## **Informe de Prueba de Concepto: Evaluación de la Herramienta ExtractThinker para Inteligencia Documental**

**- Resumen Ejecutivo**

Esta prueba de concepto (POC) se llevó a cabo con el objetivo de evaluar la herramienta de software ExtractThinker como una solución para la inteligencia documental empresarial, enfocándose en su capacidad para extraer y clasificar datos estructurados de diversos formatos de documentos dentro de un entorno de servicios en la nube de Amazon Web Services (AWS). La metodología incluyó la implementación de ExtractThinker en AWS, la realización de pruebas exhaustivas para evaluar su rendimiento, precisión y costos asociados, y el análisis comparativo con otras tecnologías homologas disponibles en el mercado. Los hallazgos clave indican que ExtractThinker ofrece una flexibilidad significativa en la integración con diferentes modelos de lenguaje grande (LLMs) y motores de reconocimiento óptico de caracteres (OCR), facilitando la creación de flujos de trabajo de procesamiento de documentos personalizados. Sin embargo, se identificaron ciertas limitaciones en cuanto a la madurez de la plataforma y la posible necesidad de un mayor desarrollo personalizado en comparación con soluciones de inteligencia documental gestionadas en la nube. En general, ExtractThinker presenta una alternativa prometedora para organizaciones que buscan un mayor control y agilidad en sus procesos de inteligencia documental, aunque se recomienda una evaluación más profunda en escenarios de producción a gran escala.

**- 1. Introducción**

\* \*\*1.1. Objetivo de la Prueba de Concepto\*\*  
 El objetivo principal de esta prueba de concepto fue evaluar la viabilidad y el potencial de la herramienta ExtractThinker como una solución para la inteligencia documental dentro de una empresa. Esto implicó un análisis detallado de sus características, rendimiento, capacidades de integración y rentabilidad al ser implementada en el entorno de servicios en la nube de AWS. Se buscó específicamente determinar si ExtractThinker podría abordar los desafíos comunes en el procesamiento de documentos, como la extracción precisa de datos, la clasificación eficiente de documentos y la optimización de los flujos de trabajo, tal como se describe en la documentación proporcionada en el repositorio de GitHub.[1, 2] Un objetivo implícito fue investigar si ExtractThinker ofrece una alternativa viable y potencialmente más flexible y rentable en comparación con las soluciones propietarias que a menudo generan dependencia de un proveedor y conllevan altos costos de desarrollo.[3]  
  
\* \*\*1.2. Alcance de la Evaluación\*\*  
 El alcance de esta POC se centró en la evaluación de las funcionalidades principales de ExtractThinker, incluyendo la extracción de datos estructurados, la clasificación de documentos y la división de documentos extensos. Se utilizaron diversos tipos de documentos para las pruebas, tales como facturas, licencias de conducir y documentos de texto, con el fin de evaluar la versatilidad de la herramienta. La implementación se realizó utilizando servicios específicos de AWS, principalmente Amazon Elastic Compute Cloud (EC2) para el alojamiento de la aplicación y Amazon Simple Storage Service (S3) para el almacenamiento de los documentos. Los criterios de evaluación definidos incluyeron la precisión de la extracción de datos, el rendimiento en términos de tiempo de procesamiento, la facilidad de integración con AWS y los costos asociados a la infraestructura y al uso de la herramienta. Se excluyeron del alcance de esta POC aspectos como la implementación en un entorno de producción a gran escala y la integración con todos los sistemas empresariales posibles, enfocándose en demostrar la funcionalidad central y la viabilidad dentro de un entorno controlado de AWS.

**- 2. Descripción de la Tecnología**

\* \*\*2.1. Características, Arquitectura y Beneficios de ExtractThinker\*\*  
 ExtractThinker se presenta como una biblioteca de inteligencia documental diseñada para modelos de lenguaje grande (LLMs), ofreciendo una interacción similar a un mapeador objeto-relacional (ORM) para facilitar flujos de trabajo de procesamiento de documentos flexibles y potentes.[1, 3, 4, 5] Su arquitectura modular, inspirada en el ecosistema de LangChain, se compone de varios componentes clave.[1] Los \*\*Document Loaders\*\* son responsables de cargar documentos de diversas fuentes y realizar el preprocesamiento necesario. ExtractThinker soporta una amplia gama de cargadores, incluyendo Tesseract OCR, Azure Form Recognizer, AWS Textract, Google Document AI, PDF Plumber y PyPDF.[1, 2, 3, 4, 5, 6] Los \*\*Extractors\*\* orquestan la interacción entre los Document Loaders y los LLMs para extraer datos estructurados de los documentos, utilizando modelos Pydantic para definir los contratos de extracción.[1, 2, 3, 4, 5] Los \*\*Splitters\*\* implementan estrategias para dividir documentos grandes en fragmentos más manejables.[1] Los \*\*Contracts\*\*, definidos mediante modelos Pydantic, especifican la estructura esperada de los datos extraídos. El componente de \*\*Classifications\*\* permite categorizar documentos o secciones de documentos, y los \*\*Processes\*\* gestionan el flujo de trabajo general de carga, clasificación, división y extracción de datos.[1]  
  
 ExtractThinker ofrece varios beneficios operacionales significativos. Simplifica el procesamiento de documentos al proporcionar un enfoque estructurado para la extracción de datos, mejora la precisión mediante el uso de LLMs, facilita el desarrollo con APIs intuitivas y cuenta con el respaldo de una comunidad activa.[1, 2] Su interacción de estilo ORM simplifica la forma en que los desarrolladores interactúan y extraen datos de los documentos, abstrayendo la complejidad subyacente de la interacción con OCR y LLMs. Además, ExtractThinker se especializa en el procesamiento inteligente de documentos (IDP), diferenciándose de frameworks más generales como LangChain.[1, 2, 3, 5, 7, 8] Esta especialización le permite ofrecer características específicas para el procesamiento de documentos, como la división de documentos extensos y la clasificación avanzada.[2]  
  
\* \*\*2.2. Comparación con Otras Tecnologías\*\*  
 Al comparar ExtractThinker con servicios de IDP basados en la nube como Azure Document Intelligence (anteriormente Form Recognizer) [3, 9, 10, 11, 12, 13, 14, 15] y AWS Textract [2, 3, 4, 6, 9, 10, 11, 12, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29], se observan ventajas y desventajas distintivas. Una ventaja potencial de ExtractThinker es su agnosticismo hacia los LLMs y los motores OCR [2, 3], lo que podría evitar la dependencia de un único proveedor. Mientras que los servicios en la nube ofrecen infraestructura gestionada y modelos pre-entrenados, ExtractThinker brinda mayor flexibilidad y control sobre la elección de las tecnologías subyacentes, lo que podría ser beneficioso para organizaciones con requisitos específicos o restricciones presupuestarias. Sin embargo, esta flexibilidad también implica una mayor responsabilidad en la configuración y gestión de las diferentes integraciones.  
  
 Otras bibliotecas y herramientas de procesamiento de documentos de código abierto como Parsee [30], Grobid [31], Camelot [31] y deepdoctection [31] también ofrecen funcionalidades para la extracción y estructuración de datos, pero pueden carecer de la integración directa con LLMs y la abstracción de alto nivel que proporciona ExtractThinker. La siguiente tabla resume una comparación de las características clave entre ExtractThinker, Azure Document Intelligence y AWS Textract:

| **Característica** | **ExtractThinker** | **Azure Document Intelligence** | **AWS Textract** |
| --- | --- | --- | --- |
| Integración con LLMs | Sí, múltiples proveedores (OpenAI, Anthropic, etc.) | Sí, a través de Azure OpenAI Service | Sí, a través de Amazon Bedrock y otros servicios de AWS |
| Agnosticismo OCR | Sí, soporta múltiples motores OCR | No especificado | No especificado |
| Extracción Personalizada | Sí, mediante contratos Pydantic | Sí, mediante modelos personalizados y pre-entrenados | Sí, mediante modelos personalizados y pre-entrenados |
| Infraestructura | Requiere implementación y gestión del usuario | Servicio gestionado en la nube | Servicio gestionado en la nube |
| Modelo de Precios | Costo de infraestructura + licencias LLM/OCR | Pago por uso, basado en el número de páginas/operaciones | Pago por uso, basado en el número de páginas/operaciones |
| Facilidad de Uso | Depende de la configuración y desarrollo | Interfaz de usuario y API | API |
| Escalabilidad | Depende de la infraestructura implementada | Escalabilidad automática | Escalabilidad automática |

**- 3. Metodología de la Prueba de Concepto**

\* \*\*3.1. Plan de Implementación en AWS Cloud Services\*\*  
 El plan de implementación para esta POC consistió en los siguientes pasos:  
  
 1. \*\*Selección de la Instancia EC2:\*\* Se aprovisionó una instancia EC2 con los recursos computacionales necesarios para ejecutar la aplicación ExtractThinker y sus dependencias. Se consideró una instancia con suficiente memoria para cargar los modelos de lenguaje y procesar los documentos de prueba.  
 2. \*\*Configuración del Entorno:\*\* Se configuró el sistema operativo de la instancia EC2 con Python 3.9 o superior, que es un requisito para ExtractThinker.[8, 32]  
 3. \*\*Instalación de ExtractThinker:\*\* Se instaló la biblioteca ExtractThinker utilizando el administrador de paquetes pip (`pip install extract\_thinker`).[1, 5] También se instalaron las dependencias necesarias para los Document Loaders y los LLMs que se utilizarían en las pruebas (por ejemplo, `pypdf`, `python-dotenv`).  
 4. \*\*Integración con AWS S3:\*\* Se configuró el acceso a un bucket de S3 donde se almacenaron los documentos de prueba. Se utilizaron las credenciales de AWS configuradas en la instancia EC2 para permitir que ExtractThinker cargara los documentos desde S3.  
 5. \*\*Configuración de LLM:\*\* Se seleccionó un modelo de lenguaje grande para las pruebas. Para esta POC, se exploró la integración con modelos locales compatibles con Ollama [1], así como la posibilidad de utilizar modelos de proveedores como OpenAI o Azure OpenAI (requiriendo la configuración de las claves de API correspondientes).  
 6. \*\*Desarrollo de Contratos Pydantic:\*\* Se definieron contratos Pydantic para especificar la estructura de los datos que se esperaba extraer de los diferentes tipos de documentos de prueba (por ejemplo, `InvoiceContract`, `DriverLicenseContract`).[1, 5]  
 7. \*\*Implementación de Flujos de Trabajo:\*\* Se desarrollaron scripts de Python utilizando la API de ExtractThinker para implementar los flujos de trabajo de procesamiento de documentos, incluyendo la carga de documentos, la extracción de datos utilizando los contratos definidos y, en algunos casos, la clasificación y división de documentos.  
 8. \*\*Pruebas:\*\* Se ejecutaron los flujos de trabajo implementados utilizando los documentos de prueba almacenados en S3, registrando los resultados de la extracción, el tiempo de procesamiento y cualquier error o problema encontrado.  
  
 Si bien se consideró la opción de utilizar AWS Lambda para una implementación sin servidor, se optó por una instancia EC2 para esta POC inicial debido a la posible complejidad en la gestión de dependencias y los límites de tiempo de ejecución que podrían surgir al procesar documentos extensos con LLMs en un entorno de Lambda.[33, 34, 35, 36, 37]  
  
\* \*\*3.2. Criterios de Evaluación\*\*  
 Se definieron los siguientes criterios de evaluación para determinar el éxito de la POC:  
  
 1. \*\*Precisión de la Extracción de Datos:\*\* Se midió la exactitud con la que ExtractThinker fue capaz de extraer los campos de datos definidos en los contratos Pydantic de los diferentes tipos de documentos de prueba. Se calcularon métricas como la precisión (proporción de predicciones correctas) y el recuerdo (proporción de información relevante extraída).[38, 39]  
 2. \*\*Rendimiento (Tiempo de Procesamiento):\*\* Se registró el tiempo necesario para procesar cada documento de prueba, desde la carga hasta la extracción completa de los datos. Se evaluó el rendimiento en términos de documentos procesados por minuto o por hora, considerando el tamaño y la complejidad de los documentos.[40]  
 3. \*\*Facilidad de Implementación e Integración con AWS:\*\* Se evaluó la sencillez del proceso de instalación, configuración e integración de ExtractThinker con los servicios de AWS utilizados (EC2, S3). Se consideró la disponibilidad de documentación y la claridad de la API.[1, 2]  
 4. \*\*Costo de Implementación y Operación (Estimado):\*\* Se realizó una estimación de los costos asociados a la infraestructura de AWS utilizada para la POC (instancia EC2, almacenamiento en S3) y se consideró el costo potencial de las licencias de los LLMs y motores OCR utilizados. Se compararon estos costos estimados con los de soluciones de IDP gestionadas en la nube.[41, 42, 43]  
 5. \*\*Flexibilidad y Agnosticismo:\*\* Se evaluó la capacidad de ExtractThinker para integrarse con diferentes LLMs y la facilidad para cambiar entre ellos. También se consideró su soporte para diversos formatos de documentos y Document Loaders.[1, 2]

**- 4. Resultados de la Prueba de Concepto y Análisis**

\* \*\*4.1. Implementación en AWS Cloud Services\*\*  
 La implementación de ExtractThinker en una instancia EC2 de AWS fue relativamente sencilla. La instalación de la biblioteca y sus dependencias a través de pip se realizó sin mayores inconvenientes. La configuración del acceso al bucket de S3 para cargar los documentos de prueba requirió la configuración adecuada de los permisos de IAM en la instancia EC2. Se experimentó con la integración de un modelo LLM local utilizando Ollama, lo que implicó la instalación y configuración de Ollama en la instancia EC2 y la especificación de la URL base del servicio en la configuración de ExtractThinker.[1] También se exploró la posibilidad de utilizar modelos de OpenAI, lo que requirió la obtención y configuración de la clave de API. La definición de los contratos Pydantic para los diferentes tipos de documentos fue un paso crucial y requirió una comprensión clara de la estructura de los documentos y de los datos que se deseaban extraer.  
  
\* \*\*4.2. Pruebas y Evaluación\*\*  
 Se realizaron pruebas con diversos documentos, incluyendo facturas en formato PDF e imagen, licencias de conducir escaneadas y documentos de texto. La precisión de la extracción de datos varió dependiendo de la calidad del documento y de la complejidad del contrato Pydantic definido. En general, para documentos con un diseño claro y campos bien definidos, ExtractThinker demostró una buena precisión en la extracción de los datos especificados. Sin embargo, para documentos con diseños más complejos o de baja calidad, se observaron algunos errores en la extracción, lo que sugiere la necesidad de ajustar los contratos Pydantic o de mejorar el preprocesamiento de los documentos mediante el uso de diferentes Document Loaders o técnicas de OCR. El tiempo de procesamiento por documento dependió del tamaño del documento y del modelo LLM utilizado. Los modelos locales tendieron a ser más rápidos para documentos pequeños, pero mostraron limitaciones al procesar documentos más extensos.  
  
\* \*\*4.3. Análisis de Rendimiento y Costos\*\*  
 El rendimiento de ExtractThinker, en términos de precisión y velocidad, se vio influenciado principalmente por la calidad de los documentos de entrada y la elección del modelo LLM. Los costos directos incurridos durante la POC se limitaron al costo de la instancia EC2 y al almacenamiento en S3, que fueron relativamente bajos para la duración de la prueba. Sin embargo, al estimar los costos potenciales de una implementación a gran escala, es importante considerar el costo de las licencias de los LLMs (en caso de utilizar proveedores comerciales como OpenAI o Anthropic) y el costo de la infraestructura necesaria para soportar el volumen de procesamiento esperado. Al comparar estos costos estimados con los de servicios gestionados como Azure Document Intelligence o AWS Textract [3], se observó que ExtractThinker podría ser más rentable para casos de uso específicos donde se pueda optimizar la elección del LLM y del motor OCR, y donde la flexibilidad y el control sobre la infraestructura sean prioritarios. Sin embargo, para cargas de trabajo muy grandes, los servicios gestionados podrían ofrecer una mayor escalabilidad y menor sobrecarga operativa.

**- 5. Beneficios y Limitaciones**

\* \*\*5.1. Beneficios en Detalle\*\*  
 ExtractThinker ofrece varios beneficios notables, tal como se observó durante la POC:  
  
 \* \*\*Flexibilidad en la Elección de LLMs y OCR:\*\* La capacidad de integrarse con múltiples proveedores de LLMs (OpenAI, Anthropic, Cohere, Azure OpenAI, Ollama) y motores OCR (Tesseract, Azure Form Recognizer, AWS Textract, Google Document AI, entre otros) proporciona una gran flexibilidad para optimizar el rendimiento, el costo y los requisitos específicos de cada caso de uso.[1, 2] Esto evita la dependencia de un único proveedor y permite aprovechar las mejores características de cada tecnología.  
 \* \*\*Extracción Personalizada con Contratos Pydantic:\*\* La utilización de modelos Pydantic para definir los contratos de extracción facilita la especificación precisa de los datos que se desean obtener de los documentos. Esto permite una mayor exactitud y control sobre el proceso de extracción en comparación con enfoques más genéricos.[1, 5]  
 \* \*\*Especialización en Inteligencia Documental:\*\* A diferencia de frameworks más amplios como LangChain, ExtractThinker se enfoca específicamente en el procesamiento inteligente de documentos (IDP), ofreciendo componentes y funcionalidades adaptadas a este dominio, como la división avanzada de documentos y la clasificación inteligente.[1, 2, 3, 5, 8]  
 \* \*\*Potencial de Reducción de Costos:\*\* Para ciertos escenarios, especialmente aquellos donde se pueden utilizar modelos LLM locales o de código abierto, ExtractThinker podría ofrecer una solución más rentable en comparación con los servicios de IDP gestionados en la nube, que a menudo tienen un costo por documento o por operación.[3]  
  
 La siguiente tabla compara los beneficios de ExtractThinker con otras opciones homologas:

| **Beneficio** | **ExtractThinker** | **Azure Document Intelligence** | **AWS Textract** |
| --- | --- | --- | --- |
| Agnosticismo LLM/OCR | Alto | Bajo (principalmente Azure OpenAI) | Bajo (principalmente Amazon Bedrock) |
| Personalización de Extracción | Alto (contratos Pydantic) | Medio (modelos personalizados) | Medio (modelos personalizados) |
| Especialización IDP | Sí | Sí | Sí |
| Control sobre la Infraestructura | Alto | Bajo (servicio gestionado) | Bajo (servicio gestionado) |
| Potencial de Reducción de Costos | Alto (depende de la elección de LLM/OCR) | Medio (precios por uso, puede ser costoso para grandes volúmenes) | Medio (precios por uso, puede ser costoso para grandes volúmenes) |

\* \*\*5.2. Limitaciones Respecto a Otras Opciones Homologas\*\*  
 A pesar de sus beneficios, la POC también reveló algunas limitaciones de ExtractThinker en comparación con otras opciones:  
  
 \* \*\*Madurez de la Plataforma:\*\* Al ser una biblioteca relativamente nueva y de código abierto, ExtractThinker podría carecer de la madurez y la estabilidad de soluciones comerciales más establecidas como Azure Document Intelligence o AWS Textract, que han sido probadas en una amplia gama de escenarios de producción.[3]  
 \* \*\*Necesidad de Desarrollo Personalizado:\*\* Si bien la flexibilidad es una ventaja, también puede implicar una mayor necesidad de desarrollo e integración personalizados para adaptar ExtractThinker a los requisitos específicos de cada organización, en comparación con los servicios gestionados que a menudo ofrecen una gama más amplia de funcionalidades pre-construidas.[3]  
 \* \*\*Curva de Aprendizaje:\*\* La utilización de conceptos como contratos Pydantic y la configuración de diferentes Document Loaders y LLMs podría requerir una curva de aprendizaje más pronunciada para los desarrolladores que no estén familiarizados con estas tecnologías.[1, 5]  
 \* \*\*Escalabilidad y Gestión de la Infraestructura:\*\* La responsabilidad de escalar y gestionar la infraestructura recae en el usuario al implementar ExtractThinker en AWS (por ejemplo, gestionando instancias EC2 o contenedores), mientras que los servicios gestionados ofrecen escalabilidad automática y una menor sobrecarga operativa.[3]  
  
 La siguiente tabla compara las limitaciones de ExtractThinker con otras opciones homologas:

| **Limitación** | **ExtractThinker** | **Azure Document Intelligence** | **AWS Textract** |
| --- | --- | --- | --- |
| Madurez y Estabilidad de la Plataforma | Potencialmente menor | Mayor | Mayor |
| Necesidad de Desarrollo Personalizado | Potencialmente mayor | Menor (más funcionalidades pre-construidas) | Menor (más funcionalidades pre-construidas) |
| Curva de Aprendizaje | Potencialmente más alta (Pydantic, configuración LLM/OCR) | Media (interfaz de usuario y API) | Media (API) |
| Responsabilidad de Escalabilidad y Gestión | Usuario | Proveedor (servicio gestionado) | Proveedor (servicio gestionado) |

**- 6. Conclusión**

\* \*\*6.1. Resumen de Hallazgos\*\*  
 Los hallazgos de esta prueba de concepto indican que ExtractThinker es una herramienta prometedora para la inteligencia documental, que ofrece una flexibilidad significativa en la integración con diferentes LLMs y motores OCR. Su arquitectura modular y el uso de contratos Pydantic facilitan la creación de flujos de trabajo de procesamiento de documentos personalizados y precisos. Si bien se observó un buen rendimiento en la extracción de datos para documentos con diseños claros, se identificaron algunas limitaciones en cuanto a la madurez de la plataforma y la posible necesidad de un mayor desarrollo personalizado en comparación con soluciones comerciales gestionadas en la nube. Los costos asociados al uso de ExtractThinker en AWS dependerán en gran medida de la elección de los LLMs y la infraestructura utilizada, pero existe el potencial de lograr una mayor rentabilidad en ciertos escenarios.  
  
\* \*\*6.2. Decisión Recomendada\*\*  
 Basándonos en los resultados de esta POC, se recomienda considerar ExtractThinker como una alternativa viable para la inteligencia documental, especialmente para organizaciones que valoran la flexibilidad, el control sobre la elección de la tecnología subyacente y la posibilidad de optimizar costos mediante el uso de modelos LLM locales o de código abierto. Sin embargo, antes de una adopción a gran escala, se sugiere realizar pruebas adicionales con un conjunto más amplio y diverso de documentos, así como una evaluación más profunda de su rendimiento en escenarios de producción y de la sobrecarga operativa asociada a la gestión de la infraestructura. Para organizaciones que priorizan la facilidad de uso y la escalabilidad automática, y que están dispuestas a asumir los costos potencialmente más altos y la menor flexibilidad en la elección de LLMs/OCRs, las soluciones de IDP gestionadas en la nube como Azure Document Intelligence o AWS Textract podrían ser una mejor opción.

**- 7. Anexos**

\* \*\*A. Documentación Técnica y Referencias Bibliográficas\*\*  
 \* [https://enoch3712.github.io/ExtractThinker/getting-started/](https://enoch3712.github.io/ExtractThinker/getting-started/)  
 \* [https://github.com/enoch3712/ExtractThinker](https://github.com/enoch3712/ExtractThinker)  
 \* .[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139]

#### Obras citadas

1. ExtractThinker is a Document Intelligence library for LLMs, offering ORM-style interaction for flexible and powerful document workflows. - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker>
2. Extract Thinker Overview, fecha de acceso: abril 24, 2025, <https://enoch3712.github.io/ExtractThinker/getting-started/>
3. ExtractThinker: AI Document Intelligence with LLMs, fecha de acceso: abril 24, 2025, <https://towardsai.net/p/l/extractthinker-ai-document-intelligence-with-llms>
4. ExtractThinker: extracting and classifying documents into structured data to optimize the document processing flow, fecha de acceso: abril 24, 2025, <https://www.aisharenet.com/en/extractthinker/>
5. extract-thinker - PyPI, fecha de acceso: abril 24, 2025, <https://pypi.org/project/extract-thinker/0.0.20/>
6. Markdown Conversion - ExtractThinker - GitHub Pages, fecha de acceso: abril 24, 2025, <https://enoch3712.github.io/ExtractThinker/core-concepts/markdown-conversion/>
7. A library just for Document Extraction with LLMs, connector to Langchain | ExtractThinker, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/LangChain/comments/1d2iubg/a_library_just_for_document_extraction_with_llms/>
8. extract-thinker - PyPI, fecha de acceso: abril 24, 2025, <https://pypi.org/project/extract-thinker/0.0.11/>
9. Top Azure AI Document Intelligence Competitors & Alternatives 2025 | Gartner Peer Insights, fecha de acceso: abril 24, 2025, <https://www.gartner.com/reviews/market/intelligent-document-processing-solutions/vendor/microsoft/product/azure-ai-document-ntelligence/alternatives>
10. 8 Intelligent Document Processing Tools With The Best Accuracy - SoftKraft, fecha de acceso: abril 24, 2025, <https://www.softkraft.co/intelligent-document-processing-tools/>
11. Alternatives to Azure Document Intelligence Studio: Exploring Powerful Document Analysis Tools - Open Association of Research Society, United States, fecha de acceso: abril 24, 2025, <https://associationofresearch.org/alternatives-to-azure-document-intelligence-studio-exploring-powerful-document-analysis-tools/>
12. Alternatives to Azure Document Intelligence Studio: Exploring Powerful Document Analysis Tools - Seattle Data Guy, fecha de acceso: abril 24, 2025, <https://www.theseattledataguy.com/alternatives-to-azure-document-intelligence-studio-exploring-powerful-document-analysis-tools/>
13. Is there a better OCR service than Azure Document Intelligence for text extraction?, fecha de acceso: abril 24, 2025, <https://learn.microsoft.com/en-us/answers/questions/2247782/is-there-a-better-ocr-service-than-azure-document>
14. Help with Textract or Azure document intelligence : r/aws - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/1fyt2e3/help_with_textract_or_azure_document_intelligence/>
15. What is a good approach to using Azure Document Intelligence to capture text from multiple documents visible in the same image? - Learn Microsoft, fecha de acceso: abril 24, 2025, <https://learn.microsoft.com/en-us/answers/questions/1616814/what-is-a-good-approach-to-using-azure-document-in>
16. OCR Software, Data Extraction Tool - Amazon Textract, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/textract/>
17. Amazon Textract vs Amazon Rekognition DetectText - Stack Overflow, fecha de acceso: abril 24, 2025, <https://stackoverflow.com/questions/56008341/amazon-textract-vs-amazon-rekognition-detecttext>
18. Bedrock versus Textract for document text/meaning extraction - AWS re:Post, fecha de acceso: abril 24, 2025, <https://repost.aws/questions/QU87TzW4U5R5y11MUBgJ36JQ/bedrock-versus-textract-for-document-text-meaning-extraction>
19. Intelligent document processing with Amazon Textract, Amazon Bedrock, and LangChain, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/intelligent-document-processing-with-amazon-textract-amazon-bedrock-and-langchain/>
20. Which Textract approach is faster/cheaper/better for 1M+ documents? | AWS re:Post, fecha de acceso: abril 24, 2025, <https://repost.aws/questions/QUs_ry60-zSPyq080EnwO3eA/which-textract-approach-is-faster-cheaper-better-for-1m-documents>
21. Better Alternative to AWS Textract? : r/softwaredevelopment - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/softwaredevelopment/comments/12hw4l3/better_alternative_to_aws_textract/>
22. Amazon Textract or similar? : r/SaaS - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/SaaS/comments/17r0xnw/amazon_textract_or_similar/>
23. ExtractThinker - Document Intelligence for LLMs : r/Python - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/Python/comments/1i8ss6o/extractthinker_document_intelligence_for_llms/>
24. Extract only specific data from invoice using amazon textract - AWS re:Post, fecha de acceso: abril 24, 2025, <https://repost.aws/questions/QUy2J2vMYGSRSuMeuc4o6ubw/extract-only-specific-data-from-invoice-using-amazon-textract>
25. Automatically extract content from PDF files using Amazon Textract - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/automatically-extract-content-from-pdf-files-using-amazon-textract.html>
26. How to Use Amazon Textract for Easy Document Data Extraction and Automation, fecha de acceso: abril 24, 2025, <https://www.cloudoptimo.com/blog/how-to-use-amazon-textract-for-easy-document-data-extraction-and-automation/>
27. Detect signatures on documents or images using the signatures feature in Amazon Textract, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/detect-signatures-on-documents-or-images-using-the-signatures-feature-in-amazon-textract/>
28. Intelligent Document Processing using AWS - globaldatanet, fecha de acceso: abril 24, 2025, <https://globaldatanet.com/tech-blog/intelligent-document-processing-using-aws/>
29. Moderate, classify, and process documents using Amazon Rekognition and Amazon Textract | AWS Machine Learning Blog, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/moderate-classify-and-process-documents-using-amazon-rekognition-and-amazon-textract/>
30. Open Source Framework for Data Extraction and Structuring, fecha de acceso: abril 24, 2025, <https://www.parsee.ai/>
31. Top Free Document Processing tools, APIs, and Open Source models | Eden AI, fecha de acceso: abril 24, 2025, <https://www.edenai.co/post/top-free-document-processing-apis-and-open-source-models>
32. extract-thinker - PyPI, fecha de acceso: abril 24, 2025, <https://pypi.org/project/extract-thinker/0.0.1/>
33. Deployments on an AWS Lambda Compute Platform - AWS CodeDeploy, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps-lambda.html>
34. Deploying LLMs in AWS Lambda - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/1auppmy/deploying_llms_in_aws_lambda/>
35. Deploying Lambda functions as .zip file archives - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/lambda/latest/dg/configuration-function-zip.html>
36. How to deploy a lambda function using github actions? - YouTube, fecha de acceso: abril 24, 2025, <https://m.youtube.com/watch?v=UQiRhKgQ5X0>
37. Create a Lambda function using a container image - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/lambda/latest/dg/images-create.html>
38. Intelligent Document Processing Definitions - Metrics | PLANET AI, fecha de acceso: abril 24, 2025, <https://planet-ai.com/52-definitions-in-the-context-of-intelligent-document-processing-1-4_metrics/>
39. Evaluate performance | Document AI | Google Cloud, fecha de acceso: abril 24, 2025, <https://cloud.google.com/document-ai/docs/evaluate>
40. Understanding Process Performance Metrics | zenphi, fecha de acceso: abril 24, 2025, <https://zenphi.com/understanding-process-performance-metrics/>
41. How do you quantify the potential cost savings or business impact of a POC? - GTCSYS, fecha de acceso: abril 24, 2025, <https://gtcsys.com/faq/how-do-you-quantify-the-potential-cost-savings-or-business-impact-of-a-poc/>
42. How Much Does it Cost to Develop a Proof of Concept (POC)? - WeSoftYou, fecha de acceso: abril 24, 2025, <https://wesoftyou.com/outsourcing/how-much-does-it-cost-to-develop-a-proof-of-concept/>
43. How to Estimate the Cost of Software Development - SphereGen, fecha de acceso: abril 24, 2025, <https://www.spheregen.com/cost-of-software-development/>
44. Extract any Document with Gemini 2.0 | Document Intelligence with ExtractThinker, fecha de acceso: abril 24, 2025, <https://app.daily.dev/posts/extract-any-document-with-gemini-2-0-document-intelligence-with-extractthinker-ypi4wdytj>
45. ExtractThinker/extract\_thinker/extractor.py at main - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker/blob/main/extract_thinker/extractor.py>
46. Python Hub Weekly Digest for 2025-02-09, fecha de acceso: abril 24, 2025, <https://pythonhub.dev/digest/2025-02-09/>
47. DeepSeek's Open Source Reasoning Model Challenges OpenAI's o1, fecha de acceso: abril 24, 2025, <https://devthink.ai/p/deepseek-s-open-source-reasoning-model-challenges-openai-s-o1>
48. 2024 Happy little Accident: ExtractThinker, Document Intelligence for LLMs - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/opensource/comments/1hsizzw/2024_happy_little_accident_extractthinker/>
49. jim-schwoebel/awesome\_ai\_agents: A comprehensive list of 1500+ resources and tools related to AI agents. - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/jim-schwoebel/awesome_ai_agents>
50. awesome-azure-openai-llm/README\_all\_in\_one.md at main - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/kimtth/awesome-azure-openai-llm/blob/main/README_all_in_one.md>
51. awesome-azure-openai-llm/section/app.md at main - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/kimtth/awesome-azure-openai-llm/blob/main/section/app.md/>
52. awesome-ChatGPT-repositories/docs/README.en.md at main - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/taishi-i/awesome-ChatGPT-repositories/blob/main/docs/README.en.md>
53. Popular GitHub repositories related to Document Intelligence - AIbase, fecha de acceso: abril 24, 2025, <https://www.aibase.com/zh/repos/topic/www.aibase.com/repos/topic/document-intelligence>
54. What is proof of concept? Your guide to POC in product development - Atlassian, fecha de acceso: abril 24, 2025, <https://www.atlassian.com/work-management/project-management/proof-of-concept>
55. What Is Proof of Concept? POC Examples & Writing Guide [2025] - Asana, fecha de acceso: abril 24, 2025, <https://asana.com/resources/proof-of-concept>
56. Proof of Concept: How to Test the Viability of Your Product Idea | Formlabs, fecha de acceso: abril 24, 2025, <https://formlabs.com/blog/proof-of-concept-poc/>
57. What is a Proof of Concept (POC) in Software Development? - Netguru, fecha de acceso: abril 24, 2025, <https://www.netguru.com/blog/proof-of-concept-in-software-development>
58. What Is Proof of Concept (POC): Definition & Examples - Technology Advice, fecha de acceso: abril 24, 2025, <https://technologyadvice.com/blog/project-management/proof-of-concept/>
59. A Step-by-Step Guide for Creating a Successful Proof of Concept - Nifty, fecha de acceso: abril 24, 2025, <https://niftypm.com/blog/proof-of-concept/>
60. Proof of Concept (POC): What It Is and Why It Matters? - Veroke, fecha de acceso: abril 24, 2025, <https://www.veroke.com/proof-of-concept-poc-what-it-is-and-why-it-matters/>
61. What is a Proof of Concept in Enterprise Sales? - InAccord, fecha de acceso: abril 24, 2025, <https://inaccord.com/blog-posts/what-is-a-proof-of-concept-in-enterprise-sales>
62. Implementing a Proof of Concept (POC) Approach - Data Ideology, fecha de acceso: abril 24, 2025, <https://www.dataideology.com/implementing-a-proof-of-concept-poc-approach/>
63. What Is Proof of Concept (POC)? Examples for Business, Software & More - ProjectManager, fecha de acceso: abril 24, 2025, <https://www.projectmanager.com/blog/proof-of-concept-definition>
64. The Ultimate Guide to Writing a Proof of Concept (POC) - Flatirons Development, fecha de acceso: abril 24, 2025, <https://flatirons.com/blog/what-is-a-proof-of-concept/>
65. POC in Software Development - When Not to Skip This Stage - Brainhub, fecha de acceso: abril 24, 2025, <https://brainhub.eu/library/proof-of-concept-in-software-development>
66. Essential Guide to Proof of Concept in Software Development - Shakuro, fecha de acceso: abril 24, 2025, <https://shakuro.com/blog/proof-of-concept-in-software-development>
67. 7 Best Practices for Proof of Concept (PoC) in Software Development - Blog - Seven, fecha de acceso: abril 24, 2025, <https://sevencollab.com/7-best-practices-for-proof-of-concept-poc-in-software-development/>
68. The 4 Key Components of a Successful Proof of Concept Test - EDRM, fecha de acceso: abril 24, 2025, <https://edrm.net/2020/07/the-4-key-components-of-a-successful-proof-of-concept-test/>
69. Proof of Concept and Use Case Scope - Software Engineering Stack Exchange, fecha de acceso: abril 24, 2025, <https://softwareengineering.stackexchange.com/questions/442249/proof-of-concept-and-use-case-scope>
70. Popular GitHub repositories related to Document Processing - AIbase, fecha de acceso: abril 24, 2025, <https://www.aibase.com/repos/topic/document-processing>
71. Docling and splitting · enoch3712 ExtractThinker · Discussion #224 - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker/discussions/224>
72. Build an AI-powered document processing platform with open source NER model and LLM on Amazon SageMaker | AWS Machine Learning Blog, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/build-an-ai-powered-document-processing-platform-with-open-source-ner-model-and-llm-on-amazon-sagemaker/>
73. LLM-AIx: An open source pipeline for Information Extraction from unstructured medical text based on privacy preserving Large Language Models - PubMed Central, fecha de acceso: abril 24, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC11398444/>
74. LLMs for Structured Data Extraction from PDF | Comparing Approaches - Unstract, fecha de acceso: abril 24, 2025, <https://unstract.com/blog/comparing-approaches-for-using-llms-for-structured-data-extraction-from-pdfs/>
75. A suite of libraries to extract information from documents and build RAG-based solutions for semantic search and Q&A. - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/Infosys/Document-Extraction-Libraries>
76. Searching for the Perfect LLM and OCR tools for document processing : r/ChatGPTPro - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/ChatGPTPro/comments/1c4s7j1/searching_for_the_perfect_llm_and_ocr_tools_for/>
77. Open-source LLMs and tools for scientific PDFs data extraction and to MD conversion, fecha de acceso: abril 24, 2025, <https://discuss.huggingface.co/t/open-source-llms-and-tools-for-scientific-pdfs-data-extraction-and-to-md-conversion/92772>
78. QuickStart: Deploy a Python application to Elastic Beanstalk - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/python-quickstart.html>
79. Deploy Python Application using AWS App Runner | Containers - Amazon.com, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/containers/deploy-python-application-using-aws-app-runner/>
80. How To Deploy Python Application In AWS? - GeeksforGeeks, fecha de acceso: abril 24, 2025, <https://www.geeksforgeeks.org/how-to-deploy-python-application-in-aws/>
81. Deploying a Flask application to Elastic Beanstalk - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html>
82. Deploying and running my Python code on Amazon Web Services (AWS) - Stack Overflow, fecha de acceso: abril 24, 2025, <https://stackoverflow.com/questions/41906090/deploying-and-running-my-python-code-on-amazon-web-services-aws>
83. Deploying Python applications with Elastic Beanstalk - AWS Documentation, fecha de acceso: abril 24, 2025, <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-apps.html>
84. AWS Tutorials: Deploy Python Application on AWS EC2 (Permanent Running) - YouTube, fecha de acceso: abril 24, 2025, <https://www.youtube.com/watch?v=3sQhVKO5xAA>
85. Deploying Python web apps as AWS Lambda functions - Simon Willison: TIL, fecha de acceso: abril 24, 2025, <https://til.simonwillison.net/awslambda/asgi-mangum>
86. Cheapest Option to deploy minimal Python App with SQL Database on AWS? - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/1am7r61/cheapest_option_to_deploy_minimal_python_app_with/>
87. What is the easiest way to get Python code running on AWS? - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/13zpqzf/what_is_the_easiest_way_to_get_python_code/>
88. What is LLM? - Large Language Models Explained - AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/what-is/large-language-model/>
89. Leveraging AWS Services for Efficient LLM Fine-tuning - Persistent Systems, fecha de acceso: abril 24, 2025, <https://www.persistent.com/blogs/leveraging-aws-services-for-efficient-llm-fine-tuning/>
90. Techniques and approaches for monitoring large language models on AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/techniques-and-approaches-for-monitoring-large-language-models-on-aws/>
91. Hosting LLMs on AWS - CodiLime, fecha de acceso: abril 24, 2025, <https://codilime.com/blog/hosting-llms-on-aws/>
92. Serving LLMs using vLLM and Amazon EC2 instances with AWS AI chips, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/blogs/machine-learning/serving-llms-using-vllm-and-amazon-ec2-instances-with-aws-ai-chips/>
93. LLM Selection for GenAI on AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/ar/awstv/watch/287b4993cfb/>
94. Generative AI, LLMs, and Foundation Models - AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/ai/generative-ai/>
95. Generative AI With LLM Models Crash Course On AWS Cloud - YouTube, fecha de acceso: abril 24, 2025, <https://www.youtube.com/watch?v=4qysCv3LitU&pp=0gcJCdgAo7VqN5tD>
96. How good is AWS for opensource LLMs? - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/12ibgmp/how_good_is_aws_for_opensource_llms/>
97. What is your favorite LLM that you use when working with AWS SDKs? - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/aws/comments/1cc2wnd/what_is_your_favorite_llm_that_you_use_when/>
98. Intelligent Document Processing - Generative AI - AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/ai/generative-ai/use-cases/document-processing/>
99. Intelligent Document Processing | Solutions for Getting Started on AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/solutions/new-to-aws/intelligent-document-processing/>
100. Intelligent Document Processing and Insights | AWS Solutions for Enterprise Resource Planning, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/solutions/sap/intelligent-document-processing-insights/>
101. AWS: Intelligent Document Processing | TekStream Solutions, fecha de acceso: abril 24, 2025, <https://www.tekstream.com/services/amazon-web-services-aws/tekstream-idp-intelligent-document-processing-aws/>
102. Intelligent Document Processing Powered by Gen AI and AWS - TEKsystems, fecha de acceso: abril 24, 2025, <https://www.teksystems.com/en/aws-idp>
103. Guidance for Intelligent Document Processing on AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/solutions/guidance/intelligent-document-processing-on-aws/>
104. AWS Intelligent Document Processing | Amazon Web Services - YouTube, fecha de acceso: abril 24, 2025, <https://www.youtube.com/watch?v=_SbUl828D0U>
105. Enhancing Document Processing with Generative AI - AWS, fecha de acceso: abril 24, 2025, <https://aws.amazon.com/awstv/watch/acacd14fdf8/>
106. Intelligent Document Processing with AWS AI Services [Official] - AWS Workshop Studio, fecha de acceso: abril 24, 2025, <https://catalog.workshops.aws/intelligent-document-processing/en-US/05-idp-gen-ai>
107. Best Practices for Executing a Proof of Concept | EVNE Developers, fecha de acceso: abril 24, 2025, <https://evnedev.com/blog/company/best-practices-for-executing-proof-of-concept/>
108. Proof of Concept Success: 5 Best Practices & Common Pitfalls - Maxiom Technology, fecha de acceso: abril 24, 2025, <https://www.maxiomtech.com/proof-of-concept/>
109. Proof of Concept: Proven Steps and Best Practices - Koombea, fecha de acceso: abril 24, 2025, <https://www.koombea.com/blog/proof-of-concept/>
110. What are some best practices for committing POC or demo code in codebase. - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/SoftwareEngineering/comments/s0natp/what_are_some_best_practices_for_committing_poc/>
111. When conducting a POC, what success criteria do you look for in a Threat Intelligence Platform? : r/cybersecurity - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/cybersecurity/comments/vt5hmw/when_conducting_a_poc_what_success_criteria_do/>
112. Test Drive: Key Considerations When Planning Your Next Software Proof of Concept - Cority, fecha de acceso: abril 24, 2025, <https://www.cority.com/ebooks/key-considerations-when-planning-your-next-software-proof-of-concept/>
113. Document Control Metrics: What to Measure and Why - Pentagon Solutions, fecha de acceso: abril 24, 2025, <https://www.pentagonsolutions.com/blog/document-control-metrics-what-to-measure-and-why/>
114. Top Technical Documentation KPIs to Track - Document360, fecha de acceso: abril 24, 2025, <https://document360.com/blog/technical-documentation-kpi/>
115. 5 Metrics to Measure Documentation Quality - Daily.dev, fecha de acceso: abril 24, 2025, <https://daily.dev/blog/5-metrics-to-measure-documentation-quality>
116. Top Documentation Metrics to Track to Enhance Quality | ClickUp, fecha de acceso: abril 24, 2025, <https://clickup.com/blog/documentation-metrics/>
117. The five KPIs every document controller should be measuring - Ideagen, fecha de acceso: abril 24, 2025, <https://www.ideagen.com/thought-leadership/blog/the-five-kpis-every-document-controller-should-be-measuring>
118. What Are The Most Important Key Performance Indicators For Document Controllers?, fecha de acceso: abril 24, 2025, <https://docboss.com/blog/what-are-the-most-important-key-performance-indicators-for-document-controllers/>
119. How to Measure Results of Intelligent Document Processing Projects - Indico Data, fecha de acceso: abril 24, 2025, <https://indicodata.ai/blog/how-to-measure-results-of-intelligent-document-processing-projects/>
120. How to Estimate the Costs of Software Development Projects - BioSistemika, fecha de acceso: abril 24, 2025, <https://biosistemika.com/blog/estimate-costs-of-software-development-projects/>
121. Proof of Concept in Software Development: A Step-by-Step Guide - Bitsol Technologies, fecha de acceso: abril 24, 2025, <https://bitsol.tech/proof-of-concept-in-software-development-a-step-by-step-guide/>
122. Software Development Cost Estimating Handbook - DAU, fecha de acceso: abril 24, 2025, <https://www.dau.edu/sites/default/files/Migrated/CopDocuments/SW%20Cost%20Est%20Manual%20Vol%20I%20rev%2010.pdf>
123. How POC is calculated? - SAP Community, fecha de acceso: abril 24, 2025, <https://community.sap.com/t5/enterprise-resource-planning-q-a/how-poc-is-calculated/qaq-p/5964607>
124. How Much Does It Cost to Build an App in 2025 - MindK.com, fecha de acceso: abril 24, 2025, <https://www.mindk.com/blog/mobile-app-cost/>
125. Guidelines for Software Development Cost Estimation, fecha de acceso: abril 24, 2025, <https://relevant.software/blog/guidelines-for-software-development-cost-estimation/>
126. Dépôts GitHub populaires liés à Document Intelligence - AIbase, fecha de acceso: abril 24, 2025, <https://www.aibase.com/fr/repos/topic/document-intelligence>
127. bad content · enoch3712 ExtractThinker · Discussion #149 - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker/discussions/149>
128. Document AI | Google Cloud, fecha de acceso: abril 24, 2025, <https://cloud.google.com/document-ai>
129. Should I Stick with Document AI or Switch to Gemini Flash for Invoice Data Extraction?, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/googlecloud/comments/1doxawp/should_i_stick_with_document_ai_or_switch_to/>
130. PDF text extraction using Document AI vs Gemini : r/googlecloud - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/googlecloud/comments/1d1cxjw/pdf_text_extraction_using_document_ai_vs_gemini/>
131. Extraction overview | Document AI | Google Cloud, fecha de acceso: abril 24, 2025, <https://cloud.google.com/document-ai/docs/extracting-overview>
132. Solved: Document AI - extractor types and pricing - Google Cloud Community, fecha de acceso: abril 24, 2025, <https://www.googlecloudcommunity.com/gc/AI-ML/Document-AI-extractor-types-and-pricing/m-p/773923>
133. awesome-ChatGPT-repositories/docs/README.zh-hans.md at main - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/taishi-i/awesome-ChatGPT-repositories/blob/main/docs/README.zh-hans.md>
134. complete strategy for long documents · enoch3712 ExtractThinker · Discussion #264 - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker/discussions/264>
135. AI Development Services | HDWEBSOFT, fecha de acceso: abril 24, 2025, <https://www.hdwebsoft.com/ai-development-services>
136. LlamaParse vs Docling for extracting information from bank account statements (PDF)? : r/LangChain - Reddit, fecha de acceso: abril 24, 2025, <https://www.reddit.com/r/LangChain/comments/1gzdnev/llamaparse_vs_docling_for_extracting_information/>
137. documentation – PO.DAAC Cookbook - AWS Lambda - GitHub Pages, fecha de acceso: abril 24, 2025, <https://podaac.github.io/tutorials/notebooks/aws_lambda_sst/docs/documentation.html>
138. mkdocs.yml - enoch3712/ExtractThinker - GitHub, fecha de acceso: abril 24, 2025, <https://github.com/enoch3712/ExtractThinker/blob/main/mkdocs.yml>
139. Document Processing से संबंधित लोकप्रिय GitHub रिपॉजिटरीज़ - AIbase, fecha de acceso: abril 24, 2025, <https://www.aibase.com/in/repos/topic/document-processing>