StructiDoc AI - Document Intelligence

Unlocking the Power of Unstructured Enterprise Data into Machine-Interpretable Knowledge

May 12, 2025

- **The Challenge: Proliferation of Unstructured Data**
  - Most enterprise data (e.g., scanned PDFs, contracts, invoices, spreadsheets, handwritten notes) is unstructured or semi-structured.
  - Extracting structured, machine-readable data from unstructured or semi-structured data sources remains a major bottleneck for AI-powered automation, analytics, and decision-making.
- **The Role of Structured Data in Enhancing LLM Performance:**
  - LLMs achieve factual consistency, higher accuracy, and reliability when grounded in structured, high-quality data.
  - Production AI systems demand structured inputs—yet most enterprise data is unstructured. Bridging this gap requires automated pipelines to parse, validate, and transform raw documents at scale.
- **Growing Demand for AI-Ready Data**
  - AI-native enterprises increasingly require clean, structured, and explainable data:
    - Fine-tune small-scale language models(SLMs) on domain-specific corpora for task-specific customization
    - Build Retrieval-Augmented Generation (RAG) systems
    - Power document search, summarization, and reasoning agents
- **Our Proposed Solution:** A document ingestion platform designed to convert complex, unstructured documents into structured, machine-readable data optimized for LLMs and AI workflows.

- **Challenges with Complex Layouts in Unstructured Documents**:
  - Multi-column formats, nested tables, and embedded visuals(Figures, Images, etc) often lead to misinterpretation or data loss.
    Example:
    - A **research paper** with 2 columns is extracted as a jumbled text stream, mixing left and right column content.
    - A **nested table** in an invoice is read row-by-row, losing column associations (e.g., "Quantity" vs. "Unit Price").
- **Limited Contextual Understanding**:
  - Traditional OCR extracts text but cannot infer meaning or relationships.
    Example:
    - * Extracts "Apple" but cannot tell if it's **fruit** or **brand**.
    - Reads "Total: $100" and "Due: 30/05/2024" but fails to link them as part of the **same payment information**.

  AI Solution: LLMs classify "Apple" by context and group invoice fields logically.
- **Restricted Language and Font Support**:
  - Struggles with non-Latin scripts, handwritten text, or stylized fonts.
    Example:
    - * A **Japanese Kanji** receipt is misread as random symbols.
    - **Doctor's handwritten prescription** is rendered as gibberish.

  AI Solution: Multilingual transformers (e.g., VLMs such as GPT-4o) improve accuracy across scripts.

- **Dependence on Image Quality**:
  - Traditional OCR fails on poor scans or noisy images; requires ideal input.
    Example:
    - Blurry ID "DL 8HX" misread as "DL 8KX" by Tesseract.

    AI Solutions:
    - VLMs infer noisy text(blurred/unclear text) correctly using contextual understanding.
- **Security and Compliance Risks**:
  - Traditional cloud OCR exposes sensitive data to third-party services.
    Example:
    - Hospital records processed on external servers violate HIPAA(Health Insurance Portability and Accountability Act).

    LLM Solutions:
    - Apple's on-device AI combines OCR via the Vision framework and SLMs via Apple Intelligence to enable secure, on-device or on-premises processing (e.g., Apple's on-device AI on iPhones, iPads, and Macs).

Limitations of Traditional OCR/RPA Tools vs. LLM/VLM Solutions

- **Inflexibility**:
  - Robotic Process Automation(RPA) tools are software robots (or "bots") to automate repetitive, rule-based tasks that are typically performed by humans in digital systems.
    What RPA Tools Can Do with Documents:
    - Open a scanned PDF → run OCR → extract key fields → enter in form
    - Watch a folder → detect a new document → rename and route to system
    - Parse structured forms → validate values → trigger further workflows
  - RPA tools are effective only when document formats are rigid and predictable.
  - But RPA breaks on slight variations in structure or layout.
    Example:
    - A new invoice template requires re-training the entire RPA pipeline.

    VLM Solutions:
    - Claude 3 handles 100+ invoice formats out-of-the-box with layout-agnostic reasoning.
- **High Costs**:
  - Traditional systems require constant maintenance
    Example:
    - Full-Time Equivalents(FTEs/full-time staff members) needed to correct insurance claim OCR errors

    LLM Advantages:
    - Fine-tuned SLMs automate extraction, reducing the need for manual FTE intervention.

- **Vision-Centric Document Processing Engine:**
  - Uses layout-aware models to segment and classify document components—such as text blocks, tables, images, and figures—across both standard and non-standard layouts, including multi-column and nested structures
  - Applies specialized extraction pipelines for each content type (e.g., figure-caption linking), preserving semantic relationships and visual hierarchy
  - Reconstructs the document into a structured, LLM-ready format that retains its original meaning and context, enabling accurate downstream applications like RAG and semantic search
- **Advanced Document Parsing:**
  - Uses multi-pass Agentic OCR with VLMs to accurately extract structured data from complex documents. Generates machine-readable formats (`JSON`, `XML`, `HTML`) optimized for LLM pipelines and retrieval systems.
  - Supports custom schema definitions to fit domain-specific data extraction needs.
- **Deployment Flexibility:**
  - Supports both cloud-hosted SaaS and secure on-premises installations, ideal for regulated industries handling sensitive documents.
  - Provides REST APIs and Python SDKs for both synchronous and asynchronous ingestion workflows.
- **Security and Compliance:**
  - Enforces zero data retention—no documents are stored post-processing.
  - Offers air-gapped and on-premises deployment for maximum data privacy.
  - Compliant with HIPAA and SOC 2 Type 2, ensuring security and privacy controls.

- **1. RAG Accuracy (End-to-End QA / Semantic Fidelity)**
  - Measures how accurately the system extracts and interprets document content for downstream RAG workflows (e.g., search, summarization, question answering).

  Metrics:
  - **Exact Match (EM)** and **F1-score** on benchmark QA pairs.
  - **BERTScore**, **ROUGE**, and **BLEU** for evaluating summarization quality.
- **2. Processing Speed (Throughput and Latency)**
  - Measures how efficiently the system processes documents under various load conditions.

  Metrics:
  - **Latency per document** (seconds per document).
  - **Throughput** (documents per second or pages per minute).
  - **Time to JSON**: Time taken from PDF ingestion to generating structured, machine-readable output (e.g., JSON).
  - **Inference time** for layout models and information extraction modules.

Why Now? The Urgency to Act

- **Explosion of LLM and RAG Adoption**
  - Enterprises are racing to deploy LLMs, but struggle with poor-quality, unstructured data—stalling ROI on multi-million dollar AI investments.
- **Unaddressed Bottleneck: Document Ingestion at Scale**
  - Existing OCR and RPA solutions fail on real-world complexity—costing time, risking compliance, and limiting LLM effectiveness.
- **Emerging Standards: AI-Ready Data Pipelines**
  - Enterprises are shifting budgets from raw model training to data quality and retrieval augmentation—StructiDoc AI sits at this critical intersection.
- **Immediate Market Opportunity**
  - Growing demand in regulated industries (e.g., healthcare, finance, legal) for secure, explainable, and compliant document understanding solutions.
  - StructiDoc AI's on-premises, privacy-first, and LLM-aligned design positions it for early customer wins.
- **Call to Action:**
  - We are focused on accelerating go-to-market, scaling deployment, and meeting enterprise demand in this rapidly growing market.

- **End-to-End AI Workflow: From Data Ingestion to Model Serving**
  - **StructiDoc AI:** Converts unstructured enterprise documents (PDFs, images, scanned records) into structured, machine-readable data optimized for RAG pipelines.
  - **OptiInfer AI:** Delivers cost-efficient, high-speed, and scalable inference on these structured inputs using advanced system-level and reasoning-level optimizations without requiring model fine-tuning for accurate response generation.
- **Unlocking Value Across the Full Stack**
  - **Data Layer:** Automate extraction, structuring, and validation of enterprise data.
  - **Model Layer:** Optimize runtime performance and reliability of language models at scale.
- **Why This Integration Matters:**
  - Maximizes ROI by addressing both the **data quality bottleneck** and the **inference cost-performance trade-off**.
  - Enables enterprises to deploy production-grade AI solutions that are both **accurate** and **cost-effective**.
  - Provides a unified platform spanning **data ingestion**, **retrieval-augmented reasoning**, and **high-throughput model serving**.

OptiInfer AI – Test-Time Inference Optimization as a Service

Optimizing Language Model Serving for Speed, Efficiency, and Scalability

May 12, 2025

- **1. Inefficiency of Current Retrieval-Augmented Generation (RAG) Systems**
  - Most real-world LLM applications (e.g., search, chatbots, copilots) rely on RAG to reduce hallucinations and stay up-to-date. However, static RAG systems suffer from:
  - Redundant or irrelevant retrievals, increasing latency (↑ ms or s per query).
  - Unnecessary computational load, reducing inference throughput (↓ tokens/s).
  - Memory bottlenecks in Key-Value (KV) Caching, limiting the maximum context window length due to increased GPU VRAM consumption (↑ GB).
  - These limitations make RAG systems inefficient, costly, and difficult to scale for production workloads.

- **2. High Cost of Real-Time LLM Inference**
  - Serving LLMs in real-time incurs high costs due to:
    - High GPU memory demands (↑ GB), driven by KV cache growth in GPU VRAM. Increased memory usage reduces batch sizes, lowering throughput and raising costs.
    - Latency penalties from excessive retrieval or deep autoregressive decoding, where each additional token generation step compounds computation time (↑ ms or s per token/query).
    - Inefficient compute utilization for simple/short queries, further degrading throughput (↓ tokens/s).
    - Consequently, organizations incur steep cloud compute expenses, hindering cost-effective LLM deployment at scale.

Why Policy-Optimized Retrieval-Augmented Generation (PORAG) Is Relevant

- **Problem:** Existing RAG systems struggle with effective utilization of retrieved context
- Group Relative Policy Optimization (GRPO) is a reinforcement learning-based fine-tuning algorithm to enhance the reasoning capabilities of LLMs.
- **Our Approach:** Fine-tune SLMs through policy optimization over retrieved contexts
  - Integrates retrieval directly into the instruction tuning process
  - The "policy" refers to the SLM's parameters that govern text generation
  - Uses Group Relative Policy Optimization (GRPO) to update these parameters
  - Keeps retrieval mechanism fixed (computational efficiency)
- **Group Relative Policy Optimization (GRPO):**
  - Evaluates groups of generated responses relative to each other
  - **Process:** (1) Generate multiple responses per prompt, (2) Calculate rewards, (3) Normalize scores within groups, (4) Update policy parameters
  - Minimizes the **clipped advantage-weighted policy loss** with KL divergence regularization to ensure stable and controlled updates
  - Token-Level Loss Computation: GRPO applies the same advantage to all tokens, enabling finer-grained loss computation across long or multi-step outputs.
  - Outcome and Process Supervision: GRPO supports rewards on final outputs (outcome) and intermediate reasoning steps (process), improving learning for long or multi-step chain-of-thought responses.

- **The GRPO Loss Function:**
  - The GRPO loss function is a clipped policy optimization objective with group-relative advantage and KL penalty, formulated as:
  - The GRPO loss is defined as:

$$\mathcal{L}_{\mathsf{GRPO}}(\theta) = \mathbb{E}_{o\in\mathcal{G}}\left[\min\left(r(o;\theta)\cdot A(o),\ \mathrm{clip}(r(o;\theta), 1-\epsilon, 1+\epsilon)\cdot A(o)\right) - \beta\cdot \mathrm{KL}(\pi_\theta(o)\,\|\,\pi_{\mathsf{ref}}(o))\right]$$

  - **Where:**
    - $o \in \mathcal{G}$ are sampled outputs in a group $\mathcal{G}$
    - $r(o;\theta) = \frac{\pi_\theta(o)}{\pi_{\mathsf{old}}(o)}$ is the probability ratio, $A(o) = \frac{r(o)-\mu}{\sigma}$ is the group-normalized advantage
    - ϵ is the clipping threshold, β is the KL penalty coefficient
    - $\mathrm{KL}(\pi_\theta(o)\,\|\,\pi_{\mathsf{ref}}(o))$ penalizes large deviations from the reference model
  - **Fine-tuning Efficiency:** Uses QLoRA to reduce memory and compute overhead during training
- **Composite Reward Function for SLM Policy Optimization:**
  - Applies only to generated responses (retrieval is fixed)
  - $R(y) = 0.3 \times \text{ROUGE-L F1} + 0.2 \times \text{Length Ratio Penalty} + 0.5 \times \text{LLM-as-Judge Score}$
  - Optimizes for semantic similarity, brevity, factual correctness, and relevance
- **Key Benefits:**
  - **Efficient Inference:** Single-shot decoding with standard sampling
  - **No Multi-Candidate Ranking:** Avoids expensive reward computation at inference
  - **Superior Performance:** Significantly outperforms vanilla RAG on factuality metrics
- **Scalability:** Suitable for deployment in memory-constrained environments

- **Limitations of Static RAG:**
  - **Unnecessary Retrievals:** Always retrieves without checking if the available context is already sufficient, resulting in unnecessary latency and higher retrieval costs.
  - **Imprecise Querying:** Builds a static query from the initial user input, missing opportunities to adapt queries based on evolving context or partial answers, leading to incomplete or inaccurate responses.
  - **Fixed Reasoning Depth:** Uses static generation lengths, risking over-generation on simple tasks or under-generation on complex tasks.
- **Our Inference-Time Optimization:**
  - It modifies the behavior at inference time by dynamically deciding when to retrieve and what to retrieve based on the evolving context during generation without altering the model weights.
  - **Dynamic Retrieval:** Retrieves only when the technique detects gaps in knowledge or high uncertainty.
  - **Context-Aware Querying:** Leverages attention over the entire context to build precise, context-aware queries targeting missing information to fill information gaps to generate accurate response.
  - **Adaptive Reasoning:** Varies generation depth based on task complexity and evolving context, balancing quality and efficiency.

.

- **Key Advantage:**
  - Improves retrieval precision, reduces latency, and enhances factuality.

- We focus on low-level system-level optimizations that improve hardware-level performance to maximize runtime performance of SLMs.
- Inference-time optimization focuses on improving runtime efficiency of language models without modifying their parameters, targeting key system-level metrics: latency, throughput, and memory usage.

Key Performance Metrics:

- **Latency:** Time taken to generate a complete response (lower is better)
- **Throughput:** Number of tokens generated per second (higher is better)
- **Memory Efficiency:** GPU memory (VRAM) consumption impacting batch size and scalability

Techniques:

- **FlashAttention:** Efficient attention computation reduces memory bandwidth bottlenecks, improving both **latency** and **throughput**.
- **PagedAttention with KV-Cache Quantization:** Organizes the KV-cache into non-contiguous memory blocks to avoid fragmentation, improving **memory efficiency** and supporting larger batch sizes.
- **Lookahead Decoding:** Speculatively generates and verifies tokens in parallel to reduce generation latency while maintaining **output quality**.

Characteristics:

- Require **no retraining or fine-tuning** of model weights. Do not require multiple decoding passes, focus on accelerating vanilla decoding while maintaining output quality. Purely engineering/system-level improvements.

At test time, algorithmic or reasoning-level optimizations can significantly improve the **factuality**, **reliability**, and **quality** of model outputs by modifying the generation strategy—**without requiring any fine-tuning or retraining of model weights**.

Key Focus Areas:

- **Multi-Path Reasoning:** Explore multiple reasoning trajectories and select the most consistent answer to improve robustness.
- **Expert-Like Reflection:** Simulate expert behaviors such as critique, reflection, and structured re-evaluation.

Core Characteristics:

- Works entirely at inference-time without modifying model weights.
- Focuses on improving **output quality** rather than computational speed.
- May increase computational cost by generating and evaluating multiple candidate responses.
- Relies on advanced decoding algorithms rather than parameter updates or retraining.

Benefits at a Glance:

- ✓ Improve response quality without model fine-tuning.
- ✓ Enhance factuality by verifying consistency across reasoning paths.
- ✓ Dynamically control computational effort based on task complexity.
- ✓ Simulate expert-like critique and structured reasoning to improve reliability.

- **Self-Consistency:** Selects the most consistent answer by clustering multiple independently generated reasoning paths.
- **Best-of-N Sampling:** Picks the best from N candidates by self-evaluating response quality.
- **Chain-of-Thought with Reflection:** Guides reasoning through structured thinking, reflection, and answering phases in a single pass.
- **Entropy-Guided Decoding:** Dynamically adjusts sampling parameters based on model uncertainty to balance exploration and precision.
- **Chain-of-Thought Decoding:** Explores multiple reasoning paths and selects the most reliable based on token-level scoring.
- **RE^2 (Re-Reading and Re-Analyzing):** Structures reasoning into reading, re-reading, and final answer phases for deeper analysis.
- **Mixture of Agents (MoA):** Combines diverse generation, critique, and synthesis to produce refined responses.
- **Reimplementation Then Optimize (RTO):** Refines solutions by re-implementing from extracted specs and optimizing the final output.
- **PlanSearch:** Decomposes complex queries into multi-step planning and transformation stages before answering.
- **Monte Carlo Tree Search (MCTS):** Searches through reasoning paths using simulation and backpropagation for optimal responses.
- **R* Algorithm:** Uses guided tree search with consistency checks to ensure reliable and structured reasoning.

- **Policy-Optimized RAG (PORAG):**
  - RL fine-tuning technique for domain customization of SLMs
  - Significantly improves factual accuracy in responses
  - Enhances context utilization without increasing inference costs
  - Optimizes for brevity and relevance in generated content
- **Adaptive Inference for RAG:**
  - Reduces unnecessary retrievals, lowering latency and cost
  - Improves retrieval precision by targeting information gaps
  - Balances generation quality and computational efficiency
  - Enhances factuality without model retraining
- **Inference-Time Optimizations: Modular, plug-and-play framework:**
  - Accelerates token generation and reduces memory usage
  - Improves output quality through multi-path verification
  - Enhances reliability through expert-like reasoning patterns
  - Achieves better results without the need for fine-tuning

Key Impact: Enables faster, more accurate, and cost-efficient RAG across diverse applications and environments.

Integrated Workflow

Unstructured Documents

PDFs, Images, Scans

↓ StructiDoc AI

Structured Data

JSON, HTML, Knowledge Graphs

↓ OptiInfer AI

Optimized AI Responses

Fast, Accurate, Cost-Effective

Questions & Discussion