2026 Volume: 02 Issue: 03 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-36

A COMPARATIVE ANALYSIS OF ZERO-SHOT, RETRIEVAL-AUGMENTED, AND RULE-BASED METHODS FOR STRUCTURED DATA EXTRACTION FROM BUDGETARY DOCUMENTS

[*1]Uchenna Onyeji, [2]Frank Ekene Ozioko, [3]Ozioko Anastasia Chinonye

[1,2]*Department of Computer Science, Enugu State University of Science and Technology (ESUT)* Enugu State, Nigeria.

[3]*Department of Accountancy, Enugu State Polytechnic (ESPOLY), Iwollo* Enugu State, Nigeria.

Article Received: 15 February 2026
Article Revised: 06 March 2026
Published on: 26 March 2026

***Corresponding Author: Uchenna Onyeji**
Department of Computer Science, Enugu State University of Science and Technology (ESUT) Enugu State, Nigeria.

DOI: https://doi-doi.org/101555/ijrpa.4798

ABSTRACT

The automated extraction of structured data from semi-structured government documents remains a significant technical challenge for public sector transparency and admin- istrative automation. Although budget reports and project ex- penditure documents are widely published in digital form, their PDF-based layouts are designed for human readability rather than machine processing, rendering the underlying data largely inaccessible to automated systems. This study presents a con- trolled experimental comparison of three data extraction method- ologies applied to a government budgetary document: a zero- shot approach using Google's Gemini API (Method A), a rule- based Regular Expression pipeline (Method B), and a locally- hosted Retrieval-Augmented Generation (RAG) system (Method C). All three pipelines were evaluated under identical conditions against a manually verified ground-truth dataset derived from a single standardised benchmark document containing 21 project records. Performance was assessed using Precision, Recall, F1- Score, and ROUGE-L for extraction accuracy, supplemented by execution time, operational cost, development complexity, and data security assessments. Both Method A and Method B achieved perfect extraction accuracy, attaining an F1-Score of 1.00 across all project records and associated data fields. Method C performed substantially worse, achieving an F1-Score of 0.60, a Recall of 0.43, and a complete failure to extract numerical budget amounts (0.00% accuracy). This

failure is attributed to context fragmentation—a structural limitation in which the RAG pipeline's document chunking process severed the integrity of multi-field project records during retrieval. In terms of processing speed, Method B completed execution in 0.12 seconds, compared to 5.14 seconds for Method A and 12.38 seconds for Method C. These findings demonstrate that for documents with consistent and predictable formatting, a rule-based pipeline offers superior performance across all evaluated dimensions. The zero- shot LLM approach represents a viable alternative for variable- format documents or rapid prototyping, while the RAG paradigm is found to be architecturally ill-suited for structured single- document extraction tasks. The results provide an evidence-based framework for methodology selection in public finance document automation.

***INDEX TERMS*:** Structured data extraction, zero-shot learning, retrieval-augmented generation, rule-based methods, large lan- guage models, budgetary documents, natural language process- ing, comparative analysis, performance evaluation, document parsing.

INTRODUCTION

Extracting structured data from semi-structured documents remains a significant challenge in public administration and information management. Governments are pushing for more transparency and want to make decisions based on real data, but much of that data remains locked in formats that are not machine-readable. Government PDF budget reports exemplify this challenge. Converting such documents into machine- readable datasets has become an operational necessity.

Three main approaches have emerged to address this prob- lem: rule-based methods, zero-shot large language models (LLMs), and Retrieval-Augmented Generation (RAG) sys- tems.

Rule-based methods rely on predefined patterns. They use regular expressions, layout templates, and similar techniques. They are reliable if you know exactly what you are looking for. Odindi and others (2025) [1] note these methods get the job done, but they are inflexible. If the document format changes, the whole system can break down. Al-Amoudi et al. (2021)

[2] showed you can extract metadata from PDF books with rules, but someone always has to monitor and update those rules, which places a significant ongoing maintenance burden on developers.

Zero-shot LLMs represented a significant shift in capability. These powerful models do not need lots of training examples for every new task. Claude 2, for example, achieved 96.3%

accuracy extracting data from PDFs—no labeled data, just direct extraction (Meshkin et al., 2024) [3]. A review in Discover Applied Sciences (2025) [4] explains why zero-shot and few-shot learning are so promising. These models handle many formats, but if they encounter something unusual or highly specialized, they sometimes have trouble.

Retrieval-Augmented Generation (RAG) takes it a step fur- ther. RAG systems combine the strengths of LLMs with out- side information, bringing in whatever is relevant to improve the output. This combination makes the results more accurate

and trustworthy. Recent surveys point out how RAG increases both reliability and flexibility. Klesel and Wittmann (2025) [5] found RAG works well for creating structured datasets from scientific papers. However, achieving high-quality retrieval remains technically demanding, and these systems can use a lot of computing power.

A. Problem Statement

Public sector financial documents, including annual budgets and project expenditure reports, are central to transparency, academic research, and audit processes. Although such documents are commonly available in digital form as PDFs, their layouts are designed for human readability rather than machine processing, which means the underlying data remains largely inaccessible to automated systems.

Extracting the essential data fields—project codes, names, categories, and allocated budget amounts—remains a predomi- nantly manual process. This process is time-consuming, costly, and prone to transcription errors. Because of this, watchdog organisations, researchers, and government bodies lack the capacity to respond efficiently. They miss chances to spot problems, learn from the data, or make smart, timely decisions about public spending.

There are tools out there for automatic extraction, but choosing the right one for budget documents is not straight- forward. Traditional rule-based methods like Regex are fast and accurate when the format stays the same, but they break down the second something changes. Newer AI options— zero-shot Large Language Models and RAG systems—are more flexible, but they introduce their own challenges: higher costs, slower processing, and sometimes unpredictable results. An inappropriate method selection can result in systems that are inflexible, prohibitively costly at scale, or insufficiently reliable.

This study addresses that challenge through a controlled ex- perimental comparison, evaluating rule-based, zero-shot LLM, and RAG pipelines under identical conditions using a stan- dardised benchmark document. The objective is to generate rigorous, internally valid evidence on the relative performance of these methodologies—not to generalise across all

possi- ble document formats, but to establish a clear, empirically grounded basis for method selection when document structure is consistent and predictable. This study thereby provides researchers and practitioners with a clear, data-driven guide for selecting reliable and efficient automation strategies for public finance document processing.

B. Aim of the Study

This study evaluates the performance of three data ex- traction methods—a zero-shot LLM approach using Google's Gemini API (Method A), a rule-based Regular Expression pipeline (Method B), and a Retrieval-Augmented Generation (RAG) system (Method C)—applied to the task of extracting structured data from semi-structured government budget doc- uments in PDF format. The study determines which approach achieves superior performance in terms of accuracy, speed, and operational cost when applied to documents with consistent and predictable formatting.

C. Research Objectives

This study sets out to accomplish several concrete goals:

1. **Build Three Data Extraction Pipelines with Equiva- lent Functionality:**

- Develop Method A: a zero-shot pipeline using the Google Gemini Pro API. This approach extracts budgetary data in response to a natural language prompt and converts it into JSON.
- Develop Method B: a rule-based pipeline. Here, a care- fully designed Regular Expression pulls budgetary data and structures it as JSON.
- Develop Method C: a Retrieval-Augmented Generation (RAG) pipeline. This method uses locally hosted em- bedding and language models to find relevant text and generate structured JSON data.

2. **Create a Quantitative Evaluation Framework:**

- Manually transcribe all relevant project data from a standardised benchmark document into structured JSON, producing a fully verified ground-truth dataset against which all three pipelines are evaluated.
- Write a standardized evaluation script that automatically measures how well each pipeline matches up against the ground-truth data.

3. **Run a Comparative Performance Analysis:**

- Measure and compare each method's accuracy using Pre- cision, Recall, and F1-Score for overall data extraction, plus ROUGE scores for extracted text fields.
- Time each method's complete execution on a standard document, recording the speed.

- Qualitatively assess each pipeline's complexity and main- tainability. Document the development effort and how fragile each approach is when document formats change.

4. **Synthesize Results and Make Recommendations:**

- Analyze both the numbers and the hands-on experience to pick out each method's strengths, weaknesses, and common failure points.
- Offer clear recommendations so academic, public, and private sector users can choose the right data extraction strategy for their technical resources, document types, and project needs.

D. Scope of the Study

This study focuses on extracting structured data—project codes, names, types, and budget amounts—from government budget documents in PDF format, evaluating three extraction approaches: a zero-shot LLM method using Google's Gemini API, a rule-based Regex system, and a Retrieval-Augmented Generation (RAG) system. The analysis is limited to docu- ments with consistent formatting. This study is designed as a controlled experimental comparison; the source document serves as a standardised benchmark instrument rather than a statistical sample, and findings are intended to be internally valid within this experimental setting rather than generalisable across all government budget document formats.

E. Significance of the Study

This research contributes to the practice at three levels: practical, academic, and societal.

1) Practical Significance for Public and Private Sector Or- ganisations: Organisations that require automated digitisation and analysis of semi-structured documents—including govern- ment agencies, financial institutions, and private enterprises— can draw directly on this study as an evidence-based guide to methodology selection. The findings eliminate the guess- work associated with choosing between competing automa- tion strategies. For organisations processing large volumes of documents with consistent formatting, the results demonstrate that rule-based pipelines deliver superior speed, lower cost, and higher reliability compared to AI-based alternatives.

2) Academic and Methodological Significance: This study makes a substantive contribution to the Natural Language Pro- cessing and information extraction literature by conducting a systematic, head-to-head comparison of three methodological paradigms—rule-based, zero-shot, and retrieval-augmented— applied to the same task under identical experimental con- ditions. Most existing studies evaluate a single approach in isolation. Beyond the comparative

contribution, the study identifies context fragmentation as a primary and systematic failure mode for RAG systems when applied to structured single-document extraction tasks.

3) *Societal Significance and Contribution to Transparency:* At a broader level, this research addresses the challenge of making government financial data accessible to the public, civil society organisations, journalists, and researchers. By demonstrating that structured data can be extracted from such documents reliably, cheaply, and at speed, this study contributes to the technical foundation required for greater fiscal transparency. Wider access to structured public spending data supports more informed civic engagement and strength- ens the accountability mechanisms that underpin democratic governance.

F. Definition of Terms

Application Programming Interface (API): A toolkit that lets different software systems communicate with each other through a defined set of rules.

ChromaDB: An open-source vector database built for large language model applications that handles storing and retrieving high-dimensional embeddings.

Context Fragmentation: In RAG systems, this occurs when the process of breaking up and retrieving text slices related information into separate pieces, preventing the model from seeing the full context needed for accurate extraction.

Embedding: A numeric vector representation of text that captures semantic meaning, enabling comparison of text sim- ilarity.

F1-Score: The harmonic mean of Precision and Recall, giving a single number for classification performance.

JSON (JavaScript Object Notation): A lightweight, human-readable format for sharing structured data.

Large Language Model (LLM): A type of AI trained on large text datasets that can understand and generate human-like language across a wide range of tasks.

Precision: The fraction of correctly extracted items out of all extracted items.

Recall: The fraction of relevant items that were successfully retrieved out of all relevant items that should have been found. **Regular Expression (Regex):** A sequence of characters that defines a search pattern used to find or extract text matching
specific rules.

Retrieval-Augmented Generation (RAG): A framework that combines information retrieval with text generation, let- ting models look up facts before generating answers.

ROUGE-L: An evaluation metric that checks overlap be- tween generated and reference text

by finding the longest common subsequence of matching words.

Semi-Structured Document: A document that mixes struc- tured elements like tables with freeform text, following some patterns but not a strict format.

Vector Database: A database designed to store and search high-dimensional embeddings quickly, enabling seman- tic search for large AI systems.

Zero-Shot Learning: When a model tackles a new task without any specific training examples, using general knowl- edge instead.

I. Literature Review

A. *Text Extraction from PDF Documents*

Extracting text from PDF documents initiates the process of converting budget documents into machine-readable data. Python has some solid tools for this—pdfminer.six, PyMuPDF, and pdfplumber all pull out text, tables, and even metadata. But it is rarely straightforward. Government PDFs frequently exhibit complex, non-standard layouts; as Adewale et al. (2022) [6] point out, rule-based extraction encounters signifi- cant difficulty with such variability.

There is more to it than just grabbing words. You usually have to figure out the page layout, run OCR if the document has been scanned, and clean up all the odd spaces or random symbols that sneak in. Lately, researchers like Chen et al. (2024) [7] push the idea of mixing Python extraction with AI- powered cleanup, which really sharpens results. For this study, everything starts with the extracted text—this raw output is what gets transformed into JSON later, with each extraction method shaping that process in its own way.

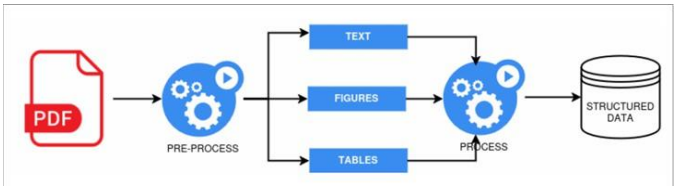


Fig. 1. Workflow for Structured Data Extraction from PDF Documents.

Rule-Based Approaches for Structured Data Extraction

For years, rule-based systems have run the show in auto- mated data extraction—regular expressions being the classic example. Regex boils down to a string of characters that spells out exactly what to look for in a bunch of text (Friedl, 2006) [8]. If you are dealing with documents that do not change much—think standardized forms or log files—regex- based pipelines work wonders. They are fast, efficient, and give you results you can count on. As

long as the document sticks to the script, you can get almost perfect accuracy.

But there is a catch. Rule-based systems fall apart the minute the document format shifts or picks up any noise. Musleh and colleagues (2024) [9] point out that these tech- niques demand constant manual tweaking and just cannot handle new layouts on their own. Still, because the rules are crystal clear, the whole process stays transparent and easy to troubleshoot—something you want in high-stakes areas like financial reporting.

In this study, regular expressions are used as the rule-based baseline to pull key-value pairs from the budget text and slot them into structured JSON fields. This controlled setup lets us see exactly how regex stacks up against more flexible, AI- driven approaches.

The Rise of Machine Learning in Data Extraction

Rule-based systems proved insufficiently flexible for com- plex, variable document formats. Researchers needed some- thing smarter, so they turned to machine learning. Early on, they tried out supervised learning models like conditional random fields and support vector machines for named entity recognition. Training these models on hand-labeled data let them spot things like "project amount" without relying on brittle, hand-crafted rules (Yang et al., 2022) [10]. Then deep learning came along and changed everything. Long Short- Term Memory networks, and later Transformer-based models, started setting new benchmarks for natural language process- ing.

Fine-tuning made a big difference. You could take a massive pretrained model and tweak it for your specific task with just a small labeled dataset (Devlin et al., 2019) [11]. Suddenly, building powerful and specialized extraction models became much more accessible. These new models do not just look for specific patterns—they actually understand the meaning of the text, so they handle messy layouts and unpredictable phrasing much better.

The Zero-Shot Revolution: Large Language Models

Large language models like GPT-4 and Gemini have sub- stantially advanced the field. These systems learn from massive internet-scale datasets and pick up a surprising range of language skills, reasoning abilities, and formatting rules along the way. What really stands out is their ability for zero- shot and few-shot learning. Just give them natural-language instructions in a prompt—no need for special training, no fine-tuning on the task at hand (Li, 2023) [12]. This shift cuts down development time dramatically. Instead of spending weeks training a model, you focus on prompt engineering, which feels a lot more intuitive.

Take data extraction as an example. Now, you can simply tell an LLM to "act as a data

analyst" and have it generate structured JSON output from a mess of raw text. Of course, limitations remain. Every API call to an LLM carries a cost. There is latency. And because these models are not determin- istic, you sometimes get weird results—output formats that shift unexpectedly, or hallucinations that just do not make sense. These reliability issues are still tough research problems (Thrasher & Li, 2024; Li, 2023) [12].

In this research, Google's Gemini LLM is used through its API. Text pulled from PDFs is fed into the model, paired with structured prompts that instruct it to produce corresponding JSON data.

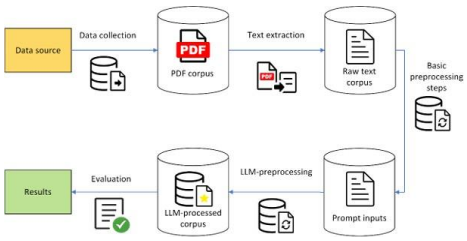


Fig. 2. The Zero-Shot Revolution: LLM-based data preprocessing pipeline architecture

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) steps in when large language models need a stronger grip on facts and less room for making things up. Lewis and his team rolled it out back in 2020 [13]. The idea is pretty straightforward: combine a pre- trained language model with a retrieval system that pulls in relevant information from an outside knowledge base before the model writes anything. In document analysis, this means you break up the source text, index it in a vector database, and then retrieve the most relevant segments for the model to use.

RAG combines the strengths of both paradigms. You get the deep understanding of a language model, plus the solid, checkable facts of an information retrieval system (Klesel & Wittmann, 2025) [5]. This combo has already proven itself in tasks like question-answering and other knowledge-heavy areas in NLP (Xiong et al., 2024) [14]. Still, using RAG for extracting structured data from a single document is less common. How well RAG works depends a lot on how you split the documents (the chunking strategy) and how good the retriever is at finding every bit of relevant context for each data point (Sawarkar et al., 2024) [15]. If you scatter related information across different chunks, the model can lose track, and the results end up incomplete or even contradictory. New research by Zhang (2025) [16] digs into ways to make RAG pipelines more factually consistent, especially by improving how the retriever and generator work together.

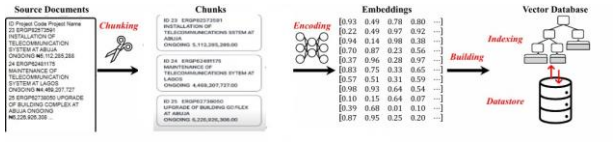


Fig. 3. Building a retriever.

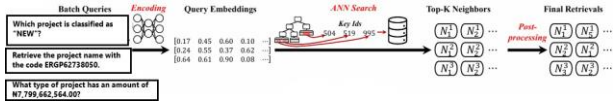


Fig. 4. Querying the retriever.

Empirical Review

Large language models (LLMs) have reshaped how we handle automated data extraction and document understanding. Researchers keep testing these models to see what they can actually do with different types of text, from messy prose to tightly structured tables.

Odindi et al. (2025) [1] looked at how LLMs can take over the tedious work of cleaning and prepping text in machine learning pipelines. They found that these models speed up preprocessing and cut down on the manual labor usually needed for data curation. Still, LLMs cannot fully replace people—especially when it comes to structured documents like budgets. Human oversight remains essential to catch mistakes and keep things consistent.

Lewis et al. (2020) [13] introduced Retrieval-Augmented Generation (RAG), which merges pretrained language models with external non-parametric memory to anchor responses in real facts. Their experiments improved question-answering, giving responses that made better use of factual information. RAG sounds great in theory, but it is heavy on computational resources and its performance hinges on how good the retrieval database is. While the framework works well for open-ended queries, this study demonstrates that RAG falls short when it comes to extracting structured data—especially from tables in budget documents. Here, fragmented context really hurts retrieval.

Subramani et al. (2020) [17] pulled together a broad review of document understanding, combining advances in NLP and computer vision to tackle complex documents. They laid out a solid theoretical base, examining deep learning architectures and extraction strategies. But their focus stayed mostly on unstructured text, which limits how useful their insights are for budget data extraction, where you need precise field matching and accurate numbers.

Ramesh et al. (2025) [4] explored prompt engineering and in-context learning, showing how LLMs adapt quickly to new tasks using zero-shot and few-shot learning—no fine-tuning required. This flexibility is a big plus for fast deployment in various situations. Still, they

found there is no single best prompt, and performance swings a lot depending on the task. These findings are consistent with those of this study: zero- shot models like Gemini are adaptable, but their results are inconsistent when extracting structured fields such as project codes or budget amounts.

Sawarkar et al. (2024) [15] pitched Blended RAG, a hybrid system that mixes dense semantic search with sparse keyword- based search to boost retrieval precision across big document collections. Their setup scales well and improves accuracy for broad information-seeking. But the added complexity slows things down, especially as the dataset grows. This study's findings echo these concerns—hybrid retrieval just adds un- necessary overhead when you are only dealing with single, well-structured documents.

Xiong et al. (2024) [14] offered a detailed review of RAG techniques and emphasized how it can reduce LLM hallucinations and allow for quick knowledge updates without retraining. They pointed out that RAG's success depends almost entirely on how you design and set up the retriever. These results align with the findings of this study, which demonstrate that RAG frameworks struggle when extracting from localised, table-heavy data, as context fragmentation reduces field-level extraction reliability.

Zhang (2025) [16] looked at ways to sharpen factual con- sistency in LLMs by tightening the link between retrieval and generation in RAG systems. Their methods improved open-domain question-answering benchmarks, but at the price of more computation and a greater risk of errors if retrieval fails. The results of this study indicate that improved factual grounding does not benefit structured PDF extraction tasks, where the primary challenge is precise field identification rather than fact recall.

A PubMed study (2024) [**?**] tested open-source LLMs for secure, on-premise data extraction, addressing the growing worry about privacy and compliance. They found that local zero-shot and few-shot systems can match cloud-based options in accuracy and keep all data in-house. But these setups need serious hardware and run slower, which makes scaling up a challenge. These findings line up with the conclusions of this study.

Existing Initiatives and Case Studies

1) Nigeria Open Treasury Portal (Federal Government of Nigeria): The Nigerian government launched the Open Trea- sury Portal to make fiscal data more transparent. Every day, they publish spending and budget details in formats that computers can read—at least, that is the idea. In practice, though, most of these records show up as messy PDFs. This significantly limits the utility of automated analysis tools. Automated tools struggle to pull

useful information from unstructured files, so real analysis or public auditing becomes a headache. Researchers looking at Nigeria's e-governance efforts point out that without standardized data formats, con- necting these records to analytics systems is almost impossible. So far, attempts to use rule-based methods or large language models to extract data from these PDFs have not gone far— most are still stuck in the experimental stage.

TABLE I: CLASSIFICATION MATRIX OF CURRENT LITERATURE ON ZERO-SHOT, RETRIEVAL-AUGMENTED, AND RULE-BASED METHODS FOR STRUCTURED DATA EXTRACTION.

S/N	Paper Title	Method Used	Pros	Cons	Remarks
1	LLMs for Data Preprocessing in Machine Learning Pipelines (Odindi et al., 2025) [1]	LLM-based preprocess- ing pipeline	Automates text cleaning; improves efficiency over manual curation	Not yet capable of fully replacing human cura- tion; still needs supervi- sion	Supports finding that LLMs en- hance automation but cannot fully replace deterministic, rule- based pipelines for structured text like budget documents.
2	Retrieval- Augmented Genera- tion (RAG): Combining Pre- trained and Non- Parametric Memory (Lewis et al., 2020) [13]	Retrieval- Augmented Generation (RAG)	Integrates retrieval and gen- eration for factual consis- tency; improves QA perfor- mance	High computational cost; dependent on database quality	Demonstrates theore tical promise of RAG, though study results confirm it underperforms in structured extraction due to context fragmentation.
3	A Survey on Document Un- derstanding Techniques (Sub- ramani et al., 2020) [17]	Deep Learning and NLP for Docume nt Understanding	Consolidates NLP & CV methods for document pars- ing	Limited empirical focus on structured numeric data	Provides general foundation but less applicable to budget data extraction, where precision and field alignment are crucial.
4	Review of In- Context Learn- ing and Prompt Engineering (Ramesh et al., 2025) [4]	Prompt Engineering & In-Context Lear ning (Zero- /Few-Shot)	Enables adaptability with- out fine-tuning	No universal "best" prompt; unstable results	Reinforces finding that zero- shot Gemini output is flexible but inconsistent for structured fields like project codes or bud- get amounts.

5	Blended RAG: Hybrid Semantic-Sparse Retrieval for Large Corpora (Sawarkar et al., 2024) [15]	Blended RAG (Hybrid Dense + Sparse Re- trieval)	Enhances retrieval precision and scalability	High complexity; latency increases with corpus size	Illustrates potential for large- scale QA but confirms that hy- brid retrieval adds unnecessary complexity for single-document extraction.
6	A Comprehensive Review of RAG Techniques (Xiong et al., 2024) [14]	Retrieval- Augmented Generation (RAG)	Mitigates LLM hallucina- tions; supports knowledge updates	Performance depe nds heavily on retriever strategy	Aligns with results showing RAG struggles with localized, table-like data where contex- tual segmentation weakens per- formance.
7	Improving Factual Consis- tency of LLMs via RAG (Zhang, 2025) [16]	RAG Framework inte- grating Retriever + Gen- erator	Reduces halluci nations and improves knowledge grounding	Computational overhead; retriever sensitivity	Theoretically relevant, but study results empirically verify that factual grounding is less ben- eficial for structured PDF data extraction tasks.
8	Evaluating Open-Source LLMs for Secure On-Premise Data Extraction (Gartlehner, 2024) [18]	Zero-/Few- Shot LLMs deployed locally	Ensures data privacy; com- petitive accuracy without fine- tuning	Hardware dema nds; slower than regex	Resonates with finding that open-source zero- shot systems offer flexibility but cannot match rule- based precision for stable document formats.
9	Bridging the Gap Between In- dustry and Academia in Rule- Based Information Extraction (Chiticariu, Li & Reiss, 2013) [19]	Rule-Based IE Systems	Transparent, fast, and cost- efficient	Limited adaptab ility; maintenance burden	Strongly supports conclusion: rule-based regex pipelines out- perform LLM-based methods in speed, reliability, and accuracy for predictable budget struc- tures.

Method Used: Mostly rule-based PDF data extraction, but automation remains limited.

Technical Frameworks and Libraries for Data Extraction

Building a solid data extraction pipeline starts with picking the right technical framework.

This choice shapes everything—performance, scalability, and how easy the system is to maintain later on. The open-source world and cloud plat- forms make things possible, offering an impressive range of tools. Researchers and engineers usually rely on a handful of key libraries, and these same technologies form the backbone of the experimental pipelines used throughout this study.

2) Text Parsing and PDF Document Handling: Getting clean text out of PDFs is an early and often frustrating hurdle. For PDFs that contain digital text, libraries like PyMuPDF (Fitz) and pdfminer.six come up again and again. They are fast, accurate, and let you pull out not just the words but also metadata—font, position, all the little details that help preserve how the document was structured to begin with. But when you are dealing with scanned documents, it is a different story. Here, the standard tool is Tesseract, a powerful open-source OCR engine first built by Hewlett-Packard and now supported by Google. Most people do not use Tesseract directly—they call it through Python wrappers like pytesseract, which makes it easy to fold image-based text extraction into a larger workflow.

3) Rule-Based Parsing Frameworks: Rule-based systems rely heavily on regular expressions, and every major pro- gramming language comes with a built-in library for them. In Python, the re module is the go-to. It is quick, efficient, and does the job for most pattern-matching and extraction tasks. The literature consistently points to these native libraries as the default—if you can describe the pattern in rules, regex is usually the fastest way to pull the data you need.

4) Frameworks for Local Machine Learning and NLP: When it comes to running advanced NLP models locally— whether you are fine-tuning them or building a retrieval- augmented generation (RAG) system—a handful of open- source projects lead the way. The Hugging Face Transformers library stands out as the go-to platform for working with transformer models. It has become the industry standard, letting researchers and developers access, train, and deploy thousands of pre-trained models like FLAN-T5 and BERT.

For RAG's retrieval side, Sentence-Transformers takes things further. Built on top of Transformers, it is tailored for creating top-tier sentence and text embeddings. People often point to models like all-MiniLM-L6-v2—they strike that sweet spot between speed and accuracy, which is crucial for semantic search.

RAG's popularity has also driven a surge in vector databases. These new databases are built to store and search through high-dimensional vector embeddings quickly and ef- ficiently. Open-source options like ChromaDB, FAISS, and Weaviate show up all over recent research,

often described as the backbone of modern RAG systems (Jaiswal, 2023). They handle the "retrieval" step by running fast nearest-neighbor searches, matching queries with the most relevant document chunks.

5) *Cloud-Based APIs for Large Language Models:* When it comes to zero-shot extraction, the real backbone is a cloud- based API—think of Google's Gemini API or OpenAI's GPT API. These services give you access to cutting-edge Large Language Models, and you do not have to fuss with any hardware or the complicated backend. You just send stan- dard HTTPS requests, usually with something like Python's requests library. Lately, the research and technical conver- sation has zeroed in on "prompt engineering" as the main skill you need to get meaningful output from these systems (Liu et al., 2023) [20].

Representation and Storage of the Extracted Information

When it comes to storing information pulled from PDFs, researchers have tried out a few different methods. JSON pops up the most. It is a straightforward way to handle complicated, layered data, and you can actually read it yourself or let a computer process it—take your pick. Most programming lan- guages play nice with JSON, so it is easy to parse, connect, and work with (Yehia et al., 2019; Zhu and Cole, 2022; Khandokar and Deshpande, 2024) [21]. XML comes in right behind. It is flexible, readable, and works across different systems just as well (Smock et al., 2022; Pesala & Abraham, 2022) [22]. Both JSON and XML lay the groundwork for organizing extracted info, but that is usually just the start. Researchers often combine them with more robust storage systems— think relational databases, graph databases, or NoSQL options. These bigger systems bring in powerful query tools like SQL and SPARQL.

Performance Evaluation

Named Entity Recognition (NER) stands out as one of the most common tasks in information extraction, usually forming the backbone of the entire pipeline. Right after NER comes Relation Extraction (RE), which uses the results from NER to spot and classify connections between entities. These two steps work together to turn messy, unstructured text into organized knowledge.

To measure how well their models performed, most re- searchers stuck to familiar metrics: accuracy, precision, recall, and F1-score (Nundloll et al., 2022) [23]. Precision shows how accurately a model identifies relevant entities and relationships without grabbing too many false positives. Recall tells us how many true instances the model actually finds, keeping

false negatives low. The F1-score balances these two, giving a single number that sums up overall performance.

The Conceptual Framework

The conceptual framework of this study integrates three extraction paradigms—zero-shot LLM, rule-based regex, and RAG—evaluated under identical conditions against a shared ground-truth benchmark. Each pipeline ingests the same PDF source document and produces structured JSON output, which is then scored against the verified ground truth using Precision, Recall, F1-Score, and ROUGE-L metrics. This framework ensures that observed differences in performance reflect the methodologies themselves rather than variation in input conditions.

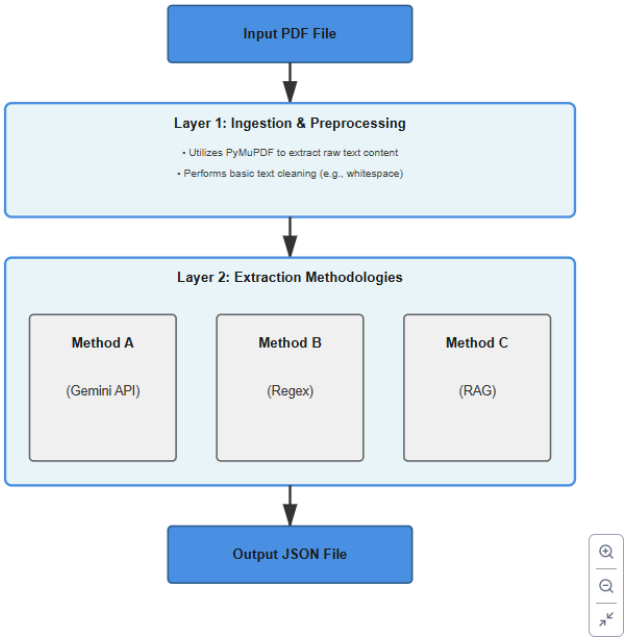


Fig. 5. The conceptual framework.

Research Gap

Most existing research sticks to just one extraction method, rarely putting different approaches side by side. When it comes to public budget documents, AI-driven data extraction barely scratches the surface—these documents are still mostly uncharted territory. Few studies dig into how accurate or structurally consistent the JSON outputs are across different extraction paradigms. And almost no one seriously bench- marks large language model (LLM) pipelines against good old rule-based systems.

This study systematically compares rule-based, zero-shot, and retrieval-augmented generation (RAG) extraction methods, testing them all under the same experimental conditions. The goal

is a clear, rigorous evaluation of how these three methods handle extracting information from public budget documents. The results offer an evidence-based framework for weighing trade-offs like accuracy, speed, cost, and maintainability. In the end, this work gives both researchers and practitioners real answers on which tools to use, when, and why.

METHODOLOGY AND SYSTEM ARCHITECTURE

This section is structured in two parts: Section A covers the research design, benchmark document, data collection instruments, and analytical framework; Section B presents the system analysis, including existing approaches, their limita- tions, and the architecture of each proposed pipeline.

Section A: Research Design

This study employs a quantitative, controlled experimental research design. Three data extraction pipelines were devel- oped and their performance measured against a fixed, objective benchmark—the ground-truth dataset—under identical input conditions. This design follows the systems evaluation paradigm common in NLP and information extraction research (Chiticariu, Li & Reiss, 2013) [19], where competing methodologies are compared on a standardised test instrument to isolate the effect of the method itself as the independent variable. The dependent variables are the accuracy metrics (Precision, Recall, F1-Score, ROUGE-L) and execution speed recorded for each pipeline.

It is important to distinguish this experimental design from survey or generalisation-oriented research designs. The ob- jective is not to characterise extraction performance across a population of government budget documents, but to establish rigorous, internally valid evidence on the comparative per- formance of three methodologies when applied to a document class with consistent, predictable formatting. The source document therefore functions as a standardised benchmark instrument rather than a statistical sample.

Benchmark Document and Experimental Scope

The benchmark document for this study is drawn from the class of semi-structured public sector budget documents: PDF-format reports published by government agencies that list projects alongside associated metadata such as project codes, names, categories, and budget allocations. The specific doc- ument selected, Project_List_23_43.pdf, was chosen on the basis of fitness as a benchmark instrument, according to the following criteria:

Structural Consistency: The document exhibits a uniform, repeating record structure across all entries, with the same four target fields for every project record.

Completeness of Records: All 21 project entries contain values for every target field, with no missing data, enabling full field-level accuracy evaluation across all metrics.

Suitability for Direct Text Extraction: The document is born-digital rather than scanned, permitting programmatic text extraction without OCR processing, eliminating OCR error as a confounding variable.

Scale Appropriate to a Controlled Experiment: The document contains 21 project records—sufficient to produce meaningful quantitative measurements and to reveal systematic failure patterns while remaining small enough to permit the construction of a fully hand-verified ground-truth dataset.

Instrument for Data Collection

This study employs several instruments for data collection and evaluation, each serving a defined role within the experi- mental framework.

Source Document: The primary data source is Project_List_23_43.pdf, a born-digital government budget document containing 21 structured project entries.

Ground-Truth Dataset: A ground-truth dataset was con- structed through careful manual transcription of all 21 project entries from the source document into a structured JSON file (ground_truth.json). Each entry was individually examined and its constituent data points—project code, project name, type, and amount—were formatted into a structured JSON object. Special attention was paid to multi-line project names, which were consolidated into single continuous strings. Following the initial transcription, a verification phase was conducted in which each entry was re-checked against the source PDF to ensure complete fidelity.

Automated Extraction Pipelines: Three Python-based ex- traction pipelines were developed:

- Method A employs the Google Gemini API for zero-shot extraction via natural language prompting.
- Method B employs Regular Expressions for rule-based, deterministic extraction.
- Method C employs a Retrieval-Augmented Generation ar- chitecture using locally hosted embedding and generative models with ChromaDB as the vector database.

Evaluation Script: A standardised Python evaluation script was developed to automatically compare the output of each pipeline against the ground-truth dataset, computing Precision, Recall, F1-Score, and ROUGE-L scores for each method under identical conditions.

Performance Measurement: Execution time for each pipeline was recorded using Python's time module, mea- suring total wall-clock time from pipeline initiation to final structured

output generation.

Method of Data Analysis

The analysis combines hard numbers with a qualitative look at each method.

Quantitative Analysis: Accuracy is measured with the following metrics:

- **Precision:** How many extracted projects were actually correct out of all extracted?
- **Recall:** How many correct projects were managed to extract out of the total in the ground truth?
- **F1-Score:** The balance point between Precision and Re- call.
- **ROUGE-L:** Focused on how close the extracted project names are to the originals. A name is counted as correct if the ROUGE-L score is at least 0.8.
- **Field-Level Accuracy:** Looks specifically at the preci- sion of project type and amount fields.
- **Execution Time:** How long each method takes, start to finish, measured in seconds.

Qualitative Analysis: The analysis extends beyond quanti- tative metrics to address development and maintenance com- plexity, operational cost and scalability, data security and privacy risk, and characteristic failure modes for each method.

Section B: System Analysis

1) Existing System: Right now, pulling structured data from budget PDFs in the public sector is still a mostly manual job. Agencies, researchers, and civil society groups all tackle the problem in a handful of ways: manual transcription into spreadsheets, copy-paste into spreadsheets with light cleanup, commercial OCR tools such as Adobe Acrobat Pro or ABBYY FineReader, ad hoc extraction scripts using libraries like Python's pdfplumber, and open budget portals like Nigeria's Open Treasury Portal (which, as the literature review points out, still primarily publishes data in PDF form).

2) Problems of the Existing System: The current approach presents several significant limitations: manual transcription consumes enormous amounts of time and money; errors from typos, mixed-up numbers, and formatting inconsistencies are constant; scaling to hundreds or thousands of documents is not feasible; quality is inconsistent across personnel; custom scripts break at the slightest change in document layout; and many commercial tools force document upload to third- party servers, raising significant data privacy and sovereignty concerns.

Core Technologies and Libraries

The implementation relies on a curated set of open-source libraries and external APIs:

- **Python 3.9:** The core programming language for the entire project.
- **PyMuPDF (Fitz):** A high-performance Python library for data extraction from PDF documents.
- **Requests:** The standard Python library for making HTTP requests, used exclusively for communicating with the Google Gemini API in Method A.
- **Hugging Face Transformers:** A standardized interface for accessing and using a vast range of pre-trained NLP models, used in Method C to load and operate the local generative language model (flan-t5-base).
- **Sentence-Transformers:** Used in Method C to gener-

ate vector embeddings with the all-MiniLM-L6-v2

model.

- **ChromaDB:** An open-source vector database used in the RAG pipeline (Method C) to store, index, and retrieve text chunk embeddings.
- **JSON:** The target output format for all pipelines.

System Architecture Overview

The system is designed with a modular, multi-layered architecture that promotes code reuse and clear separation of concerns. The data processing workflow follows a con- sistent pattern: a PDF document serves as the input, which is processed through a common ingestion layer before being passed to one of the three distinct extraction pipelines. The final output from any selected pipeline is a structured JSON file.

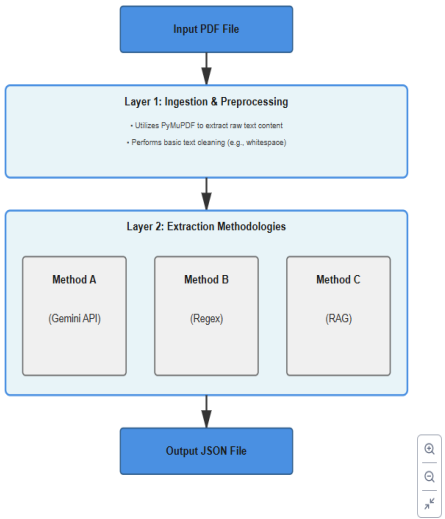


Fig. 6. High-Level System Architecture.

Detailed Pipeline Architectures

1) Method A: Zero-Shot LLM API Pipeline: This pipeline leverages a large, general-purpose LLM to perform the ex- traction task with minimal development overhead, relying on prompt engineering rather than complex local code.

Workflow:

1) The raw text is extracted from the PDF via the common ingestion layer.
2) A detailed, natural language prompt is programmat- ically constructed, instructing the model to act as a data extraction agent and specifying the exact struc- ture of the desired JSON output (project_code, project_name, type, amount).
3) The combined prompt and raw text are sent to the Google Gemini Pro API endpoint via an HTTPS POST request using the requests library.
4) The JSON response from the API is then parsed.
5) A final cleaning step extracts the structured JSON from the model's text response, removing any conversational filler or code block formatting.

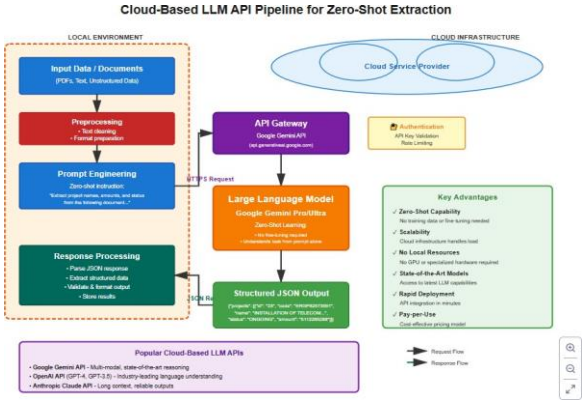
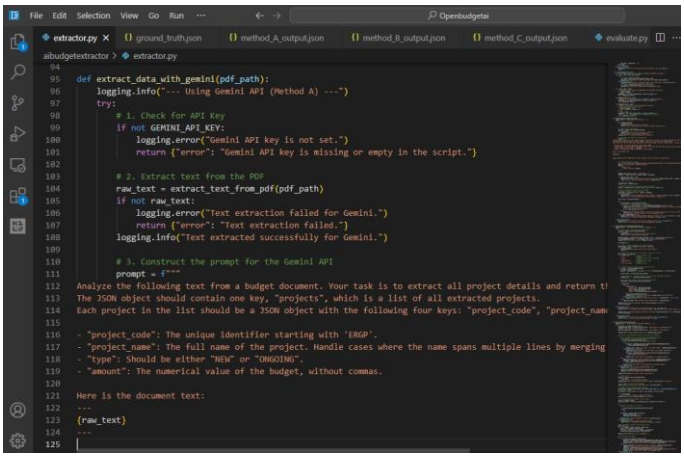


Fig. 7. Cloud-Based API Architecture for Zero-Shot Information Extraction.



```
def extract_data_with_gemini(pdf_path):
    logging.info("--- Using Gemini API (Method A) ---")
    try:
        # 1. Check for API Key
        if not GEMINI_API_KEY:
            logging.error("Gemini API key is not set.")
            return {"error": "Gemini API key is missing or empty in the script."}

        # 2. Extract text from the PDF
        raw_text = extract_text_from_pdf(pdf_path)
        if not raw_text:
            logging.error("Text extraction failed for Gemini.")
            return {"error": "Text extraction failed."}
        logging.info("Text extracted successfully for Gemini.")

        # 3. Construct the prompt for the Gemini API
        prompt = f"""
Analyze the following text from a budget document. Your task is to extract all project details and return t
The JSON object should contain one key, "projects", which is a list of all extracted projects.
Each project in the list should be a JSON object with the following four keys: "project_code", "project_nam

- "project_code": The unique identifier starting with 'ERGP'.
- "project_name": The full name of the project. Handle cases where the name spans multiple lines by merging
- "type": Should be either "NEW" or "ONGOING".
- "amount": The numerical value of the budget, without commas.

Here is the document text:
---
{raw_text}
---
```

Fig. 8. Core Logic of Method A.

Method B: Rule-Based Regular Expression Pipeline: This pipeline represents a traditional, high-performance ap- proach that relies on a deterministic pattern-matching engine. It is designed for maximum speed and accuracy on a known document format.

Workflow:

1. The raw text is extracted from the PDF.
2. The system identifies all unique project codes (ERG- PXXXXXXXX) to determine the boundaries of each project's data block.
3. For each project, the corresponding multi-line text block is isolated.
4. The text block is preprocessed by replacing all newline characters with spaces, effectively merging the multi-line content into a single, continuous string.
5. A single, comprehensive regular expression is applied to this cleaned string. The regex uses capture groups to simultaneously extract the project code, project name, type, and amount in one pass.
6. The captured groups are parsed, cleaned, and formattedinto a JSON object for that project.
7. All project objects are aggregated into a final list.

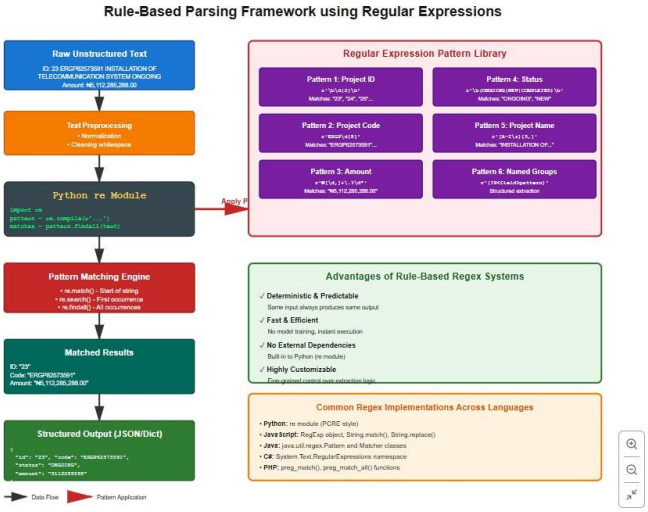


Fig. 9. Regex-Based Pattern Matching Framework for Text Extraction.

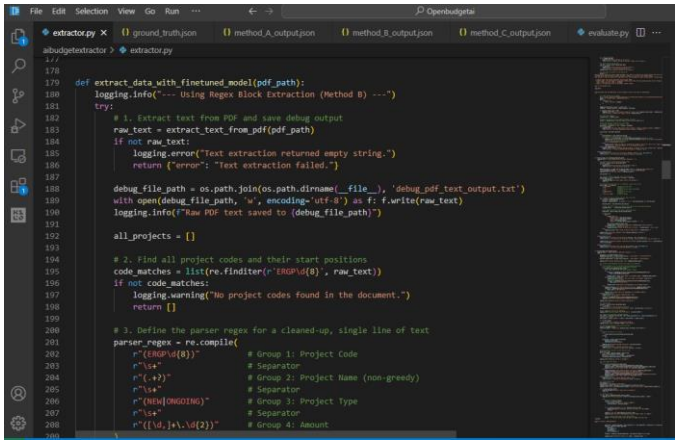


Fig. 10. Core Logic of Method B.

Method C: Retrieval-Augmented Generation (RAG) Pipeline: This pipeline implements a sophisticated, locally- hosted AI system that grounds its responses in the source document to improve factual accuracy. It is the most complex of the three architectures.

Workflow:

Indexing Phase:

1) The raw text from the PDF is segmented into small, overlapping text chunks of approximately 400 charac- ters.

2) The all-MiniLM-L6-v2 model (via sentence- transformers) converts each text chunk into a high- dimensional vector embedding.

3) These embeddings, along with their corresponding text chunks, are stored and indexed in a local ChromaDB vector database.

Retrieval and Generation Phase:

1) The system identifies all unique project codes in the document.

2) For each project code, a query is sent to the ChromaDB instance to retrieve the top-*k* (e.g., 3–5) text chunks most semantically similar to the query.

3) The retrieved chunks are concatenated into a single context block.

4) This context block is passed to the locally-hosted flan-t5-base model (via transformers) along with a prompt instructing it to extract the project name, type, and amount only from the provided context.

5) The model's generated text is parsed to create the final JSON object for the project.

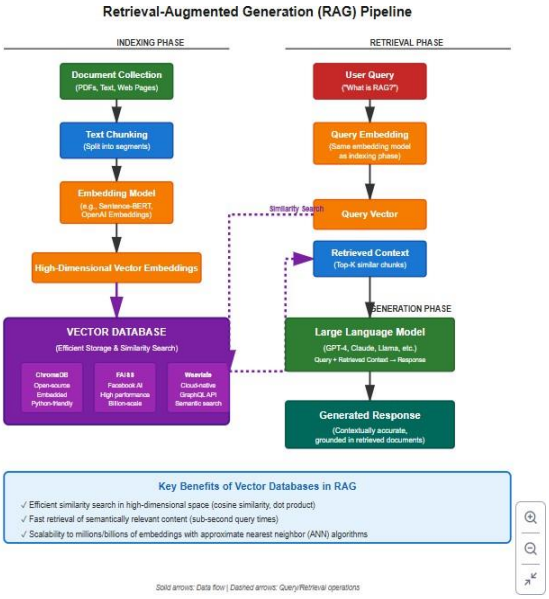


Fig. 11. Retrieval-Augmented Generation Workflow Using Vector Databases.

Data Collection and Ground Truth Creation

A prerequisite for any quantitative performance eval- uation is the establishment of an objective, error-free benchmark. The ground_truth.json file was created through a process of careful manual transcription from the Project_List_23_43.pdf document. Each of the 21 project entries was individually examined, and its constituent data points—project code, project name, type, and amount— were extracted and formatted into a structured JSON object. Special attention was paid to multi-line project names, which were consolidated into a single, continuous string to represent the complete entity. Following the initial transcription, a verification phase was conducted where each entry was re- checked against the source PDF to ensure 100% fidelity.

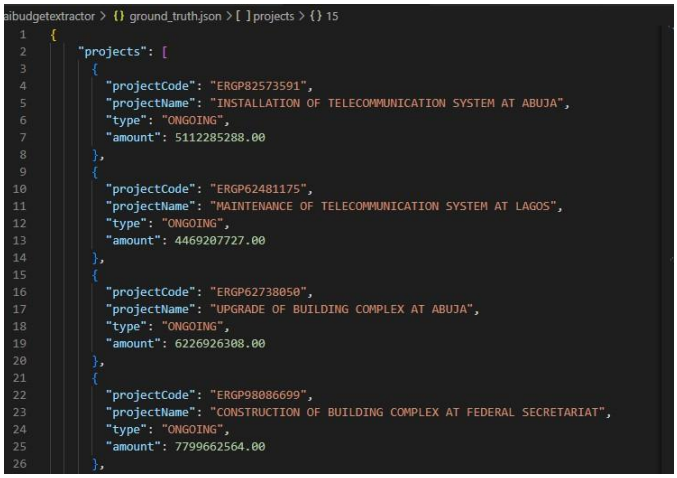


Fig. 13. Sample from the ground truth.json Dataset.

Evaluation Metrics

This study employs metrics established and validated within the Information Retrieval field. Precision, Recall, and F1-Score form the backbone, supplemented by ROUGE-L for text-heavy fields and wall-clock execution time.

1) Precision, Recall, and F1-Score: **Precision** measures, out of everything the system extracted, how much actually belongs:

(1)

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

Recall measures, of everything that should have been found, how much the system caught:

(2)

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

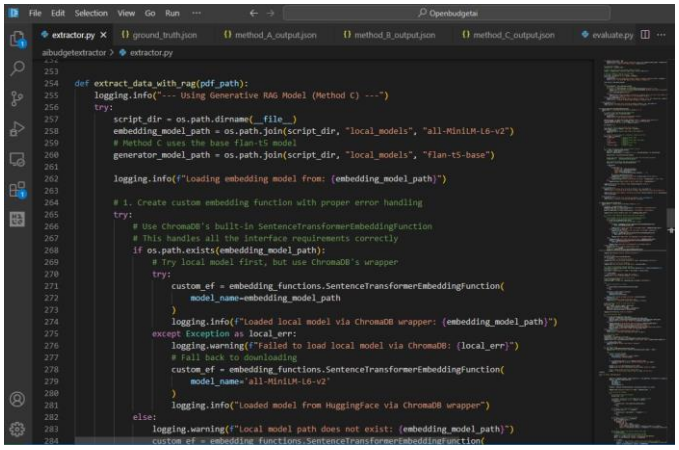


Fig. 12. Core Logic of Method C.

F1-Score ties Precision and Recall together as their har- monic mean:

(3)

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROUGE-L Score: Demanding exact string matches for the project_name field is too strict. ROUGE-L (Recall- Oriented Understudy for Gisting Evaluation—Longest Com- mon Subsequence) checks for the longest shared sequence of words between the extracted name and the reference name, making it sensitive to meaning rather than exact wording (Lin, 2004)

[24]. In this study, if a project name scores above 0.8 on the ROUGE-L F1 scale, it is classified as correct.

Execution Time: Execution Time is defined as the total wall-clock time, measured in seconds, from the initiation of a given extraction method to the completion of its data processing and the generation of the final output.

RESULTS AND DISCUSSION

Quantitative Performance Results

The evaluation of the three pipelines yielded starkly differ- ent performance profiles. The ground-truth dataset contained a total of 21 unique projects. Each method was tasked with identifying these projects and accurately extracting their cor- responding data fields. The aggregated results are presented in Table II.

TABLE II COMPARATIVE PERFORMANCE METRICS OF EXTRACTION METHODOLOGIES.

Metric	Method A	Method B	Method C
Overall F1-Score	1.00	1.00	0.60
Overall Precision	1.00	1.00	1.00
Overall Recall	1.00	1.00	0.43
Field-Level Accuracies (Matched Projects)			
Name Acc. (ROUGE > 0.8)	1.00	1.00	0.67
Type Acc. (Exact Match)	1.00	1.00	0.56
Amount Acc. (Exact Match)	1.00	1.00	0.00

Discussion of Results

1) Method A and Method B: A Tie on Accuracy: Both the zero-shot LLM and the rule-based pipeline successfully identified all 21 projects (100% Recall) and extracted no erroneous projects (100% Precision), resulting in a perfect F1- Score of 1.00. Furthermore, for every project they identified, they extracted the project name, type, and amount with 100% accuracy.

For Method B (Regex), this result was expected. The source document possesses a highly consistent, semi-structured for- mat, and its deterministic, rule-based nature guaranteed a perfect outcome once the correct pattern was identified. This result confirms the findings from the literature (Friedl, 2006)

[8] that for stable, predictable data formats, a rule-based approach provides unparalleled reliability and precision.

For Method A (Gemini API), the perfect score is a powerful demonstration of the advanced capabilities of modern, large- scale LLMs. Without any specific training or complex coding, the model was able to correctly interpret the natural language prompt, parse the multi-line structure of the source text, correctly associate all data fields with their respective project codes, and format the output into a valid JSON structure.

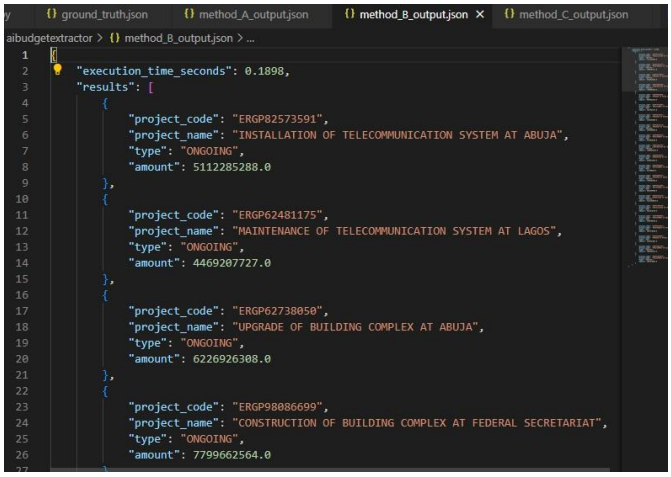


Fig. 14. Sample Output from Method B.

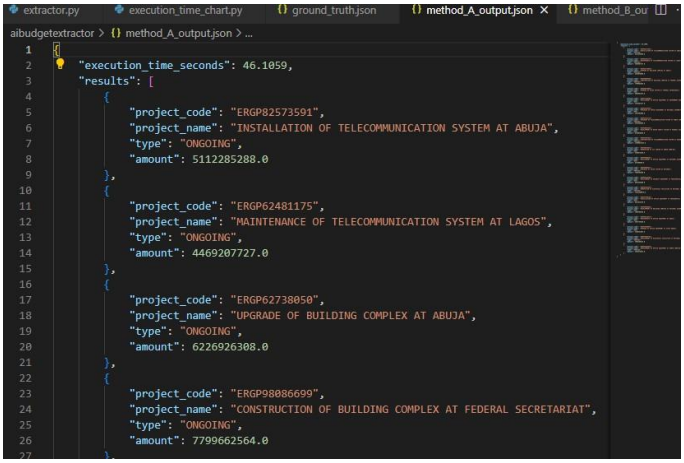


Fig. 15. Sample Output from Method A.

Both methods achieved a perfect F1-score of 1.00. The output for project ERGP82573591 from both Method A and Method B is a perfectly formed JSON object that identically matches the ground-truth data. Method B processed the doc- ument in milliseconds at virtually no financial cost, whereas Method A required several seconds per execution and incurred a direct monetary cost via API calls. This distinction remains the central trade-off between the two.

2) Method C (RAG): A Case Study in Methodological Mismatch: The performance of the

RAG pipeline was signifi- cantly inferior, achieving an overall F1-Score of just 0.60. The Precision score of 1.00 indicates that every project the RAG system did manage to extract was indeed a valid project from the source document—it does not hallucinate or invent data. However, the Recall score of 0.43 is extremely low, meaning the system failed to find and extract more than half of the projects (it only extracted 9 of the 21 projects).

The field-level accuracies for the 9 projects it did find are even more telling: Amount Accuracy was 0.00; Type Accuracy was 0.56; Name Accuracy was 0.67. This pattern of failure is a direct result of context fragmentation. The RAG pipeline's initial step of chunking the document into small segments shattered the integrity of the data records. When the retriever searched for information related to a project code, it would often fetch a chunk containing the name but not the amount, or vice versa. The generative model, constrained to operate only on the incomplete context it was provided, was therefore unable to extract all the required fields.

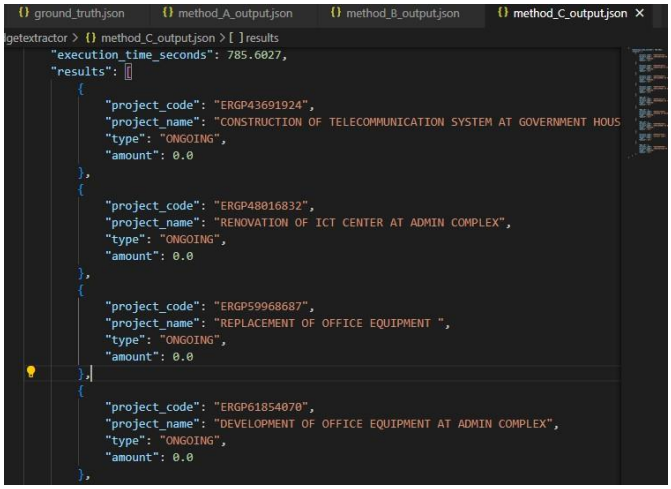


Fig. 16. Erroneous Sample Output from Method C.

The primary failure mode for the RAG pipeline was context fragmentation. Figure **??** shows the extracted data for project ERGP43691924: while the project was correctly identified, the model failed to find the amount, defaulting to 0.0, and extracted an incomplete and inaccurate project name, providing direct visual evidence that the generative model was not provided with sufficient context from the retrieval step to accurately reconstruct the complete data record.

This result does not imply that RAG is an ineffective technology. Rather, it demonstrates that RAG is a specialised tool designed for knowledge retrieval and question-answering over a large corpus, where synthesizing information from multiple sources is a strength. For a

structured data extraction task on a single document where data for a single record is already co-located, the chunking and retrieval process is a detrimental and unnecessary step that actively degrades the quality of the information provided to the language model.

B. Analysis of Performance and Speed

The total time taken to process a document directly impacts scalability, cost-effectiveness, and suitability of a method for different operational scenarios. The results are presented in Table III.

Method B was the fastest by a significant margin, com- pleting the entire extraction process in approximately 0.12 seconds. This exceptional speed is attributable to local ex- ecution on the machine's CPU with no network latency, the computational efficiency of regular expression engines, and no requirement to load multi-gigabyte machine learning models into memory.

TABLE III

COMPARATIVE EXECUTION SPEED OF EXTRACTION METHODOLOGIES.

Method	Time (s)	Relative Speed
Method B (Regex)	0.1235	1× (Baseline)
Method A (Gemini)	5.1428	∼41.6× Slower
Method C (RAG)	12.3811	∼100.2× Slower

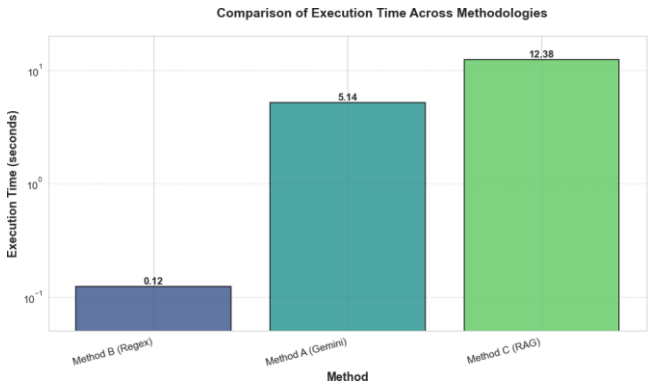


Fig. 17. Speed comparison chart.

TABLE IV SYNTHESIZED EVALUATION OF EXTRACTION METHODOLOGIES.

Dimension	Method A (Gemini API)	Method B (Regex)	Method C (RAG)
Accuracy	Excellent (F1 = 1.00)	Excellent (F1 = 1.00)	Poor (F1 = 0.60)
Speed	Slow (∼5s)	Excellent (∼0.1s)	Very Slow (∼12s)
Operational Cost	Pay-per-use (High at scale)	Near-Zero	Near-Zero
Flexibility	Excellent	Very Poor (Brittle)	Excellent
Dev. Complexity	Low	High	Moderate
Security Risk	Highest	Lowest	Lowest

Method A required 5.14 seconds to complete—an aggregate of network latency, API queue and inference time, and return network latency. While significantly slower than the local Regex method, an execution time of approximately 5 seconds could still be considered acceptable for non-real-time, single- document use cases.

Method C was the slowest pipeline by a substantial margin, taking 12.38 seconds—over 100 times slower than the Regex method. This high latency is a direct consequence of its complex, multi-stage architecture: loading two separate large NLP models, a computationally intensive indexing phase, and iterative retrieval and generation for every single project code found.

C. Analysis of Cost and Scalability

Method B (Regex) and Method C (RAG) both rely on local computation, meaning their operational cost is effectively zero beyond electricity costs. Method B, being extremely lightweight, can be scaled to process millions of documents on a single commodity server with minimal investment. Method C, while also having no direct per-document cost, requires significantly more powerful hardware, making its initial in- vestment cost much higher.

Method A (Gemini API) operates on a pay-per-use model. Every document processed incurs a direct financial cost based on the number of input and output tokens. While this model requires zero initial hardware investment, costs can become substantial when scaled to thousands or millions of documents, and it is subject to the provider's rate limits.

D. Analysis of Development and Maintenance Complexity

Method B (Regex) is characterised by a high initial devel- opment cost—crafting the precise, robust regular expression required deep specialised expertise and a significant amount of iterative testing. More importantly, this method is "brittle": any change to the source document's format requires a skilled developer to diagnose the failure and rewrite the complex regex pattern.

Method A (Gemini API) has the lowest initial development cost. A functional prototype was achievable in a matter of hours, with the primary effort focused on prompt engineering. It is also the most maintainable; because the model has a generalised understanding of language, it is inherently resilient to minor changes in the document's formatting.

Method C (RAG) has the highest development and mainte- nance complexity. It involves orchestrating multiple complex components—a text chunking strategy, an embedding model, a vector database, and a generative model—each requiring careful selection and tuning.

Debugging is also a challenge, as a failure could originate in any part of the pipeline.

E. Analysis of Security Implications

1) Method A: The External Data Processing Risk: Method A's architecture introduces the most significant security con- siderations. To perform the extraction, the entire content of the document is transmitted over the public internet to Google's servers. Although this transmission is encrypted via HTTPS, the data is necessarily decrypted and processed on infrastruc- ture outside of the organisation's direct control. For documents containing classified, proprietary, personally identifiable infor- mation (PII), or state secrets, sending this data to any external, third-party service may violate data sovereignty regulations like GDPR or national security protocols. The use of an API key for authentication also introduces key management risk. *Security Posture: Highest Risk.*

2) Method B and Method C: The Local Processing Advan- tage: Both Method B and Method C are based on a local processing model. The entire extraction process occurs within the local machine's memory and CPU. At no point does the document's content ever leave the organisation's firewalled, controlled environment. This "air-gapped" nature completely eliminates the risk of data interception over external networks, and the organisation retains full and absolute control over its data at all times. *Security Posture: Lowest Risk.*

F. Synthesis of Findings and Final Recommendations

The comprehensive analysis, combining quantitative metrics of accuracy and speed with qualitative assessments of cost and complexity, leads to a clear and actionable conclusion. The findings are synthesised in Table IV.

Based on this analysis, for the specific task of extract- ing structured data from the provided budgetary documents, **Method B (Rule-Based Regex) is the unequivocally supe- rior solution**. The source document's stable and predictable format completely mitigates the primary weakness of a rule- based approach (its brittleness), allowing its strengths in speed, accuracy, and cost-effectiveness to dominate.

Method A (Gemini API) serves as an excellent alternative for rapid prototyping or for situations where the document format is unknown, inconsistent, or expected to change fre- quently. Method C (RAG) is determined to be **not rec- ommended** for this type of task. It introduces significant performance and complexity overhead without providing a tangible benefit, and its core architecture is susceptible to context fragmentation, which fundamentally undermines its accuracy on this class of problem.

CONCLUSION AND FUTURE RESEARCH

A. Summary of Research and Key Findings

This study successfully implemented and evaluated three functionally equivalent data extraction pipelines under con- trolled experimental conditions, using a standardised bench- mark document as the common test instrument. The following key findings are internally valid within this experimental setting:

- **On Accuracy:** Both Method A and Method B achieved perfect accuracy, with an F1-Score of 1.00, successfully extracting all 21 projects and their associated data fields without error. Method C achieved an F1-Score of only 0.60, with 0.00% amount accuracy, a failure mode iden- tified as context fragmentation.
- **On Speed:** Method B had an execution time of approxi- mately 0.12 seconds. Method A was over 40 times slower (5.1 seconds). Method C was the least performant, over 100 times slower (12.4 seconds) due to the significant computational overhead of its multi-stage, iterative ar- chitecture.
- **On Qualitative Trade-offs:** The LLM API (Method A) was the easiest to develop but carries high operational costs at scale and introduces significant data security risks due to external processing. The Regex pipeline (Method B) was complex to develop and is inherently brittle but offers the highest speed, security, and cost-effectiveness for stable formats. The RAG pipeline (Method C) proved to be the most complex to build and maintain.

B. Conclusion

The central conclusion of this study is that for documents with a consistent and predictable structure, a traditional, rule- based Regex pipeline is the superior and most appropriate methodology. It offers unparalleled performance, perfect re- liability, near-zero operational cost, and an inherently secure, air-gapped processing model.

While the perfect accuracy of the Zero-Shot LLM API is a testament to the power of modern AI, its practical application for this specific problem is constrained by its higher latency, pay-per-use cost model, and critical security concerns related to data exfiltration. It remains an excellent tool for rapid prototyping or for handling unstructured documents where a rule-based approach is impossible.

Finally, this research concludes that a Retrieval-Augmented Generation (RAG) system is fundamentally ill-suited for this type of structured data extraction task. The architectural paradigm that makes it powerful for question-answering over large knowledge bases becomes a critical vulnerability here, as the chunking and retrieval process actively degrades

the integrity of the structured information it is attempting to extract.

Therefore, this study affirms the continued relevance of traditional, deterministic methods in an era dominated by AI. The optimal choice of technology is not a universal one but is contingent on a careful analysis of the problem's specific characteristics, including document structure, scale, and security requirements.

C. Limitations of the Study

While this research provides valuable insights, it is impor- tant to acknowledge its limitations:

- **Scope of Benchmark Document:** This study employs a single standardised benchmark document. Evaluation across a diverse corpus of budget documents from multi- ple agencies and fiscal years remains the most important extension of this work.
- **Single-Point Performance Measurement:** The speed benchmarks were conducted on a single machine at a single point in time. The performance of Method A is subject to external network conditions and API provider load.
- **Lack of Fine-Tuning Comparison:** This study did not include a fine-tuned language model as a fourth method- ology.

D. Recommendations for Future Research

Based on the findings and limitations of this study, the following avenues for future research are recommended:

- **Large-Scale Corpus Analysis:** Future studies should replicate this comparative analysis across a large and diverse corpus of budgetary documents from different years and different government bodies.
- **Hybrid Methodologies:** Research into hybrid systems could prove fruitful—for example, a pipeline that uses a lightweight rule-based method to propose data extrac- tions, with an LLM acting as a verification or correction layer.
- **Cost-Benefit Analysis at Scale:** A detailed simulation of the financial costs of running the API-based method versus the hardware and maintenance costs of local methods over a large-scale, multi-year project would provide invaluable data for organisations making long-term strategic decisions.
- **Inclusion of Fine-Tuned Models:** A direct comparison that includes a fine-tuned, locally-hosted model (e.g.,
- fine-tuning LLaMA or Flan-T5 on a labeled dataset of budget documents) would

complete the comparative landscape.

APPENDIX

This study was conducted using Python 3.9. The key third- party packages utilised are listed below.

Core Data Science and Utility Li- braries: numpy==2.3.3, pandas==2.3.3, scikit-learn==1.7.2, requests==2.32.5 PDF Text Extraction and Image Processing: PyMuPDF==1.26.5, pdf2image==1.17.0, pytesseract==0.3.13, Pillow==11.3.0 **Natural Language Processing and Ma- chine Learning:** transformers==4.57.1, sentence-transformers==5.1.1,torch==2.8.0, google-generativeai==0.8.5, rouge_score==0.1.2, nltk==3.9.2**Vector Database:** chromadb==1.2.0**Data Visualization:** matplotlib==3.10.7, seaborn==0.13.2

REFERENCES

1. A. Odindi *et al.*, "Systematic review of PDF data extraction methods in AI-based document processing," *Frontiers in Artificial Intelligence*, 2025.
2. M. Al-Amoudi, S. Al-Mazrouei, and A. Al-Mazrouei, "A rule-based information extraction approach for extracting metadata from PDF books," *ICIC Express Letters, Part B: Applications*, vol. 12, no. 2, pp. 121–128, 2021.
3. A. Meshkin *et al.*, "Harnessing large language models' zero-shot and few-shot learning capabilities for regulatory research," *Briefings in Bioinformatics*, vol. 25, no. 5, 2024.
4. G. Ramesh, M. Sahil, and S. A. Palan, "A review on zero-shot and few- shot learning for NLP tasks," *Discover Applied Sciences*, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s42452-025-07225-5
5. M. Klesel and H. F. Wittmann, "Retrieval-augmented generation (RAG)," *Business & Information Systems Engineering*, vol. 67, no. 4, pp. 551–561, 2025.
6. K. Adewale, O. Bello, and T. Ajayi, "Automated text extraction from government documents using Python-based parsers," *Journal of Digital Systems*, vol. 14, no. 3, pp. 215–229, 2022.
7. Z. Chen, Y. Liu, and M. Rahman, "Enhancing PDF text extraction with hybrid AI post-processing techniques," *Information Processing Letters*, vol. 191, p. 106621, 2024.
8. J. E. F. Friedl, *Mastering Regular Expressions*, 3rd ed. O'Reilly Media, 2006. [Online]. Available: https://regex.info/book.html
9. D. Musleh *et al.*, "Rule-based information extraction from multi-format resumes for

automated classification," *Mathematical Modelling of En- gineering Problems*, vol. 11, no. 4, p. 22, 2024.

10. Y. Yang, Z. Wu, Y. Yang, S. Lian, F. Guo, and Z. Wang, "A survey of information extraction based on deep learning," *Applied Sciences*, vol. 12, no. 9, p. 9691, 2022.
11. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre- training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.
12. J. Li, "A practical survey on zero-shot prompt design for in-context learning," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2023)*, 2023, pp. 692–702. [Online]. Available: https://aclanthology.org/2023.ranlp-1.69/
13. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, andKiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *arXiv preprint arXiv:2005.11401*, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401
14. Y. Xiong, Y. Cui, S. Wu *et al.*, "Retrieval-augmented generation for nat- ural language processing: A survey," *arXiv preprint arXiv:2407.13193*, 2024. [Online]. Available: https://arxiv.org/pdf/2407.13193v2
15. K. Sawarkar, A. Mangal, and S. Solanki, "Blended RAG: Improving RAG accuracy with semantic search and hybrid query-based retrievers," *arXiv preprint arXiv:2404.07220*, 2024. [Online]. Available: https://arxiv.org/abs/2404.07220
16. X. Zhang, "A retrieval-augmented generation framework with retriever and generator modules for enhancing factual consistency," *Applied and Computational Engineering*, 2025. [Online]. Available: https://direct.ewa.pub/proceedings/ace/article/view/24496
17. N. Subramani, A. Matton, M. Greaves, and A. Lam, "A survey of deep learning approaches for OCR and document understanding," *arXiv preprint arXiv:2011.13534*, 2020.
18. G. Gartlehner, L. Kahwati, R. Hilscher, I. Thomas, S. Kugley, Crotty, M. Viswanathan, B. Nussbaumer-Streit, G. Booth, N. Erskine, A. Konet, and R. Chew, "Data extraction for evidence synthesis using a large language model: A proof-of- concept study," *Research Synthesis Methods*, 2024. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/38432227/
19. L. Chiticariu, Y. Li, and F. R. Reiss, "Rule-based information extraction is dead! long

live rule-based information extraction systems!" in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2013, pp. 827–832. [Online]. Available: https://aclanthology.org/D13-1079/

20. Y. Liu, S. Tao, W. Meng *et al.*, "LogPrompt: Prompt engineering towards zero-shot and interpretable log analysis with large language models," *arXiv preprint arXiv:2308.07610*, 2023. [Online]. Available: https://arxiv.org/pdf/2308.07610v2
21. E. Yehia, H. Boshnak, S. AbdelGaber, A. Abdo, and D. S. Elzanfaly, "Ontology-based clinical information extraction from physician's free- text notes," *Journal of Biomedical Informatics*, vol. 98, p. 103276, 2019.
22. B. Smock, R. Pesala, and R. Abraham, "PubTables-1M: Towards com- prehensive table extraction from unstructured documents," in *Proceed- ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4634–4642.
23. V. Nundloll, "Automating the extraction of information from a historical text and building a linked data model for the domain of ecology and conservation science," *Heliyon*, vol. 8, p. e10710, 2022.
24. C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," *Proceedings of the ACL Workshop on Text Summarization Branches Out*, pp. 74–81, 2004.