Compositional AI-Service Pipeline to Generate Interactive Structured-Data from Scanned Images

Anthony Savidis * , Yannis Valsamakis , Theodoros Chalkidis , Stephanos Soultatos

Posted Date: 19 January 2026

doi: 10.20944/preprints202601.1279.v1

Keywords:

compositional AI; document intelligence; structured data extraction; NLP; OCR; image recognition

 doi:10.20944/preprints202601.1279.v1

Article

Compositional AI-Service Pipeline to Generate Interactive Structured-Data from Scanned Images

Anthony Savidis [1,2,*], Yannis Valsamakis [2], Theodoros Chalkidis [2] and Stephanos Soultatos [2]

[1] Computer Science Department, University of Crete, Greece

[2] Alpha Omega Zed SA, Research and Development Department, Greece

* Correspondence: savidis@csd.uoc.gr

Highlights

What are the main findings?

- Compositional AI pipeline design: We propose a modular architecture that decomposes document understanding into reusable AI services (OCR, image recognition, form modeling, semantic enrichment), allowing flexible orchestration and independent optimization.
- Interactive form extraction and normalization: We introduce AI-driven services for generating structured forms from raw text, incorporating synonym handling, value-type filters, and domain-specific constraints to ensure consistency and usability across heterogeneous sources.
- Bidirectional linkage between structured data and source images: We develop methods to maintain precise references between extracted entities and their visual origin within scanned documents, enabling interactive navigation, error analysis, and provenance preservation.
- Scalable and extensible framework for document intelligence: We demonstrate that the proposed compositional pipeline generalizes across domains (e.g., archival collections, enterprise repositories, scientific literature) and supports future extensions through the integration of additional specialized services.

What is the implication of the main finding?

- Enabling the automatic extraction of structured data from unstructured binary image data derived from scanned handwritten, typewritten or typeset documents.
- Supporting linkage of structured data to original areas of images for interactive browsing and search purposes.
- Promoting fully compositional processing pipelines for intelligent data extraction and processing.

Abstract

This paper presents a compositional Artificial Intelligence (AI) service pipeline for generating interactive structured data from raw scanned images. Unlike conventional document digitization approaches, which primarily emphasize optical character recognition (OCR) or static metadata extraction, the proposed framework adopts a modular architecture that decomposes the problem into specialized AI services and orchestrates them to achieve higher-level functionality. The pipeline integrates core services including OCR for text conversion, image recognition for embedded visual content, interactive form modelling for structured data and NLP for extraction of structured representations from raw text. The form models incorporate various rules like value-type filtering and domain-aware constraints, thereby enabling normalization and disambiguation across heterogeneous document sources. A key contribution is the interactive browser linking extracted structures back to the original scanned images, thus facilitating bidirectional navigation between unstructured input and structured content. This functionality enhances interpretability, supports error analysis, and preserves the provenance of extracted information. Furthermore, the compositional design allows each service to be independently optimized, replaced, or extended,

 doi:10.20944/preprints202601.1279.v1

ensuring scalability and adaptability to diverse application domains such as press archives, enterprise repositories and government documents.

Keywords: compositional AI; document intelligence; structured data extraction; NLP; OCR; image recognition

---

1. Introduction

1.1. Problem Definition

Unstructured data, massive data, linking, processing with flexible configurations.

1.2. Proposed Method

Compositional (programmable) processing pipeline, with NLP, OCR and IR services, including navigation and cross-reference.

1.3. Contributions

Automatic generation of data, configurable outputs and regeneration on-demand, interactive browsing and searching, substitutability of AI tools.

2. Related Work

Recent trends in document intelligence have shifted from monolithic OCR pipelines to modular, service-oriented designs enabling custom processing branches (Agarwal et al., 2024). Systems such as GROBID and the OCR-D framework exemplify this compositional approach by exposing components for segmentation, text recognition, and structured export, allowing flexible orchestration and independent optimization (Lopez et al., 2019; OCR-D Project, 2021).

Deep neural architectures that jointly leverage textual, layout, and visual cues have further advanced document understanding. Notable examples include LayoutLM, TILT, and Donut, which integrate multi-modal features to improve downstream tasks such as entity extraction and classification (Xu et al., 2019; Powalski et al., 2021; Kim et al., 2021). These models form the foundation for building our proposed compositional AI services for semantic enrichment, and, also, data, metadata and interactive form modeling.

Forms extraction (i.e. structured data) and normalization remain active research areas. Work on datasets such as FUNSD, SROIE, and DocVQA demonstrates the value of integrating spatial context and domain-specific rules for robust key-value extraction (Jaume et al., 2019; Mathew et al., 2020). Typically, scanned forms are spatially set structured data with varying styles. The same also holds for typical newspaper or magazine content, where visual arrangement indicates semantics (titles, annotations, primary messages, etc.).

Our approach extends these methods by combining synonym handling, value-type filters, and user-driven normalization to ensure semantic consistency across heterogeneous sources. On the underlying technology we rely entirely on existing AI services, while introducing upper layers for semantic organization and browsing.

Another relevant line of work concerns maintaining bidirectional linkage between extracted structured data and the source image. Standards such as ALTO and PAGE-XML support fine-grained provenance, enabling backtracking from recognized entities to their exact image coordinates (OCR-D Project, 2021). This forms the conceptual foundation for our interactive provenance-preserving architecture.

Finally, the emphasis on scalability and extensibility resonates with current efforts to build cross-domain document intelligence frameworks. Datasets like PubLayNet and recent transformer-based models highlight the benefits of pretraining on multi-domain corpora while maintaining modular

 doi:10.20944/preprints202601.1279.v1

retraining capabilities (Zhong et al., 2019). Our framework advances this direction by integrating service composability, interactive feedback loops, and domain-specific adaptability.

In summary, prior research has provided valuable building blocks for modular design, multi-modal representation, and provenance tracking. However, few systems unify these concepts into a fully composable, interactive, and provenance-preserving document intelligence pipeline. Our contribution fills this gap by offering an extensible AI architecture that couples semantic enrichment and form normalization with traceable visual / spatial grounding.

3. Architecture

Building on the conceptual foundation introduced above, the following section details the *compositional architecture* that underpins the proposed AI service pipeline. This architecture is designed around the principle of modularity—each component is developed as an independent yet interoperable AI service, contributing to the overall transformation of unstructured scanned images into structured, interactive data. By decoupling functionalities such as text extraction, image interpretation, form understanding, and semantic enrichment, the architecture promotes flexibility in both development and deployment. This compositional approach not only simplifies maintenance and optimization but also enables the seamless integration of emerging AI capabilities, as services, without disrupting the existing workflow.

Thus, we continue by categorizing and describing the individual components that constitute the pipeline. Each category corresponds to a distinct functional layer in an ad-hoc processing pipeline — from perceptual services that handle raw image and text recognition, to structural and semantic services that construct meaningful data representations and enforce domain-specific rules. Finally, integration and interaction services bind these components into a cohesive system, supporting visualization, navigation, and feedback loops that enhance accuracy and interpretability. Together, these elements form a scalable, extensible architecture capable of adapting to diverse document domains and evolving analytical needs.

3.1. Components

The components are enumerated below, providing an acronym and their brief role:

- OCR (Optical Character Recognition): scans text from images and returns recognized text, split in sentences, words and characters, together with geometric information of the respective bounding boxes in the original binary image
- NLP (Natural Language Processing): accepts text and applies various functions such as spell checking, data record extraction based on described record prompts (entity classes), together with information on the original text fragments (either corrected or from which content was extrapolated)
- IR (Image Recognition): scans an image to capture specific sub-images (first level containment) and returns all recognized occurrences as bounding boxes
- III or I^3 (Image in Image): scans an image to recognize sub-images within other images (second level containment) and returns all recognized occurrences as bounding boxes including the outer and inner ones
- UI4T (UI for Text): a non-AI component that accepts as input the outcome of OCR or NLP components, offering many interactive features including browsing, text search, bounding box reviewing, lens feature supporting juxtaposition of text with the respective parts in the original image, etc.
- UI4I (UI for Images): a non-AI component that accepts as input the outcome of IR or III components, enabling interactively review and even edit (scaling, stretching, selection and cropping) all recognized sub-images together with their placement in the original images

 doi:10.20944/preprints202601.1279.v1

Besides their specific functionality, all components share a common API in terms of: (a) the returned data, which include success scoring (confidence as percentage) and invocation cost (for paid services), and (b), provider metadata, which include company, service name and version information.

3.2. Interfaces

The detailed APIs (interfaces) per component, together with external services, and cataloging are provided under Figure 1.

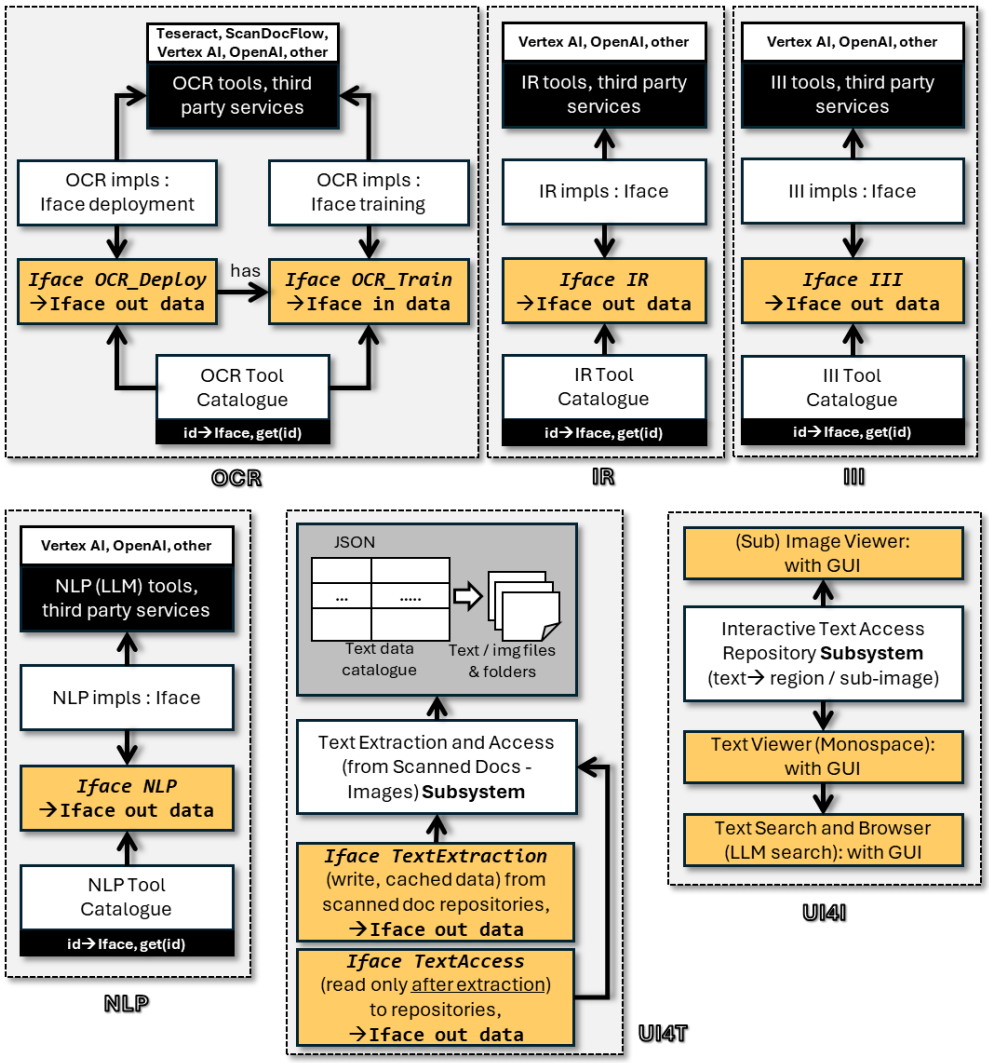


Figure 1. The OCR, IR, III, and NLP primary component interfaces and the UI4T and UI4I supporting subsystems.

As shown, all interfaces are stored in a catalogue with string keys so that the client application may concurrently deploy any of the supplied services. Additionally, only for the case of OCR, there is a distinction between the typical deployment version and the one enabling further training with custom datasets. The latter is offered by many OCR tools and it is particularly critical when custom handwritten text needs to be recognized, something that usually requires a custom training process before results of acceptable recognition accuracy can be obtained.

A detailed view of one of these interfaces is provided under Figure 2, showing the basic functionality that is required by an IR (Image Recognition) service. This specialization of AI services to mission-specific functionality allows handle not only multiple concurrent services for the same role / interface but also involving comprehensive AI services with distinct facets. The latter affects

 doi:10.20944/preprints202601.1279.v1

deployment costs since functionality-specific access to AI services is more cost effective compared to invocation of general purpose layers with higher-level prompts.

```
interface IR {
     getTools()                      : String[]
     getToolsMetadata()              : ToolMetadata[]
     getToolMetadata(tool: Tool)     : Expected<ToolMetadata, ToolNotFoundError>
     getToolSchema(tool: Tool)       : Expected<ToolSchema, ToolNotFoundError>
     newJob (jobData: JobData)       : Job
     newJobList (jobData: JobData[]): Job[]
     getJob (id: UUID)               : Expected<Job, JobNotFoundError>
     cancelJob (id: UUID)            : Void
}
type Tool = String
type ToolMetadata {
     name       : Tool
     active     : Boolean
     default    : Boolean
}
type ToolSchema = Map<String, Any>
type JobData {
     batch:          Batch
     tool:                Optional<Tool>
     responseCallback:    Optional<ResponseCallback>
}
type Job {
     id                  : UUID
     batch               : Batch
     tool                : Tool
     status              : JobStatus
     responseCallback    : Optional<ResponseCallback>
     results             : Optional<Expected<JobResult[], Error>>
}
type Batch           { imageFilePaths: String[] }
enum JobStatus       { in_progress,  completed, failed, cancelled }
enum JobResultStatus { running, success, failed }
type ResponseCallback = Function<Void(Job)>
type JobResult {
     fileKey    : String
     status     : JobResultStatus
     result     : Optional<Expected<String, Error>>
     extras     : Optional<Any>
}
```

Figure 2. The IR interface specification in a simple pseudo-language, showing job management, job state query and intermediate response collection.

3.3. Application

The main application involves massive processing and management of scanned published versions of magazines, newspapers, and corporate documents, while eventually delivering to the end-users facilities to search, browse and review content, enabling side-by-side viewing original and extracted document versions. In this context, it involves all previous components, including AI modules as well as the GUI ones. This architectural style is illustrated under Figure 3 with two alternative topologies, star and pipeline respectively. A view of the main application deployed to provide online versions of paper journals is provided under Figure 4.

 doi:10.20944/preprints202601.1279.v1

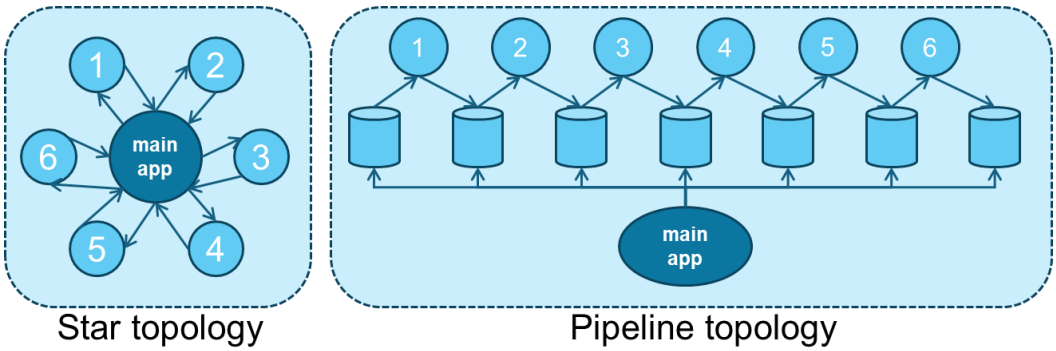
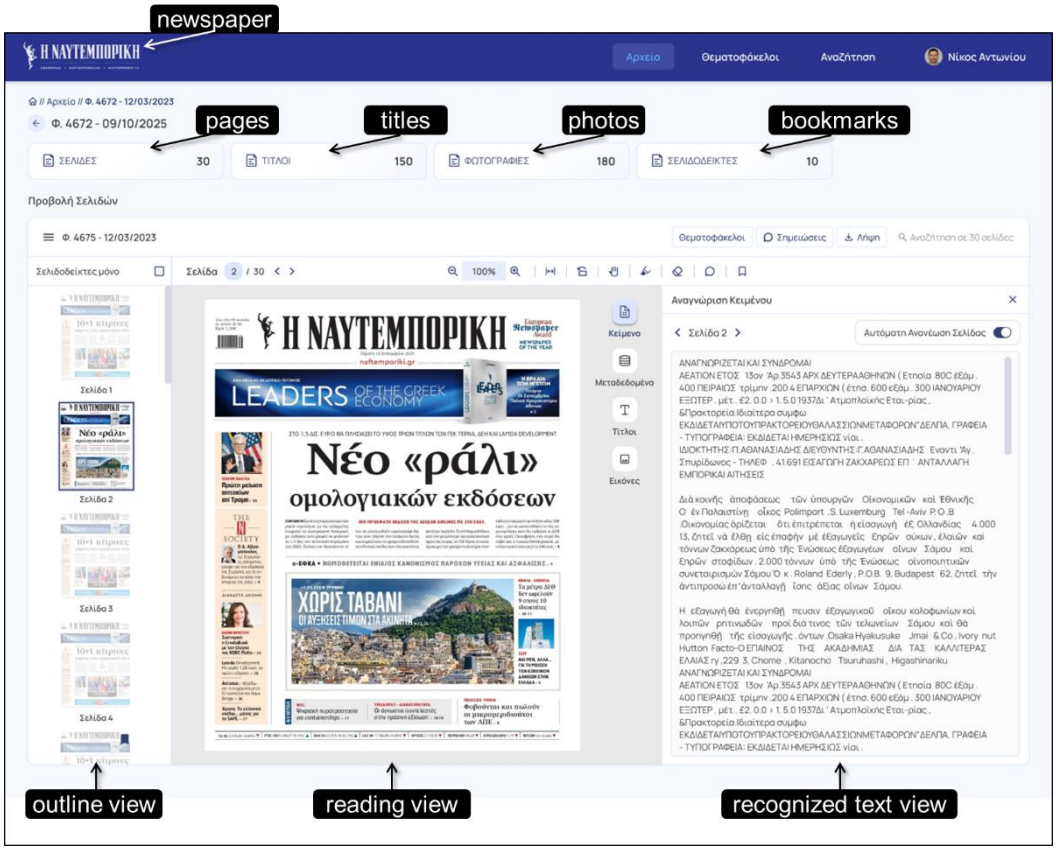


Figure 3. The main application, called Datacraft.AI™, involving sequential coordination of the various components, depicted with either a star topology (coordination-based view) or a pipelined one (phase-based view).

A shown under Figure 4 (two screenshots, first with text view and the second with all recognized images), using the GUI module UI4T (UI for Text), the application enables browsing into the scanned images (outline pane or reading view). Additionally, through the OCR and NLP modules the text is extracted (recognized) and then is provided in an interactive view with facilities for search, copy, highlighting and zooming.



 doi:10.20944/preprints202601.1279.v1

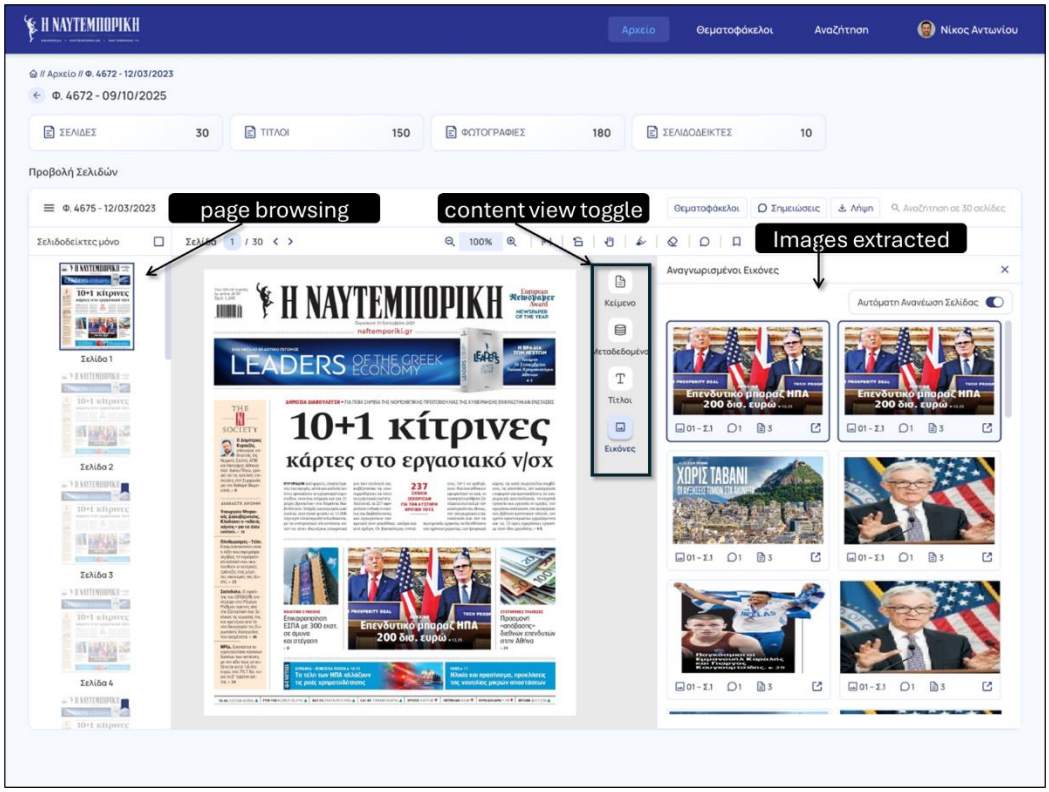


Figure 4. The actual main application deployed for building the centennial online version of the "ΝΕΥΤΕΜΠΟΡΙΚΗ" journal https://www.naftemporiki.gr/ (written "NAFTEMPORIKI" in Latin character set) for paper versions dating back to 50s. All content is a courtesy of "NAFTEMPORIKI".

5. Implementation

We provide below the primary details of our implementation, firstly enumerating the third party tools we have utilized for our case applications. It should be noted that this list is indicative, since by setting an interface-based deployment layer, there is no strict reliance on them, meaning the main application is unaware and completely independent of the actual underlying tools. Then, we outline the gluing coordination logic, which mainly performs typical data propagation by chaining producers and consumers together. Finally, we emphasize the importance of configuration, being a first-class concept in our interfaces, by requiring tools to expose configurable properties. Then, the GUI components of the main application provide them interactively to administrator end-users in order to fill-in with the desirable values supplied to the tools.

5.1. Tools

The implementation relies on a heterogeneous set of third-party AI tools, each integrated through a common interface layer corresponding to a specific functional role (OCR, NLP, IR, III). The selection of tools is indicative rather than prescriptive; the interface-based abstraction ensures that the core application remains agnostic to concrete providers, allowing tools to be added, replaced, or combined without code changes.

5.1.1. Tesseract (OCR)

Tesseract is an open-source OCR engine widely used for high-quality text recognition in scanned documents. It supports multiple languages, configurable recognition modes, and provides word- and character-level bounding boxes. In our pipeline, Tesseract is primarily employed for batch OCR processing of clean or semi-clean printed documents where deterministic behavior and zero invocation cost are required. Its primary capabilities for OCR role are enumerated below:

 doi:10.20944/preprints202601.1279.v1

- Text recognition from raster images
- Word, line, and character segmentation
- Bounding box extraction for provenance tracking
- Language-specific recognition models
- Confidence scoring per recognized segment

The deployment / integration of Tesseract as an OCR engine / module, following our interfaces, are depicted under Figure 5 (a few details omitted for clarity).

```
import { OCRService } from "./interfaces/OCR.js";
const ocr = new OCRService();
ocr.registerTool("tesseract", {
executablePath: "/usr/bin/tesseract",
languages: ["eng", "ell"],
oem: 1,
psm: 3
});
const job = ocr.newJob({
batch: { imageFilePaths: ["page1.png", "page2.png"] },
tool: "tesseract"
});
job.onComplete(result => {
result.forEach(r => {
console.log(r.fileKey, r.result.text);
});
});
```

Figure 5. Tesseract deployment as an OCR module.

5.1.2. ScanDocFlow (OCR)

ScanDocFlow is a commercial OCR service optimized for document flows involving degraded scans, historical documents, and mixed layouts. It offers enhanced preprocessing, adaptive thresholding, and layout-aware text extraction, making it suitable for archival material. Its primary capabilities for OCR role are enumerated below:

- Advanced image preprocessing
- Layout-aware OCR
- Noise and skew correction
- Batch processing with callbacks
- Fine-grained confidence metrics

The deployment / integration of ScanDocFlow as an OCR engine / module, following our interfaces, are depicted under Figure 6 (a few details omitted for clarity).

```
ocr.registerTool("scandocflow", {
endpoint: "https://api.scandocflow.ai/ocr",
apiKey: process.env.SCAN_DOC_FLOW_KEY,
layoutAnalysis: true
});
const job = ocr.newJob({
batch: { imageFilePaths: ["archive_1957.png"] },
tool: "scandocflow"
});
job.onComplete(job => console.log(job.results));
```

Figure 6. ScanDocFlow deployment as an OCR module.

5.1.3. Open AI / Chat GTP (OCR, NLP, IR, III)

 doi:10.20944/preprints202601.1279.v1

OpenAI's multimodal models are used as a general-purpose AI backend supporting OCR refinement, natural language extraction, image recognition, and image-in-image detection. These models are especially useful when semantic interpretation or flexible prompting is required beyond deterministic pipelines. Its multi-role capabilities in the context of our application are enumerated below:

- OCR correction and enrichment
- Named entity and record extraction (NLP)
- Visual object detection (IR)
- Nested image detection (III)
- Prompt-driven semantic reasoning

For the deployment, Open AI is actually a superset tool enabling to implement all specific modules, so we skip the details as it is typical interface deployment style, similar to the tools previously mentioned. However, the most important asset here is that we have turned prompting, being an essential part of AI service deployment, to a first-class concept via configurations, something that is discussed later.

5.1.4. Open CV (IR, III)

OpenCV is a classical computer vision library used for deterministic and high-performance image recognition tasks. In our system, it is primarily employed for detecting images, figures, and nested visual elements without semantic interpretation. Its primary capabilities for IR and III roles are enumerated below:

- Contour and region detection
- Feature-based image segmentation
- Hierarchical (nested) region detection
- Fast batch execution
- No external dependencies or API costs

5.1.5. Vertex AI / Gemini (OCR, NLP, IR, III)

Vertex AI (Gemini) is similar to Open AI and provides also multimodal AI services with strong layout understanding and enterprise-grade scalability. It is particularly suitable for large-scale deployments requiring reliability, quota management, and integrated monitoring. Its multi-role capabilities in the context of our application are enumerated below:

- Multimodal OCR and NLP
- Layout and object recognition
- Domain-adaptable extraction
- Scalable batch execution
- Enterprise security and governance

As with Open AI, we skip details of interface implementation and focus latter on the prompting resolution aspects, and how we turned them to configurable assets.

5.2. Gluing

The logic for gluing components together is effectively responsible for coordinating independently deployed services through their shared interfaces. Rather than binding the application to a single concrete implementation per role, all tools implementing the same interface are concurrently registered and discoverable at runtime. Additionally, as it will become clear in discussing our large case study, since tool deployment is not hard-coded, but we allow multiple tools to be chosen interchangeable for the same role, we enable administrator users to choose the actual tool in-the-fly and compare results. In other words, we shift composition to the end-users, something

 doi:10.20944/preprints202601.1279.v1

making the overall pipeline even more flexible. Each interface (OCR, NLP, IR, III) acts as a capability registry, exposing:

- available tools,
- metadata (default, active),
- schemas describing configurable parameters.

At execution time, the end-user or orchestration policy selects the tool dynamically, either explicitly or through defaults. This late-binding approach enables:

- side-by-side evaluation of alternative tools,
- failover and cost-aware selection,
- incremental replacement without redeployment.

In summary, each interface maintains its own tool catalog, book-keeping actual interface instance according to runtime service discovery and binding, while jobs are routed dynamically based on configuration and user choice.

5.3. Prompting

Prompting is an essential and open-ended aspect for AI-service deployment, since it treats service invocations as human-like directives. However, the more guided, precise and informative the prompt content is, the more accurate the functioning of the underlying service will be. In particular, our implementation treats prompts as first-class implementation concepts based on the following characteristics:

- Prompts become configurable assets
- Is keyed by interface (OCR, NLP, IR, III)
- Supports multiple AI providers (OpenAI, Gemini, etc.)
- Clearly separates functional guidelines per interface
- Is suitable both for runtime use

Under Figure 7 we exemplify configurable prompting. Again, the listing if very simplified, as we tend to use far larger and more structured prompts, however, the main idea is communicated.

```
{
"AI": {
"defaultProvider": "openai",
"OCR": {
"description":
     "Guidelines for optical character recognition and text grounding",
"providers": {
"openai": {
"model": "gpt-4.1",
```

```
"prompt":
"You are an OCR refinement engine. Given a scanned document image and raw OCR text, correct recognition errors while preserving the original layout. Return recognized text segmented into lines and words, each associated with bounding box coordinates in the original image. Do not infer missing text. Do not paraphrase. Preserve original spelling unless clearly erroneous. Provide confidence scores per segment."
```

```
},
"vertex-gemini": {
"model": "gemini-1.5-pro",
```

```
"prompt":
"Extract all visible text from the document image. Maintain spatial alignment between text segments and image regions. Output structured text with bounding boxes and confidence scores. Avoid semantic interpretation."
```

```
}
}
},
```

 doi:10.20944/preprints202601.1279.v1


```
"NLP": {
"description":
     "Guidelines for semantic extraction and structured data generation",
"providers": {
"openai": {
"model": "gpt-4.1",
"prompt":
"You are a document information extraction engine. Given OCR text with positional references, extract structured entities according to the provided entity schema. Normalize values using synonyms when appropriate. Apply value-type constraints strictly. Return structured records and maintain references to the originating text fragments and bounding boxes."
},
"vertex-gemini": {
"model": "gemini-1.5-pro",
"prompt":
"Analyze the document text and extract domain-specific entities and key-value records. Enforce schema constraints and data types. Preserve provenance by linking each extracted value to its original text span."
}
}
},
"IR": {
"description":
     "Guidelines for first-level image recognition",
"providers": {
"openai": {
"model": "gpt-4.1",
"prompt":
"You are an image recognition engine. Identify and locate all distinct visual elements embedded in the document image, such as photographs, charts, logos, or advertisements. Return bounding boxes for each detected element. Do not interpret content semantically."
},
"vertex-gemini": {
"model": "gemini-1.5-pro",
"prompt":
"Detect all non-text visual regions in the document image. Output bounding boxes and basic type labels (e.g., photo, illustration, chart). Do not infer meaning."
}
}
},
"III": {
"description":
     "Guidelines for image-in-image and nested visual detection",
"providers": {
"openai": {
"model": "gpt-4.1",
"prompt":
"You are an image-in-image detection engine. Detect visual elements contained within other images in the document (e.g., photos inside advertisements or figures within frames). Return hierarchical bounding boxes describing outer and inner containment relationships."
},
"vertex-gemini": {
"model": "gemini-1.5-pro",
"prompt":
"Identify nested visual regions within larger image regions. Preserve containment hierarchy and spatial relationships. Output structured bounding boxes."
```

 doi:10.20944/preprints202601.1279.v1


```
}
}
}
}
}
```

Figure 7. Configurable AI prompting keyed by interface names.

Based on this approach, in our compositional and pipelined architecture prompts are not hard-coded and they are interface-scoped configurations. Also, they can be swapped at runtime, tuned per domain, versioned independently of code, and exposed in the GUI property sheets. This reinforces that AI-service behavior is governed by the same configuration mechanisms as classical parameters (thresholds, models, languages), thus making prompt engineering an explicit and manageable system concern.

6. Case Study

6.1. Overview

Our case study, as previously mentioned, concerns a real-life project for a historic and very popular journal in Greece called "NAFTEMPORIKI". The project included initially digitization (scanning) of paper versions dating back to 1950s. This has resulted in almost 10.000 journal pages, with varying characteristics across time in terms of typesetting, layout, use of the Greek language, as well as noise coloring and preservation.

Then, through the presented AI-based system, we had to automate the extraction of structured information from all scanned versions, including content, titles, images, ads, version metadata, categories and classification, and provide interactive access through a web application.

6.2. Challenges

In this context, we have been faced with various challenges, which required in many cases simultaneous combined deployment of AI tools and techniques (i.e. not alternating but combining the outputs), as well as varying configuration approaches in service deployment, most of which are enumerated below:

- Typography ambiguities due to low visual quality
- Unordinary column layout disruptions
- Extended use of calligraphic text illustrations
- Ad images overlapping with news images
- Dual page images with embedded text content

The latter were addressed only through the compositional approach we have discussed, which enabled us to deploy AI services concurrently and in combination, not in a mutually exclusive manner, while being able to appropriately combine outputs. For instance, we observed that some OCR tools were behaving better than others in specific situations and vice versa. This way, we managed to link their deployment to specific data contexts, so that we could exploit their varying advantages to our maximum benefit.

6.3. Details

We provide various screenshots with actual processing results, discussing details on the way the tool pipeline has been flexibly applied to produce optimal outputs. In many cases we had to experiment with alternative combinations, testing the quality of the outcomes, before eventually crystallizing on specific service pairings. For example, we noticed that OCR followed by NLP application tended to always produce improved results, thus we adopted such added-value pairing.

 doi:10.20944/preprints202601.1279.v1

In particular, the latter is outlined under Figure 8, showing how text extraction from various journal columns is improved following NLP application. Such post-processing is unavoidable due to anticipated misses from OCR tools, considered overly normal in the Greek language once written text in formal and older dialects is handled. It should be noted that this tool is part of the administrator facilities offered in our central application, thus enabling the journal personnel to decide if the improved or the original versions are to be eventually adopted.

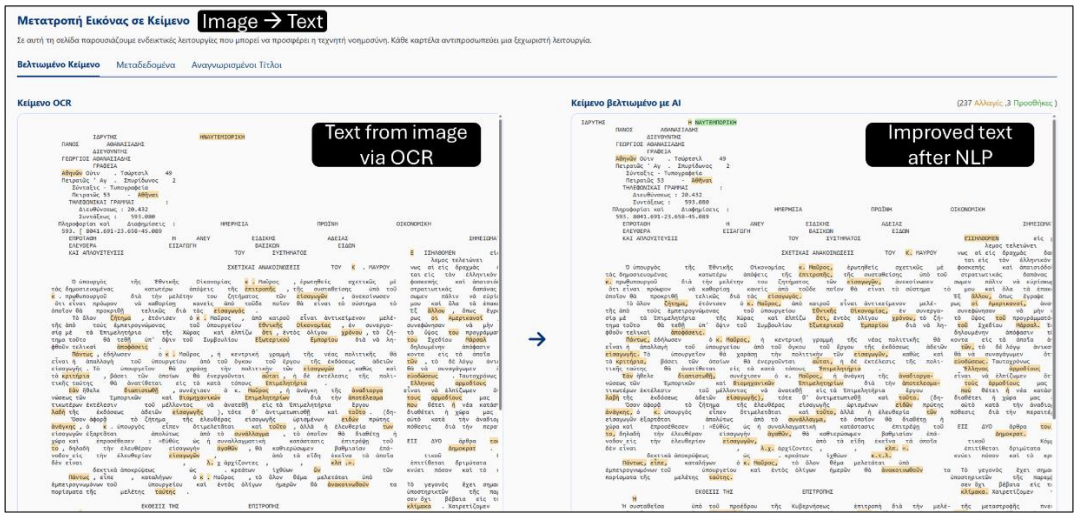


Figure 8. Text extraction and improvement from images, applying an OCR→NLP pipeline. As shown, since the OCR tool also generates respective raster bounding boxes per recognized word, we exploited to highlight all corrected or improved words after NLP tool deployment.

The application of IR modules (see Figure 9) suffered from various precision issues, appearing with different behaviors when comparing the alternative tools. The latter is not only an issue observed in old published journals, but also reappears in modern magazine illustrations. In particular, when embedded (first-level) images do not contain text, or the text is very limited, the recognition accuracy tends to be very high, while the output bounding rectangles are in the vast majority very precise and well-placed.

 doi:10.20944/preprints202601.1279.v1

Figure 9. Application of IR module (image recognition inside text, first level) together with raster bounding box extraction (shown with red outline and colored inside).

As shown under Figure 9, in cases images encompass large parts of text (two sub-images marked with yellow arrows), part of the embedded image is recognized, but the bounding boxes are not well captured. In the rest cases, even when there is text included, once the latter is not at the image boundaries, it is well assumed to be part of the picture. Unfortunately, we noted no particular configuration in the IR tools to rectify such common misses, thus we had to introduce special administration tools (GUI) to enable end-users manually fine-tune such imperfectly captured bounding rectangles.

Another important functionality of OCR tools that we thoroughly exploited to enable end-users browse journal contents is outlined under Figure 10 and concerns the capability to extrapolate detailed, and highly precise, bounding rectangles for the recognized text. This feature worked not only as a browsing feature, but has been proved very useful in the context of text search, since it allowed us to actually highlight the detected segments of text as they appear on the original scanned journal versions.

 doi:10.20944/preprints202601.1279.v1

Figure 10. Exploiting the output metadata of OCR tools to provide information on recognized text regarding bounding rectangles at the level of columns, paragraphs, sentences, words and individual characters.

Under Figure 11 the search facility is depicted (the UI4T module), showing how the pure text view is combined with the original scanned images to offer a dual interactive representation. The lens always follows the cursor position of the text view, while the focus can be tuned interactively (zoom in / out feature). Essentially, based on the OCR outcome, the internal text representation is enriched with geometric information (in raster coordinate system) to allow such multi-view search facilities. Similar functionality is also offered when browsing images (UI4I module), enabling view extracted images separately as thumbnails as well as within the original journal scanned images.

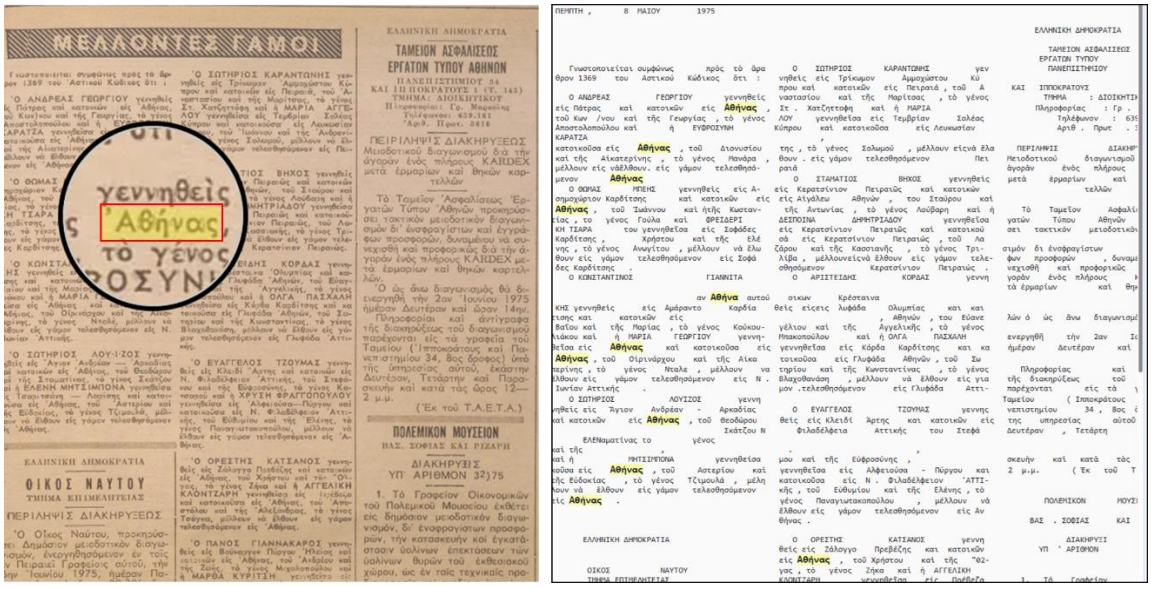


Figure 11. Detail of the interactive search facility with the lens tools, showing side-by-side: (left part) the original scanned image and the detected text area (highlighted bounding rectangle) with a magnified lens view, and

 doi:10.20944/preprints202601.1279.v1

(right part) the actual recognized text with all its occurrences, enabling move to the next (the lens will automatically readjust).

Finally, another powerful feature, depicted under Figure 12, concerned the ability to extrapolate geographic reference information directly from the recognized text content, via extra NLP processing and appropriate AI prompting. The latter enables us to provide interactive map views of all associated geographic areas (one or more), as actually mentioned in the original respective journal columns. This enable readers to directly perceive the related physical locations concerning new, events or announcements. This showed us that, following more deep content analysis, additional metadata can be extracted, such as custom classifications, enabling even more semantically rich visualizations in the future.

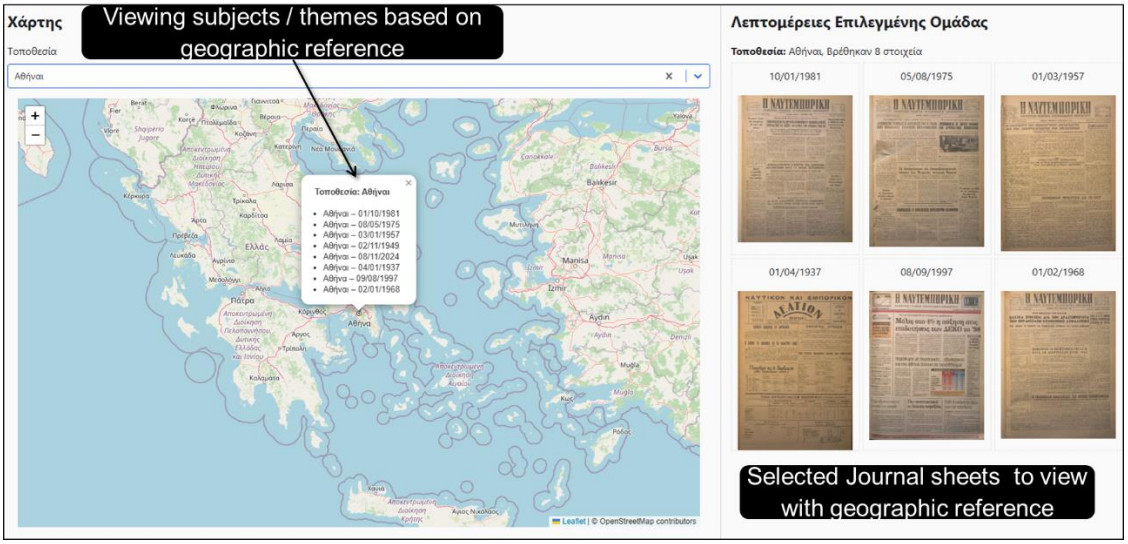


Figure 12. Extracting geographic reference information from Journal sheets enabling view on a map (left side) the spatial distribution of each separate sheet (right side). .

7. Discussion

7.1. Experience Reporting

We provide a brief account of our implementation experience in this funded research and development project, summarizing key aspects that worked well, failed either early or late, and what our future plans in extending and improving the overall infrastructure of the main application system.

7.1.1. What When Right

- First-class configurable prompting for AI services
- Managing multiple providers per AI-service interface
- Confidence scoring enabled runtime decision for alternative tools
- Configurable output redirection for data curation and improvements
- AI-cost computation as a first-class function enabling budget estimates
- Computation caching enabling to save AI-service costs
- Separation of AI-service interfaces from application GUI modules

7.1.2. What Went Wrong

- High expectations of AI tools regarding for OCR, IR and III service quality
- Continuous improvements on AI-service quality while we were developing

 doi:10.20944/preprints202601.1279.v1

- Missies in scanning which resulted in practical AI failures
- Significant quality variations in certain cases between different tools
- Non-uniform AI-service outputs requiring more work on output data interface

7.1.3. Improvements Underway

- Support more metadata categories to enable new custom consumers
- Making meta-data and consumers first-class add-ons
- Include form authoring GUI modules to extract arbitrary form data
- Graphical authoring for service pipelining and orchestration

7.2. Applying to Sensor-Oriented Applications

Although the proposed framework focuses on high-level document intelligence and compositional AI services, it is fundamentally grounded on data acquired through sensing devices and software pipelines (Chen et al., 2024). In particular, the primary input to the pipeline consists of rasterized document images produced by optical sensing systems, such as flatbed scanners, overhead book scanners, and camera-based document capture devices. These devices operate as optical sensors that transform physical paper artifacts—containing text, images, and layout information—into digital signals suitable for downstream processing.

7.2.1. Document Sensing Process (Scanners)

Modern document scanners and camera-based capture systems integrate multiple sensing components, including high-resolution CMOS or CCD image sensors, illumination modules, and optical lenses. These sensing elements directly affect the quality of the acquired data in terms of resolution, noise characteristics, color fidelity, geometric distortion, and illumination uniformity (Singh & Blumenstein, 2023). Variations in these sensing parameters have a measurable impact (Liu et al, 2024) on the performance of subsequent AI services, particularly OCR, image recognition (IR), and image-in-image (III) detection.

Additionally, we have noticed a strong impact of the human factor, affecting aspects like correct orientation and placement of the documents. In other words, there are many situations where human errors cannot be corrected even with the most advanced sensing technologies and AI processing. In this case, compliance to the digitization process protocols is of ultimate importance.

In the presented case study, the dataset was acquired using heterogeneous scanning equipment over several decades, ranging from early flatbed scanners to more recent high-resolution digitization systems. As a result, the scanned images exhibit sensor-induced artifacts such as blur, skew, uneven illumination, bleed-through, and color degradation. The proposed compositional pipeline is explicitly designed to tolerate such sensor variability by enabling the flexible combination and configuration of AI services that compensate for sensing imperfections.

7.2.2. Sensor Data Characteristics and AI-Service Interaction

From a sensing perspective, the scanned images constitute two-dimensional optical measurements sampled at discrete spatial resolutions. These measurements serve as the raw sensor signals upon which higher-level perception services operate. OCR modules can be viewed as virtual (software) sensors that transform optical measurements into symbolic textual representations, while IR and III components function as visual sensing layers that detect and localize non-textual visual elements, as in (Wang et al., 2025).

In other words, AI services, particularly the ones involved in our compositional system, constitute sensing layers which effectively propagate output to the next processing steps. The latter is also noted in (Batenburg & Sijbers, 2023).

The modular architecture allows sensor-aware adaptation of the pipeline. For example, OCR tools with stronger preprocessing capabilities can be dynamically selected for low-quality sensor

 doi:10.20944/preprints202601.1279.v1

data, while lighter-weight recognition services may be preferred for clean, high-resolution scans. Confidence scores produced by AI services can be interpreted as quality indicators of the underlying sensing process, enabling sensor-aware decision-making, quality assessment, and adaptive reprocessing.

7.2.3. Applications for Sensor-Driven Information Systems

The proposed framework is directly applicable to a wide range of sensor-driven information systems, including large-scale digitization infrastructures, digital libraries, archival preservation systems, and smart document repositories. In these applications, scanners and imaging devices act as front-end sensors that continuously generate data streams requiring automated interpretation, structuring, and indexing.

By preserving explicit links between extracted structured data and the original sensor measurements (image coordinates and bounding boxes), the system supports traceability, provenance tracking, and sensor-level error analysis. This capability is particularly important in sensing systems where reliability, transparency, and validation of extracted information are critical, such as governmental archives, cultural heritage digitization, and enterprise compliance systems.

Overall, the proposed compositional AI-service pipeline bridges low-level optical sensing with high-level semantic understanding, demonstrating how sensor-acquired visual data can be systematically transformed into interactive, structured representations through modular and extensible AI-based processing.

8. Conclusions

This work demonstrates that a fully compositional AI-service pipeline is a fundamental architectural requirement for robust document intelligence systems operating over heterogeneous, large-scale, and historically diverse scanned collections. By decomposing document understanding into well-defined interfaces and interoperable services, the proposed approach enables flexible orchestration, independent evolution, and fine-grained optimization of each processing stage. Crucially, the use of interfaces as stable abstraction boundaries allows multiple AI tools to coexist in parallel for the same functional role, making it possible to exploit their complementary strengths rather than relying on a single monolithic solution. The dynamic association between producers and consumers at runtime—rather than static, hard-coded bindings—proved essential in practice, enabling late binding, comparative evaluation, failover, and cost-aware selection of services and in many cases allow to overcome issue of imperfect outcomes from any of the tools. This dynamic, interface-driven composition directly supports scalability, long-term maintainability, and adaptability across domains with evolving document characteristics and quality constraints.

Equally important, the results highlight that modern AI services must be exploited not only through their exposed functions but also through their configuration space, which increasingly governs behavior, accuracy, and domain suitability. Prompting, model selection, thresholds, and provider-specific parameters emerge as first-class operational concerns and should therefore be elevated to first-class metadata within service interfaces. Treating configuration and prompting as explicit, versioned, and interface-scoped assets enables systematic governance, reproducibility, and interactive tuning by expert users, rather than ad-hoc, code-level adjustments. This perspective reframes AI services as configurable computational components whose semantics are co-defined by both function and configuration. Overall, the proposed pipeline illustrates that effective document intelligence systems must embrace compositionality at all levels—data, services, configurations, and metadata—thereby unlocking more transparent, extensible, and future-proof exploitation of the rapidly evolving AI ecosystem.

Author Contributions: Conceptualization, methodology, writing—original draft preparation, writing—review and editing, funding acquisition, resources , Anthony Savidis; Methodology, validation, investigation, supervision, project administration, Yannis Valsamakis; Software, investigation, visualization, data curation,

 doi:10.20944/preprints202601.1279.v1

Theodoros Chalkidis, Stephanos Soultatos; All authors have read and agreed to the published version of the manuscript.

Funding: "This research was funded by a private contract with "NAFTEMPORIKI".

References

1. Batenburg, K. J., Sijbers, J. (2023). Image acquisition and sensing challenges in large-scale digitization systems. Sensors, 23(6), 3124.
   Xu, Y., et al. (2019). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. arXiv:1912.13318.
2. Kim, G., et al. (2021). Donut: Document understanding transformer without OCR. arXiv:2111.15664.
3. Powalski, R., et al. (2021). Going Full-TILT Boogie on Document Understanding with Text-Image-Layout Transformer. arXiv:2102.09550.
4. Jaume, G., Ekenel, H. K., & Thiran, J. (2019). FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents. arXiv:1905.13538.
5. Liu, C., Zhang, Y., & Wang, J. (2024). Optical sensing and image quality assessment for document image analysis. Sensors, 24(3), 1457.
6. Chen, L., Xia, C., Zhao, Z., Fu, H., Chen, Y. (2024) AI-Driven Sensing Technology: Review, Sensors 2024, 24(10), 2958.
7. Singh, R., Pal, U., Blumenstein, M. (2023). Sensor-induced degradations in document image processing: A survey. IEEE Sensors Journal, 23(18), 21045–21060.
8. Mathew, M., Karatzas, D., Manmatha, R., & Jawahar, C. (2020). DocVQA: A Dataset for VQA on Document Images. arXiv:2007.00398.
9. Agarwal, A., Laxmichand Patel, H., Pattnayak, P., Panda, S., Kumar, P., Kumar, T. (2024). Enhancing Document AI Data Generation Through Graph-Based Synthetic Layouts. CoRR abs/2412.03590
10. Zhong, X., et al. (2019). PubLayNet: Largest dataset ever for document layout analysis. arXiv:1908.07836.
11. Lopez, P., et al. (2019). GROBID: Machine learning for extracting and structuring documents. Project documentation.
12. Wang, B., Wu, B., Li, W., Fang, M., Liang, Y., Huang, Z., Wang, H., Huang, J., Chen, L., Chu, W., Qi, Y. (2025). Infinity Parser: Layout Aware Reinforcement Learning for Scanned Document Parsing. CoRR abs/2506.03197
13. OCR-D Project. (2021). OCR-D Framework and Provenance Model. ALTO/PAGE-XML Documentation.