Abstract

The increasing volume and complexity of digital documents have created a pressing need for automated solutions capable of extracting structured information from unstructured or semi-structured data sources. This project presents the design and development of an AI-powered document data extraction system aimed at simplifying the process of retrieving key information from various document formats such as PDF, DOCX, TXT, and image files (JPG, PNG).

The system integrates advanced natural language processing and vision-language models to analyze both textual content and document layouts, enabling accurate extraction of information even from complex, non-standard document structures. The extracted data is stored in a structured format within a MongoDB database, facilitating efficient querying, filtering, and export.

A user-friendly web interface, developed using Next.js, React, and Tailwind CSS, allows users to upload documents, view extracted data, perform real-time queries, and export results to Excel. The system leverages cloud services, including Google's Generative AI for parsing, Cloudinary for secure file storage, and Vercel for deployment, ensuring scalability, reliability, and ease of maintenance.

By automating the traditionally manual and error-prone task of information extraction, this project aims to enhance operational efficiency, reduce human errors, and provide a scalable solution adaptable to diverse use cases such as resume parsing, invoice processing, and document digitization workflows.

Keywords

Document Processing; Information Extraction; Natural Language Processing; Vision-Language Models; Next.js; React; Tailwind CSS; MongoDB; Google Generative AI; Cloudinary; Resume Parsing; Data Automation; AI-powered Extraction.

1. Introduction

1.1 Introduction

In today's digital landscape, the increasing use of digital documents such as resumes, invoices, and reports has created a demand for automated solutions that can extract meaningful information quickly and accurately. Manual extraction is time-consuming, error-prone, and inefficient, especially when handling large volumes of documents in various formats like PDF, DOCX, TXT, and images.

The **Resume Parsing System** is designed to address this challenge by leveraging artificial intelligence (AI) and advanced document parsing techniques. By combining machine learning models, vision-language processing, and scalable cloud infrastructure, the system can intelligently extract structured data from unstructured and semi-structured documents. This makes it easier for businesses, HR teams, and data analysts to access, filter, and process relevant information without manual intervention.

The system features a modern web interface, built using technologies like **Next.js, React, and Tailwind CSS**, ensuring an intuitive user experience. It supports multiple file formats, integrates with third-party services like Google Generative AI for parsing, and offers functionalities such as filtering, querying, and exporting data to Excel.

1.2 Rationale

The rationale behind this project is to improve the efficiency, accuracy, and scalability of data extraction from diverse document types. Traditional methods like Optical Character Recognition (OCR) have limitations when dealing with complex layouts, tables, or mixed content formats. Manual extraction, on the other hand, is slow and often leads to human errors.

Our system aims to overcome these challenges by:

- Reducing manual effort and human error
- Handling a wide range of file types
- Delivering faster processing times and real-time querying
- Providing structured, ready-to-use data for downstream systems or analytics
  This makes the system valuable for industries like HR, finance, legal, and education, where document processing is central to operations.

1.3 Objectives

The primary objectives of the project are:

- To develop a scalable, AI-powered system for extracting structured data from unstructured documents.
- To support multiple document formats, including DOCX, PDF, TXT, and image files.
- To enable users to query extracted data through a user-friendly web interface.

- To allow exporting filtered data into Excel format.
- To reduce processing time and human errors compared to manual extraction.
- To integrate with cloud-based storage and deployment solutions for better scalability.

1.4 Scope of Project

The scope of the project includes:

- Development of a web application with an intuitive frontend and robust backend.
- Support for uploading and parsing resumes and similar documents.
- Integration with AI models (e.g., Google Gemini) for intelligent data extraction.
- Storage and management of extracted data in a MongoDB database.
- Exporting and filtering functionalities for end-users.
- Deployment on scalable cloud platforms (Vercel, Cloudinary).

The project **does not** cover tasks like job matching, applicant ranking, or custom AI training from scratch — it focuses solely on document parsing and data extraction.

1.5 Problem Statement

Manually extracting important information from documents like resumes, invoices, or text files is a time-consuming and error-prone process. Existing OCR or rule-based tools fail to handle diverse formats, complex layouts, and varied data types effectively.

The proposed system aims to solve this problem by:

- Automating the extraction process using AI
- Handling multiple file types and layouts robustly
- Providing users with fast, accurate, and searchable data
  By addressing these challenges, the system improves productivity, reduces costs, and increases the reliability of document processing workflows.

2. Literature Review

The literature review aims to provide an overview of prior research, methods, tools, and technologies relevant to document parsing and data extraction. It lays the foundation for understanding the challenges in this domain and highlights how our project builds upon and advances existing work.

2.1 Traditional Approaches

1. **Optical Character Recognition (OCR)**

OCR has been a foundational technology in digitizing printed or handwritten documents into machine-readable text. Tools like **Tesseract OCR** and **ABBYY FineReader** have been widely used in academic and commercial contexts to extract text from scanned images and PDFs.
However, OCR faces major limitations:

- Poor performance on low-resolution or noisy images
- Lack of understanding of document structure and layout
- Inability to handle tabular or complex multi-column layouts effectively

---

2. **Text Extraction Libraries**

Libraries such as **PyMuPDF**, **PDFMiner**, and **Apache PDFBox** allow the extraction of raw text from PDFs and other digital documents.
Limitations include:

- Lack of semantic understanding
- No capability to identify entities or relationships between data
- Struggles with multi-format documents (mix of text, tables, and images)

2.2 Advanced Techniques in Literature

1. **Named Entity Recognition (NER) and Machine Learning Models**

Recent years have seen a shift toward using NER, a subfield of NLP, to extract important entities (names, dates, locations, etc.) from documents.
For example:

- **Spacy NER**, **Stanford CoreNLP**, and **BERT-based models** are capable of extracting meaningful entities.
- **Limitation**: Require large labeled datasets for fine-tuning and perform best in predictable, narrow domains.

---

2. **Table Extraction**

Libraries like **pdfplumber** and **Camelot** specialize in table extraction by recognizing table borders and preserving table structure.

- **Advantage**: Maintains relational data.
- **Limitation**: Struggles with unstructured or irregular tables; sensitive to document quality.

---

3. **Vision-Language Models**

Emerging approaches combine computer vision and NLP to understand both **visual layout** and **textual content** in documents.

- Example: Google's **Donut model**, Microsoft's **LayoutLM**, and Google Gemini.
- **Advantage**: Understands the spatial context and relationships within documents, making them effective for complex layouts.

2.3 Key Related Works

Reference	Summary of Contribution
Maniyar et al., IRJET (2020)	Proposed automatic data extraction from PDFs and web sources with dynamic database creation. Highlighted the use of automation to reduce manual effort.
Fang Yuan et al., Hebei University, China	Studied information extraction from PDFs using additional uniform tags to create semi-structured data, improving queryability.
Mohamed Afifi et al., Systematic Reviews (2023)	Developed a step-by-step guide and implementation example for data extraction in systematic reviews using open-source tools.
G. Midhu Bala, International Journal of Arts, Science, and Humanities	Demonstrated web scraping techniques using R to extract and store filtered data from web pages.
N. Nagajothi et al., 2022	Presented geometric methods for image extraction from PDFs, contributing to improved visual extraction capabilities.

2.4 Gaps and Limitations in Existing Research

Despite the progress, existing solutions have critical gaps:

- **Scalability issues**: Many systems work well only in controlled environments.
- **Limited multimodal capabilities**: Traditional methods often ignore visual context, focusing only on text.

- **High customization needs**: Many solutions require manual rule setup or model fine-tuning.
- **Handling of diverse formats**: No single tool effectively handles DOCX, PDF, image, and text formats uniformly.

2.5 Contribution of This Project

Our project moves beyond traditional methods by:

- Combining **AI-based vision-language models** to analyze both the textual and visual components of documents.
- Supporting **multi-format inputs** including DOCX, PDF, image, and text files.
- Providing an **interactive web interface** for uploading, querying, filtering, and exporting data.
- Leveraging **cloud services (Google AI, Cloudinary, MongoDB Atlas)** for scalability.
- Incorporating **state-of-the-art frontend tools (Next.js, React, Tailwind)** for a modern and user-friendly experience.

3. Project Feasibility

The feasibility study evaluates whether the proposed system is practical, achievable, and beneficial in terms of technical, operational, economic, and legal dimensions. This section ensures that the project has a clear path to successful implementation and justifies the investment of time, effort, and resources.

3.1 Feasibility Study

We break the feasibility study into four key areas:

---

1. Technical Feasibility

This evaluates whether the technology required for the project is available and capable of fulfilling the project requirements.

- **Frontend**
  The system will use **Next.js 13.5.1 with TypeScript**, **React 18.2.0**, and **Tailwind CSS**. These are modern, battle-tested frameworks and libraries that ensure a fast, responsive, and scalable frontend.
- **Backend**
  The backend will be built using **Next.js API Routes**, which allow us to handle server-side logic efficiently, combined with **Google Generative AI (Gemini 1.5 Flash)** for intelligent parsing and **Mammoth.js** for DOCX parsing.
- **Database**
  We will use **MongoDB Atlas**, a cloud-hosted NoSQL database with **Mongoose ORM** for managing schema and queries.
- **File Storage**
  Cloudinary will handle secure file uploads and management, with support for images, PDFs, and DOCX files.
- **Deployment Platform**
  The system will be deployed on **Vercel**, which provides automated deployments, serverless functions, edge support, and CI/CD integration.

All required technologies are available, well-documented, and compatible, making the project technically feasible.

---

2. Operational Feasibility

This focuses on whether the system will operate effectively within the intended environment.

- **End-User Perspective**
  The system will offer an intuitive interface for HR teams, businesses, or data processors, allowing them to upload, process, and analyze documents without technical expertise.

- **Admin Perspective**
  Admin users can monitor system performance, manage files, and oversee data flow through a backend dashboard.
- **Integration Potential**
  The system is designed to integrate smoothly with third-party services (Google AI, Cloudinary) and can be expanded into larger ecosystems, such as ERP or HRMS platforms.

The system meets user expectations and operational needs.

---

3. Economic Feasibility

This assesses whether the system's benefits outweigh its costs.

- **Development Costs**
  Since we are using many open-source technologies (React, Next.js, MongoDB Community), the software costs are minimal. Paid services like Cloudinary and Google AI are usage-based and scalable.
- **Deployment and Maintenance Costs**
  Vercel's free tier or startup plans make initial deployment cost-effective. As usage grows, scaling costs can be absorbed progressively.
- **Return on Investment (ROI)**
  By reducing manual labor, improving accuracy, and speeding up document processing, the system can achieve a high ROI, especially for businesses dealing with large volumes of resumes or documents.

The project is economically viable with low upfront costs and high long-term savings.

---

4. Legal Feasibility

This ensures that the system complies with relevant laws and regulations.

- **Data Privacy and Security**
  Secure file handling using **Cloudinary**, type-safe data management with **TypeScript**, and encrypted database storage with **MongoDB** ensure data protection.
- **Compliance Requirements**
  The system will comply with applicable data protection standards such as GDPR (for EU users) or India's Digital Personal Data Protection Act if handling sensitive personal information.

The project can comply with legal and regulatory requirements.

---

4. Methodology / Planning of Work

The successful development and deployment of the Resume Parsing and Document Data Extraction System require a structured and well-defined methodology. This section outlines the systematic approach adopted to ensure efficient, scalable, and accurate system development, from requirement gathering to deployment and testing.

4.1 Methodology

The project methodology follows a **phased, iterative development cycle** inspired by Agile principles, ensuring flexibility, early feedback, and continuous improvement.

Phase 1: Requirement Analysis and Planning

- Conducted detailed requirement gathering through discussions with stakeholders.
- Identified target users (HR teams, businesses, data analysts) and their pain points.
- Defined core system features:
    - Multi-format document upload (PDF, DOCX, TXT, image)
    - AI-based data extraction
    - Data querying and filtering
    - Excel export functionality
- Created a detailed project roadmap and timeline.

Phase 2: System Design

- Designed the system architecture, including:
    - **Frontend architecture** using Next.js, React, Tailwind CSS, and shadcn/ui.
    - **Backend architecture** using Next.js API routes and MongoDB with Mongoose.
    - **AI integration layer** leveraging Google Generative AI (Gemini 1.5 Flash) and Mammoth.js.
    - **File storage system** integrated with Cloudinary.
- Created **wireframes** and **mockups** to ensure a user-friendly interface.
- Defined database schema and data models.

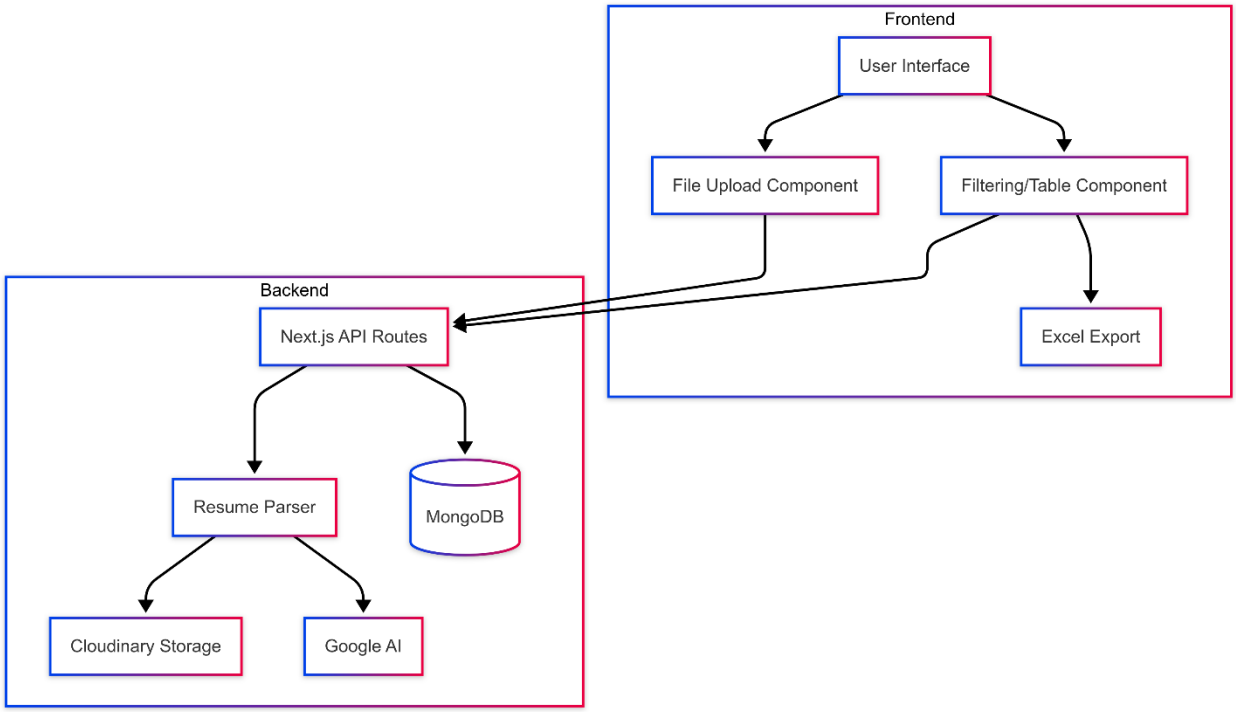

Figure 1: System Architecture

Phase 3: Development

- **Frontend Development**
  - Built the web interface with Next.js and React.
  - Implemented UI components using shadcn/ui and Lucide React icons.
  - Applied responsive design principles using Tailwind CSS.
  - Developed filtering and export features using the XLSX library.
- **Backend Development**
  - Created API routes for document upload, parsing, and data retrieval.
  - Integrated Google Generative AI for intelligent document parsing.
  - Implemented Mammoth.js for DOCX parsing.
  - Connected backend to MongoDB Atlas for data persistence.
- **File Handling and Storage**
  - Integrated Cloudinary for secure file storage.
  - Implemented file type validation and secure upload mechanisms.

Phase 4: Testing

- **Unit Testing**
    - Conducted unit tests on frontend components and backend services.
- **Integration Testing**
    - Verified end-to-end workflows: document upload → parsing → storage → display.
- **Performance Testing**
    - Tested system performance under high-volume file uploads.
- **Security Testing**
    - Ensured secure handling of sensitive data and compliance with privacy standards.

Phase 5: Deployment and Maintenance

- Deployed the system on **Vercel**, leveraging automatic deployments from the main branch.
- Configured environment variables and serverless functions.
- Set up continuous integration / continuous deployment (CI/CD) pipelines.
- Scheduled regular maintenance and updates for improvements, bug fixes, and security patches.

Key Development Practices

- **Version control** using Git and GitHub.
- **Agile workflow** with sprints, daily stand-ups, and regular retrospectives.
- **Code reviews** to ensure quality and maintainability.
- **Documentation** for architecture, APIs, and user guidelines.

4.2 Use Case Diagram

Below is an overview of the main actors and system interactions, represented in the form of a use case description:

- **User (HR/Business Analyst)**
  Uploads documents, queries extracted data, filters results, and exports to Excel.

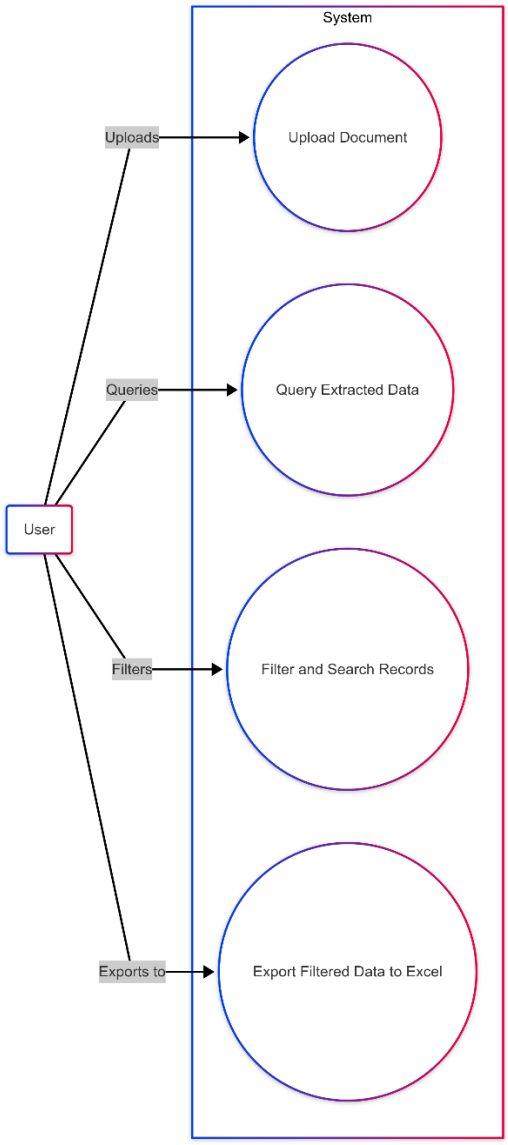

Figure 2: Use Case Diagram

This interaction model ensures a smooth, role-specific experience for both end-users and administrators.

5. Facilities Required for Proposed Work

The successful implementation of the proposed document data extraction system requires specific software and hardware facilities to ensure optimal development, testing, deployment, and maintenance. The selection of these facilities is driven by the need for a scalable, secure, and efficient architecture that aligns with modern web application standards.

5.1 Software and Hardware Requirements

This section outlines the software and hardware resources essential for the development and operation of the system.

5.1.1 Software Requirements

The following software tools and frameworks are required:

Sr. No.	Software / Tool	Version	Purpose
1	**Next.js**	13.5.1	Frontend and backend framework
2	**React**	18.2.0	Building user interface components
3	**TypeScript**	Latest stable	Type-safe development
4	**Tailwind CSS**	Latest stable	Styling and responsive design
5	**shadcn/ui**	Latest stable	Prebuilt UI components
6	**Lucide React**	Latest stable	Icon library
7	**XLSX Library**	Latest stable	Excel export functionality
8	**Mammoth.js**	Latest stable	DOCX parsing
9	**MongoDB with Mongoose ORM**	Latest stable	Database and schema management
10	**Google Gemini (Generative AI)**	1.5 Flash	AI-powered resume parsing
11	**Cloudinary SDK**	Latest stable	Cloud-based file storage
12	**Vercel CLI**	Latest stable	Deployment and CI/CD

Other development utilities include Git (version control), Postman (API testing), and an IDE like Visual Studio Code.

5.1.2 Hardware Requirements

The hardware requirements are modest, focusing primarily on supporting local development and testing. Cloud-based services (e.g., Vercel, MongoDB Atlas, Cloudinary) reduce the need for extensive local server hardware.

Sr. No.	Hardware Component	Minimum Specification
1	Processor	Intel Core i5 or equivalent (quad-core)
2	RAM	8 GB
3	Storage	256 GB SSD
4	Display	1080p resolution
5	Internet	Stable broadband (minimum 10 Mbps)

For production deployment and scaling, cloud hosting (via Vercel) and managed database services (MongoDB Atlas) will handle server-side and database operations efficiently.

6. Expected Outcomes

The proposed system is designed to deliver a comprehensive solution for automated data extraction from documents using artificial intelligence. Upon successful implementation, the system is expected to provide outcomes across user interaction, operational efficiency, data management, and security. This section outlines the anticipated deliverables of the project.

6.1 User Outputs

The system will present users with a streamlined, interactive platform that allows seamless upload, processing, and querying of documents. The frontend interface will prioritize usability while maintaining functional robustness.

6.1.1 User Dashboard

A web-based user dashboard will be developed to facilitate:

- Uploading of various document types (PDF, DOCX, TXT, JPG, PNG).
- Visualization of extracted data in structured formats.
- Interactive filtering and querying of extracted information.
- Export of filtered data into Excel format for offline analysis.

The dashboard will be responsive, accessible across devices, and designed with modern UI/UX principles.

6.2 Operational Outputs

Operational outcomes focus on automating the traditionally manual process of information extraction, reducing human error, and improving efficiency.

6.2.1 Automated Data Extraction

The system will automate the extraction of relevant information from documents using AI models capable of processing both textual and visual components. This automation will:

- Eliminate the need for manual data entry.
- Handle varying document formats and layouts with high accuracy.
- Process multiple documents in batch or individual mode.

6.2.2 Real-Time Query Handling

Users will be able to query the extracted data in real-time through the web interface. The system will respond to user queries with concise, accurate information retrieved from the processed documents.

6.3 Data Outputs

The system is expected to generate structured, queryable data outputs suitable for further integration or analysis.

6.3.1 Database Management

All extracted data will be stored in a secure MongoDB database, structured to enable efficient querying, filtering, and retrieval. The system will ensure data integrity and consistency across operations.

6.3.2 Export Capabilities

Users will have the option to export filtered datasets into Excel files using the integrated XLSX library, allowing portability and external use of the data.

6.4 Security and Integrity Outputs

Given the sensitive nature of document data, the system will prioritize security measures to protect user data and maintain system integrity.

Key security outcomes include:

- Validation of uploaded file types and formats.
- Secure file storage using Cloudinary with access control mechanisms.
- Type-safe handling of all data using TypeScript to prevent runtime vulnerabilities.
- Secure database connections via MongoDB Atlas with authentication and encryption.

7. Bibliography

This section lists the references and resources consulted during the research and development of the proposed system. The bibliography includes academic publications, technical documentation, and relevant web resources.

7.1 References

1. Ion Muslea, Steve Minton, and Craig A. Knoblock. *A Hierarchical Approach to Wrapper Induction*, Proceedings of the Third International Conference on Autonomous Agents, Seattle, WA, 1999, pp. 221–227.
2. Nicholas Kushmerick. *Wrapper Induction: Efficiency and Expressiveness*, Artificial Intelligence, vol. 118, no. 1–2, 2000, pp. 15–68.
3. Zhu Ming, Wang Jun, Wang Junpu. *Multiple Record Extraction from HTML Page Based On Hierarchical Pattern*, Computer Engineering, vol. 27, no. 9, 2001, pp. 40–42.
4. Zhu Ming, Huang Yun, Cai Qingsheng. *Information Extraction From Web Pages Based On Multi-Knowledge*, Mini-Micro System, vol. 22, no. 9, 2001, pp. 1058–1061.
5. Adobe Systems Incorporated. *Adobe Portable Document Format Version 1.4*, American Addison Wesley, 2001.
6. Ben Litchfield. *PDFBox Project*, Available at: http://sourceforge.net/projects/pdfbox.
7. Mohamed Afifi. *Data Extraction and Comparison for Complex Systematic Reviews: A Step-by-Step Guideline and an Implementation Example Using Open-Source Software*, Systematic Reviews, 2023.
8. Kiran Adnan and Rehan Akbar. *An Analytical Study of Information Extraction from Unstructured and Multidimensional Big Data*, Big Data Analytics, vol. 6, no. 91, 2019.
9. Fang Yuan, Bo Liu. *A New Method of Information Extraction from PDF Files*, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18–21 August 2005.
10. G. Midhu Bala. *Data Extraction and Scratching Information Using R*, International Journal of Arts, Science and Humanities, 2021.
11. Budak Arpinar, John Miller, Amit P. Sheth. *An Efficient Data Extraction and Storage Utility for XML Documents*, Available at: ResearchGate.
12. N. Nagajothi, S. S. Dhenakaran, S. Uma Maheswari. *A Geometric Method for Extracting Images of PDF Files*, 2022 International Conference.

7.2 Additional Resources

1. Kili Technology. *Efficient Key Information Extraction for Document Processing: 2024 Guide*, Available at: https://kili-technology.com/data-labeling/efficient-key-informationextraction-for-document-processing-2024-guide.

2. ResearchGate. *An Efficient Data Extraction and Storage Utility for XML Documents*, Available at: https://www.researchgate.net/publication/2497344_An_Efficient_Data_Extraction_and_Storage_Utility_For_XML_Documents.