** Implementation of more sophisticated techniques for the extraction and management of tables, potentially integrating artificial intelligence-based solutions to interpret and structure complex tabular data.

Optimization of LLMs: Continue refining and customizing LLMs to improve the accuracy and relevance of responses, especially in domain-specific contexts.

Reduction of Requirements: Work on optimizing the system to reduce its impact in terms of computational resources and cost, making it more accessible and sustainable.

Expansion of Querying Capabilities: Develop advanced capabilities to handle complex questions and query concatenations, providing even more precise and contextualized responses.

Description of Images: With LLMs specialized in computer vision, it would be possible to obtain a description of images found within PDF documents.

Multilingual Support: Implement models that support multiple languages beyond English.

Support for PDFs with Different Layouts: Implement strategies to support the creation of the structure of many more types of PDFs.

Summarization: Summarization of paragraphs and the PDF is definitely among the future developments.

Customization of the Knowledge Graph: Implementation of options to customize the appearance and arrangement of the knowledge graph, allowing users to adapt the visualization to their preferences and needs. For example, a total Knowledge Graph with all the paragraph knowledge graphs joined could be convenient and useful to get an overall idea of the content of the PDF document.

History: It is possible to add a history of all the questions and answers obtained previously.

Light/Dark Mode: It is possible to add an additional display mode that allows the use of darker colors to reduce eye strain in low light conditions.

Processing of Advanced Questions: Development of more sophisticated mechanisms for managing and responding to composite or concatenated questions, improving the depth and accuracy of natural language interactions. One way, for example, is Prompt Expansion, which allows dividing the question into multiple sub-questions that allow reaching the desired answer.

This project represents an important turning point in the way we interact with and analyze PDF documents. With the continuous development and adaptation of technologies such as LLMs and interactive knowledge graphs, we are on the threshold of a new era in data analysis, one that promises to radically transform our ability to extract, interpret, and use the information contained in digital documents. The road ahead of us is full of challenges, but also immense opportunities for innovation and progress.

Bibliography

[1] Eleni Adamopoulou and Lefteris Moussiades. “An overview of chatbot technology”. In: *IFIP international conference on artificial intelligence applications and innovations*. Springer. 2020, pp. 373–383.

[2] Nitish Aggarwal et al. “Knowledge graphs: In theory and practice”. In: *Conference on Information and Knowledge Management*. Vol. 17. 2017.

[3] Mistral AI. *Mistral*. 2023. URL: https://huggingface.co/mistralai.

[4] Felipe Almeida and Geraldo Xexéo. “Word embeddings: A survey”. In: *arXiv preprint arXiv:1901.09069* (2019).

[5] Zeynep Altan. “The role of morphological analysis in natural language processing”. In: (2002).

[6] Anaconda. *Anaconda*. 2024. URL: https://www.anaconda.com/.

[7] Sheena Angra and Sachin Ahuja. “Machine learning and its applications: A review”. In: *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*. IEEE. 2017, pp. 57–60.

[8] Grigoris Antoniou and Frank van Harmelen. “Web ontology language: Owl”. In: *Handbook on ontologies* (2009), pp. 91–110.

[9] Apple. *MacBook Air (M2, 2022) - Specifiche tecniche*. 2022. URL: https://support.apple.com/kb/SP869?locale=it_IT.

[10] Apple. *macOS Sonoma*. 2024. URL: https://support.apple.com/it-it/109035.

[11] Tim Bienz, Richard Cohn, and Calif.) Adobe Systems (Mountain View. *Portable document format reference manual*. Addison-Wesley Bostonˆ eMA MA, 1993.

[12] Kuei-Hu Chang. *Natural Language Processing: Recent Development and Applications*. 2023.

[13] ChatPDF. *ChatPDF*. 2024. URL: https://www.chatpdf.com/.

[14] Alebachew Chiche and Betselot Yitagesu. "Part of speech tagging: a systematic review of deep learning and machine learning approaches". In: *Journal of Big Data* 9.1 (2022), p. 10.

[15] Cohere. *Cohere*. 2024. URL: https://cohere.com/.

[16] Intel Corporation. *Intel Core i7-12650H Processor*. 2024. URL: https://www.intel.com/content/www/us/en/products/sku/226066/intel-core-i712650h-processor-24m-cache-up-to-4-70-ghz/specifications.html.

[17] Jennifer D'souza and Nandana Mihindukulasooriya. "The State of the Art on Knowledge Graph Construction from Text: Named Entity Recognition and Relation Extraction Perspectives". In: *The Knowledge Graph Conference 2022*. 2022.

[18] d3.js. *d3.js*. 2024. URL: https://d3js.org.

[19] Google for Developers. *Firebase*. 2024. URL: https://firebase.google.com/.

[20] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[21] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Bhumika Dutta. *Syntactic Analysis: an Overview*. 2021. URL: https://www.analyticssteps.com/blogs/syntactic-analysis-overview.

[23] Explosion. *en_core_web*. 2023. URL: https://spacy.io/models/en.

[24] Explosion. *spaCy*. 2024. URL: https://spacy.io/.

[25] Hugging Face. *Babelscape/rebel-large*. 2023. URL: https://huggingface.co/Babelscape/rebel-large.

[26] Hugging Face. *BERT*. 2020. URL: https://huggingface.co/docs/transformers/model_doc/bert.

[27] Hugging Face. *Bloom*. 2023. URL: https://huggingface.co/bigscience/bloom.

[28] Hugging Face. *Falcon-40b-instruct*. 2023. URL: https://huggingface.co/tiiuae/falcon-40b-instruct.

[29] Hugging Face. *Falcon-7b-instruct*. 2023. URL: https://huggingface.co/tiiuae/falcon-7b-instruct.

[30] Hugging Face. *Fuyu-8b*. 2023. URL: https://huggingface.co/adept/fuyu-8b.

[31] Hugging Face. *Gradio*. 2024. URL: https://www.gradio.app/.

[32] Hugging Face. *Guanaco*. 2023. URL: https://huggingface.co/models?other=guanaco.

[33] Hugging Face. *Persimmon-8b-base*. 2023. URL: https://huggingface.co/adept/persimmon-8b-base.

[34] Hugging Face. *Sheared-LLama 2.7B*. 2023. URL: https://huggingface.co/princeton-nlp/Sheared-LLaMA-2.7B.

[35] Hugging Face. *Transformers*. 2024. URL: https://huggingface.co/docs/transformers/index.

[36] Hugging Face. *Zephyr-7b-alpha*. 2023. URL: https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha.

[37] Mathieu Fenniak. *PyPDF2*. 2022. URL: https://pypi.org/project/PyPDF2/.

[38] Gil Fink et al. "Introducing single page applications". In: *Pro single page application development: Using backbone. js and asp. net* (2014), pp. 3–13.

[39] Python Software Foundation. *Abstract Syntax Trees*. 2023. URL: https://docs.python.org/3/library/ast.html.

[40] Python Software Foundation. *Base16, Base32, Base64, Base85 Data Encodings*. 2023. URL: https://docs.python.org/3/library/base64.html.

[41] Python Software Foundation. *Core tools for working with streams*. 2023. URL: https://docs.python.org/3/library/io.html.

[42] Python Software Foundation. *High-level file operations*. 2023. URL: https://docs.python.org/3/library/shutil.html.

[43] Python Software Foundation. *JSON encoder and decoder*. 2024. URL: https://docs.python.org/3/library/json.html.

[44] Python Software Foundation. *Python*. 2024. URL: https://www.python.org/.

[45] Python Software Foundation. *Regular expression operations*. 2024. URL: https://docs.python.org/3/library/re.html.

[46] Python Software Foundation. *Support for type hints*. 2024. URL: https://docs.python.org/3/library/typing.html.

[47] Python Software Foundation. *UUID objects according to RFC 4122*. 2024. URL: https://docs.python.org/3/library/uuid.html.

[48] Anthony Gillioz et al. "Overview of the Transformer-based Models for NLP Tasks". In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2020, pp. 179–183.

[49] GitHub. *GitHub*. 2024. URL: https://github.com/.

[50] Google. *Google Colab*. 2024. URL: https://colab.research.google.com/.

[51] GPT. *GPT*. 2024. URL: https://openai.com.

[52] Aidan Hogan et al. "Knowledge graphs". In: *ACM Computing Surveys (Csur)* 54.4 (2021), pp. 1–37.

[53] Nick Horvath. *Pyrebase4*. 2023. URL: https://github.com/nhorvath/Pyrebase4.

[54] THE WHITE HOUSE. *BLUEPRINT FOR AN AI BILL OF RIGHTS*. 2022. URL: https://www.whitehouse.gov/ostp/ai-bill-of-rights/.

[55] ijrti. *RESEARCH PAPER ON ARTIFICIAL INTELLIGENCE & ITS APPLICATIONS*. 2023. URL: https://www.ijrti.org/papers/IJRTI2304061.pdf.

[56] Samkit Jain Jeremy Singer-Vine. *PDFPlumber*. 2023. URL: https://github.com/jsvine/pdfplumber.

[57] Peiling Jiang et al. "Graphologue: Exploring large language model responses with interactive diagrams". In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–20.

[58] Diksha Khurana et al. "Natural language processing: State of the art, current trends and challenges". In: *Multimedia tools and applications* 82.3 (2023), pp. 3713–3744.

[59] Logan Kilpatrick. *tiktoken*. 2023. URL: https://github.com/openai/tiktoken.

[60] Geewook Kim. *Donut*. 2022. URL: https://github.com/clovaai/donut.

[61] Geewook Kim et al. "Ocr-free document understanding transformer". In: *European Conference on Computer Vision*. Springer. 2022, pp. 498–517.

[62] Moritz Klack. *React Flow*. 2024. URL: https://reactflow.dev/.

[63] Saurabh Kumar. *python-dotenv*. 2024. URL: https://github.com/theskumar/python-dotenv.

[64] John P Lalor, Hao Wu, and Hong Yu. "Improving machine learning ability with fine-tuning". In: *CoRR, abs/1702.08563. Version* 1 (2017).

[65] Matthias A Lee. *Python Tesseract*. 2023. URL: https://github.com/madmaze/pytesseract.

[66] Jamie Lemon. *PyMuPDF*. 2024. URL: https://pymupdf.readthedocs.io/en/latest/.

[67] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[68] Jing Li et al. "A survey on deep learning for named entity recognition". In: *IEEE transactions on knowledge and data engineering* 34.1 (2020), pp. 50–70.

[69] Timo Lins. *React Hot Toast*. 2023. URL: https://react-hot-toast.com/.

[70] Tanner Linsley. *TanStack Query v5*. 2024. URL: https://tanstack.com/query/latest.

[71] Lordicon. *Lordicon*. 2024. URL: https://lordicon.com/.

[72] Suja Cherukullapurath Mana and T Sasipraba. "Research on cosine similarity and pearson correlation based recommendation models". In: *Journal of Physics: Conference Series*. Vol. 1770. 1. IOP Publishing. 2021, p. 012014.

[73] Dastan Hussen Maulud et al. "State of art for semantic analysis of natural language processing". In: *Qubahan academic journal* 1.2 (2021), pp. 21–28.

[74] Andrew McAfee et al. "Big data: the management revolution". In: *Harvard business review* 90.10 (2012), pp. 60–68.

[75] Meta. *Client React DOM APIs*. 2024. URL: https://react.dev/reference/react-dom/client.

[76] Meta. *FastText*. 2020. URL: https://fasttext.cc/docs/en/python-module.html.

[77] Meta. *LLama*. 2023. URL: https://huggingface.co/meta-llama.

[78] Meta. *React*. 2022. URL: https://react.dev/.

[79] Metiora. *Traditional vs. Modern Data Science*. 2017. URL: https://medium.com/@Metiora/traditional-vs-modern-data-science-6f8b2fc0df27.

[80] Janis Meyer. *Deepdoctection*. 2023. URL: https://github.com/deepdoctection/deepdoctection.

[81] Microsoft. *LayoutLMv3*. 2022. URL: https://github.com/microsoft/unilm/tree/master/layoutlmv3.

[82] Ishani Mondal, Yufang Hou, and Charles Jochim. "End-to-end NLP knowledge graph construction". In: *arXiv preprint arXiv:2106.01167* (2021).

[83] Ishani Mondal, Yufang Hou, and Charles Jochim. "End-to-end NLP knowledge graph construction". In: *arXiv preprint arXiv:2106.01167* (2021).

[84] Ishani Mondal, Yufang Hou, and Charles Jochim. "End-to-end NLP knowledge graph construction". In: *arXiv preprint arXiv:2106.01167* (2021).

[85] MongoDB. *PyMongo*. 2023. URL: https://www.mongodb.com/docs/drivers/pymongo/.

[86] Humza Naveed et al. "A comprehensive overview of large language models". In: *arXiv preprint arXiv:2307.06435* (2023).

[87] David Newell. *dagre - Graph layout for JavaScript*. 2023. URL: https://github.com/dagrejs/dagre.

[88] NLTK. *NLTK*. 2024. URL: https://www.nltk.org.

[89] Ankita Nohria and Harkiran Kaur. "Evaluation of Parsing Techniques in Natural Language Processing". In: ().

[90] NumPy. *NumPy*. 2023. URL: https://numpy.org/.

[91] NVIDIA. *Driver NVIDIA Studio*. 2023. URL: https://www.nvidia.com/it-it/geforce/drivers/results/217058/.

[92] NVIDIA. *GeForce RTX 4070 family*. 2024. URL: https://www.nvidia.com/it-it/geforce/graphics-cards/40-series/rtx-4070-family/.

[93] OpenAI. *GPT*. 2024. URL: https://openai.com.

[94] OpenAI. *Models*. 2024. URL: https://platform.openai.com/docs/models/overview.

[95] OpenAI. *Models*. 2024. URL: https://platform.openai.com/docs/models/overview.

[96] Irina Pak and Phoey Lee Teh. "Text segmentation techniques: a critical review". In: *Innovative Computing, Optimization and Its Applications: Modelling and Simulations* (2018), pp. 167–181.

[97] Pallets. *Flask*. 2010. URL: https://flask.palletsprojects.com/en/3.0.x/.

[98] Jeff Z Pan. "Resource description framework". In: *Handbook on ontologies*. Springer, 2009, pp. 71–90.

[99] Ciyuan Peng et al. "Knowledge graphs: Opportunities and challenges". In: *Artificial Intelligence Review* 56.11 (2023), pp. 13071–13102.

[100] Giancarlo Perrone. *Pyvis*. 2018. URL: https://pyvis.readthedocs.io/.

[101] Matt Perry. *Framer Motion*. 2024. URL: https://github.com/framer/motion.

[102] PyTorch. *torch*. 2023. URL: https://pypi.org/project/torch/.

[103] Johannes Rausch et al. "DSG: An End-to-End Document Structure Generator". In: *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2023, pp. 518–527.

[104] Radim Rehurek. *Word2Vec*. 2022. URL: https://radimrehurek.com/gensim/models/word2vec.html.

[105] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[106] Armin Ronacher. *Flask-CORS*. 2013. URL: https://flask-cors.readthedocs.io/en/latest/.

[107] Yusuke Shinyama. *PDFMiner*. 2020. URL: https://github.com/euske/pdfminer.

[108] sigma.js. *sigma.js*. 2024. URL: https://www.sigmajs.org.

[109] K Sindhura. "SENTIMENT ANALYSIS USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING". In: *Journal of Data Acquisition and Processing* 38.2 (2023), p. 520.

[110] Markus Stoll. *Visualize your RAG Data — EDA for Retrieval-Augmented Generation*. 2024. URL: https://itnext.io/visualize-your-rag-data-eda-for-retrieval-augmented-generation-0701ee98768f.

[111] Pinecone Systems. *Pinecone*. 2024. URL: https://www.pinecone.io/.

[112] Vivienne Sze et al. "Efficient processing of deep neural networks: A tutorial and survey". In: *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329.

[113] The Matplotlib development team. *Matplotlib*. 2023. URL: https://matplotlib.org/.

[114] Amit Timalsina. *How to extract data from PDF?* 2024. URL: https://www.docsumo.com/blog/extract-data-from-pdf.

[115] Gregory Vial. "Understanding digital transformation: A review and a research agenda". In: *Managing digital transformation* (2021), pp. 13–66.

[116] Ellen M Voorhees. "Natural language processing and information retrieval". In: *International summer school on information extraction*. Springer, 1999, pp. 32–48.

[117] Zhuo Wang et al. "Scalable rule-based representation learning for interpretable classification". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 30479–30491.

[118] Stefan Weil. *Tesseract OCR*. 2024. URL: https://github.com/tesseract-ocr/tesseract.

[119] Wikipedia. *Apple M2*. 2024. URL: https://it.wikipedia.org/wiki/Apple_M2.

[120] Wikipedia. *Portable Document Format*. 2024. URL: https://it.wikipedia.org/wiki/Portable_Document_Format.

[121] xiaoting. *PaddleOCR*. 2023. URL: https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.7/README_en.md.

[122] Gokul Yenduri et al. "Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions". In: *arXiv preprint arXiv:2305.10435* (2023).

[123] Kwan-Hee Yoo, Carson K Leung, and Aziz Nasridinov. *Big data analysis and visualization: challenges and solutions*. 2022.

[124] Matt Zabriskie. *Promise based HTTP client for the browser and node.js*. 2024. URL: `https://axios-http.com/`.

[125] Jiawei Zhao et al. "GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection". In: *arXiv preprint arXiv:2403.03507* (2024).

[126] Lingfeng Zhong et al. "A comprehensive survey on automatic knowledge graph construction". In: *ACM Computing Surveys* 56.4 (2023), pp. 1–62.

[127] Vilém Zouhar et al. "Artefact retrieval: Overview of NLP models with knowledge base access". In: *arXiv preprint arXiv:2201.09651* (2022).

Gratitude

In this significant moment marking the conclusion of my academic journey, I feel the need to express my most sincere gratitude to those who have made this academic voyage possible, not only by supporting me but also by enriching it with profound meaning and unforgettable experiences.

First and foremost, I wish to extend a special thought to my family. You have been the pillars upon which I could always rely, the beacon that has illuminated my darkest days, and an inexhaustible source of love and support. Without your unconditional affection, your words of encouragement, and your unwavering belief in my abilities, I would not be the person I am today. Thank you for being my home, the place where, no matter how far I may go, I know I can always return.

Next, I would like to express my deep gratitude to my advisors. To my Italian advisor, a heartfelt thank you for skillfully guiding my ambitions and curiosity, transforming the research journey into a stimulating and enriching intellectual adventure. Your wisdom and attentive guidance provided the fertile ground on which my ideas could sprout and grow robust.

A special thanks goes to my Icelandic advisor, whose influence on this work and my personal growth has been immense. Your ability to push beyond the boundaries of conventional thinking, challenge preconceived notions, and foster endless curiosity has opened new horizons in my research and my worldview. Your unique approach to knowledge, intertwining intellectual depth and rare human sensitivity, has left an indelible mark on my education. Thank you for believing in me, for dedicatedly accompanying me on this journey of discovery, and for teaching me the value of intellectual integrity and passion in research.

I cannot forget to thank the advisor from the business field, whose practical and career-oriented contributions greatly enriched my thesis project. Your experience and willingness to share knowledge and insights added a tangible and concrete dimension to my education, opening my eyes to the real-world applications of the concepts studied.

Finally, I want to acknowledge all those who, even though not mentioned individually, contributed to my journey with small gestures, words of encouragement, or simple smiles at the right moments. Every act of kindness has left an indelible mark on my path, and for this, I am deeply grateful.

I conclude these acknowledgments with a sense of gratitude that goes beyond words, with a heart filled with hope for the future, and with a strong determination to honor, through my future actions, the support and affection you have generously bestowed upon me.

Thank you from the bottom of my heart to all of you.