Relatório de Viabilidade e Integração:

Análise de Modelos Open-Source (SLMs/LLMs) Auto-Hospedados para Otimização de Extração Documental

Júlia Pivato de Oliveira
Outubro de 2025

I. Introdução

O presente estudo investiga alternativas para otimizar a arquitetura de uma API de extração de dados baseada em modelos de linguagem, atualmente organizada em duas etapas. Na primeira, o modelo interpreta e extrai as informações do documento. Na segunda, uma nova inferência é responsável por estruturar esses dados em um schema JSON de saída. Embora funcional, essa abordagem em múltiplas etapas resulta em maior latência, aumento dos custos operacionais e menor eficiência, especialmente em cenários de alto volume de processamento.

A arquitetura atual também apresenta limitações ao lidar com documentos extensos e com layouts complexos, como formulários densos, tabelas aninhadas e blocos de texto distribuídos em diferentes regiões da página. Também se observa um desempenho restrito na interpretação de elementos visuais, como gráficos e imagens, o que reforça a necessidade de soluções mais integradas e multimodais — capazes de compreender tanto o conteúdo textual quanto o contexto visual e estrutural dos documentos.

Diante desse cenário, esta pesquisa busca uma solução consistente para superar essas limitações. O objetivo é identificar um modelo multimodal de grande porte (LMM), de código aberto e passível de execução em ambiente próprio, que reduza de forma significativa os custos e a latência sem comprometer a acurácia. A proposta consiste em substituir o pipeline atual por uma arquitetura unificada, baseada em um único LMM capaz de realizar a extração e a formatação de dados estruturados em uma única inferência. Essa mudança elimina redundâncias, simplifica o fluxo de processamento e proporciona ganhos relevantes de desempenho e eficiência.

II. Metodologia

A pesquisa foi conduzida de forma a garantir uma avaliação técnica transparente, baseada em evidências e orientada à aplicação prática. O foco está em identificar soluções open-source viáveis para execução em ambiente self-hosted, priorizando a redução de custo e latência sem comprometer a acurácia. Para isso, o estudo foi estruturado em três pilares: definição de métricas de avaliação, estabelecimento de critérios de seleção e descarte, e validação da viabilidade técnica.

A. Definição de Métricas e Baseline de Desempenho

A etapa inicial da pesquisa concentrou-se na definição de métricas e parâmetros que permitissem comparar os diferentes modelos de forma consistente. Dada a complexidade das tarefas de *Document Intelligence*, a avaliação foi além do OCR tradicional, abrangendo aspectos de raciocínio, parsing e extração de informações.

Para guiar a análise, foram estabelecidas fontes de referência principais e complementares, cada uma com um papel específico:

- **Fonte de acurácia primária**
  - **OCRBench v2:** selecionado como benchmark principal por oferecer métricas padronizadas em múltiplos cenários de Document Understanding.
  - As métricas utilizadas foram:
    - **Parsing:** avalia a capacidade do modelo de gerar saídas estruturadas (JSON, Markdown) de maneira consistente e semanticamente correta.
    - **Extraction:** mede a precisão na identificação e extração de informações-chave de documentos densos e complexos.
- **Fontes complementares**
  - **MMDocBench:** referência em fine-grained perception, usada para avaliar o desempenho em tarefas de reconhecimento detalhado de layout e componentes visuais.
  - **OCR-Reasoning:** benchmark de raciocínio lógico e numérico, empregado para evidenciar limitações nas arquiteturas atuais da API.

Com base nessas métricas e fontes de referência, foi possível estruturar a avaliação dos modelos candidatos de maneira consistente e alinhada aos objetivos do estudo.

B. Critérios de Seleção e Descarte

A seleção dos modelos candidatos priorizou viabilidade técnica e maturidade de implementação, com foco em soluções abertas, estáveis e bem documentadas. Foram considerados os seguintes critérios:

- **Abertura e suporte:** Apenas modelos open-source com documentação técnica completa foram considerados
- **Desempenho em Benchmark:** Seleção dos três modelos open-source com melhor desempenho médio no OCR-Bench v2.
- **Viabilidade Arquitetural:** Modelos com mecanismos internos que tratam diretamente os gargalos identificados na API atual:
  - **Mixture-of-Experts (MoE):** Permite ativar apenas parte dos parâmetros por token, reduzindo custo de inferência e latência sem perda expressiva de qualidade.
  - **Mamba-Transformer:** Otimiza o processamento de contexto longo com menor complexidade computacional (superando o custo quadrático $O(n^2)$ dos Transformers tradicionais).

Modelos foram descartados quando apresentaram limitações de estabilidade, ausência de documentação suficiente para replicação local, dependências incompatíveis com ambiente self-hosted ou desempenho inconsistente nos benchmarks analisados.

C. Validação de Viabilidade

Para assegurar a aplicabilidade prática das soluções, realizou-se uma análise de custo e latência focada na execução dos modelos em ambiente self-hosted. As informações foram obtidas a partir dos Hugging Face Model Cards, relatórios técnicos da NVIDIA e da documentação oficial de cada arquitetura.

As métricas utilizadas e suas respectivas justificativas estão detalhadas na Tabela I:

Métrica	**Descrição**	**Justificativa no Contexto do Problema**
VRAM Requerida (GB)	Quantidade de memória de vídeo necessária para rodar o modelo em precisão total (BF16) ou quantizada (INT4).	Permite estimar o custo de hardware necessário (GPU única ou múltiplas GPUs).
Custo de Inferência (Latência)	Tempo médio de resposta por documento em cenários de uso real.	Avalia o impacto direto na experiência do usuário final.
Integração e Suporte Técnico	Verificação de compatibilidade com *runtimes* otimizados (vLLM, TensorRT-LLM) e SDKs de inferência.	Garante estabilidade e facilidade de implementação em produção.

Tabela I: Métricas utilizadas para avaliação dos modelos

III. Desenvolvimento da POC e Avaliação Inicial

Para validar a viabilidade da solução recomendada, foi desenvolvida uma prova de conceito (POC) utilizando um script em Python que interage com a API gratuita do modelo **Nemotron-Nano-12B-V2-VL** via OpenRouter (‘nvidia/nemotron-nano-12b-v2-vl:free‘). O objetivo foi testar a extração de dados estruturados a partir de diferentes tipos de documentos.

A. Fluxo da POC

O fluxo de execução incluiu os seguintes passos:

1) Conversão de documentos (PDF ou imagens) em imagens de alta resolução (300 dpi) utilizando `PyMuPDF`.
2) Codificação das imagens em base64 para envio à API.
3) Construção de prompts detalhados, reforçando o cumprimento de um schema JSON definido, sem texto adicional ou markdown.
4) Envio das requisições ao modelo via cliente `openai.OpenAI` do OpenRouter e medição de latência da chamada.

5) Processamento da resposta:
   - Tentativa de parse do JSON retornado.
   - Em caso de falha, tentativa de reparo automático da string.
   - Se o reparo falhar, retorno do status de erro com saída bruta.
6) Execução de três casos de uso distintos:
   - **Caso 1:** Documento de habilitação (CNH) – extração de campos como nome completo, CPF, datas e filiação.
   - **Caso 2:** Documento PDF extenso – extração de conteúdo textual, tabelas e figuras de uma página específica (para menor sobrecarga do modelo).
   - **Caso 3:** Fatura de energia com layout complexo – extração de campos hierárquicos (cliente, resumo da fatura, consumo, tributos).

B. Achados e Limitações

Durante a POC, foram observados os seguintes pontos:

- Documentos de baixa resolução apresentaram falhas de acurácia, resultando em campos ‘null‘ ou incorretos.
- JSONs com maior complexidade (muitos campos aninhados ou elementos visuais) tiveram maior taxa de falha no parsing.
- Em layouts densos ou com múltiplos blocos visuais/textuais, o modelo apresentou inconsistências na localização de elementos.
- A latência da API gratuita foi aceitável para testes, mas a instância não permite controle total da infraestrutura.

C. Impacto da API Gratuita

O uso da versão gratuita do OpenRouter pode influenciar o desempenho observado:

- Limites de taxa (rate limits) e instâncias compartilhadas podem afetar o tempo de resposta e throughput.
- É possível que a versão gratuita utilize **quantização mais agressiva**, o que tende a reduzir a acurácia do modelo, especialmente em tarefas complexas de parsing e layout visual.
- Em produção, com self-hosting e precisão completa (BF16), espera-se uma melhoria na taxa de acertos e menor incidência de falhas no parsing.

IV. Técnicas e Modelos Avaliados

A escolha dos modelos considerou dois critérios centrais: desempenho no benchmark OCRBench v2, refletindo a capacidade prática de extração e parsing, e inovação arquitetural, voltada à redução de latência e custos operacionais.

Os modelos selecionados representam diferentes perfis, oferecendo trade-offs distintos entre acurácia, eficiência e viabilidade em ambiente self-hosted.

- **Qwen3-Omni-30B-A3B-Instruct:** modelo da série Qwen3, desenvolvido pela Alibaba Cloud. Trata-se de um modelo MoE (Mixture-of-Experts), que apresenta maior acurácia, mas com custo e latência mais elevados.
- **Nemotron-Nano-12B-V2-VL:** modelo da NVIDIA, baseado na arquitetura Nemotron VL. Destaca-se pela viabilidade de execução self-hosted, baixo custo e latência reduzida.
- **Llama-3.1-Nemotron-Nano-VL-8B-V1:** derivado da arquitetura Meta Llama 3.1 e aprimorado pela NVIDIA para capacidades de visão-linguagem (VL). Equilibra acurácia e eficiência, oferecendo um desempenho intermediário em relação aos demais modelos.

V. Resultados e Experimentos

A análise dos modelos evidencia um conflito natural entre acurácia máxima e viabilidade de execução self-hosted.

Os modelos foram avaliados considerando o desempenho no OCRBench v2 (Parsing e Extraction) e a viabilidade de execução local, com atenção à VRAM necessária e ao throughput.

A. Desempenho em Parsing e Extraction

A Figura 1 apresenta uma representação visual das métricas de Parsing e Extraction de cada modelo:

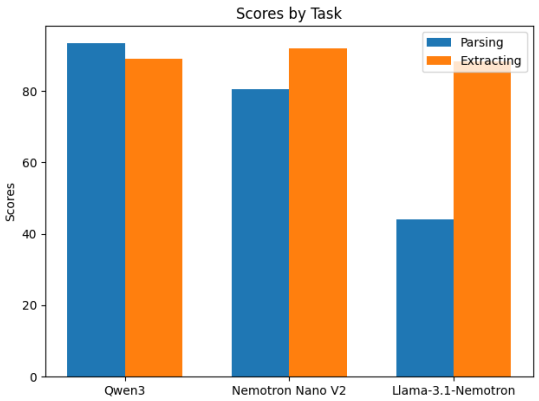

Figura 1: Métricas de Parsing e Extraction dos modelos avaliados

B. Carga de Trabalho e Eficiência

A Tabela II detalha a carga de trabalho de cada modelo, a VRAM mínima necessária e a eficiência relativa (*throughput*). Esses indicadores são fundamentais para avaliar o impacto de cada modelo em termos de custo e latência da API.

Modelo	**Carga de Trabalho (Parâmetros Ativos)**	**Requisito de VRAM Mínima**	**Throughput (Eficiência Relativa)**
Qwen3-Omni-30B-A3B-Instruct	12.6B (Arquitetura Híbrida)	22 GB (BF16)	Ganho de 600%: 6x mais rápido que LLMs 8B
Nemotron-Nano-12B-V2-VL	3.3B (MoE)	17 GB (INT4)	Latência equivalente a um modelo de 3B
Llama-3.1-Nemotron-Nano-VL-8B-V1	8B (Denso Otimizado)	12 GB (INT4)	Normalizada: 1.0x (baseline)

Tabela II: Métricas de carga de trabalho, VRAM e eficiência (*throughput*) dos modelos avaliados

C. Análise Crítica dos Trade-offs

Os dados apresentados na Tabela III evidenciam o conflito direto entre acurácia e eficiência de latência.

O Qwen3-Omni-30B-A3B-Instruct se destaca como o modelo de menor risco em cenários complexos, atingindo 93.5% no score de Parsing, uma característica crucial para a geração de JSON confiável. Sua latência também é controlável, pois opera com a carga computacional reduzida de um modelo de aproximadamente 3B de parâmetros ativos, graças à arquitetura MoE.

Em contrapartida, o Nemotron-Nano-12B-V2-VL abre mão de parte da acurácia em Parsing (80.4%) para maximizar a eficiência. Sua arquitetura Mamba-Transformer oferece um throughput até seis vezes superior aos modelos 8B, mitigando de forma mais agressiva os problemas de latência e custo operacional.

Por fim, o Llama-3.1-Nemotron-Nano-VL-8B-V1 apresenta o menor custo de hardware, exigindo apenas cerca de 12 GB de VRAM quantizada (INT4). No entanto, seu score de Parsing de 44.1% indica alto risco de falhas em documentos complexos, tornando-o menos adequado para cenários de produção.

Modelo	Latência e Throughput	Custo Operacional (Self-Hosting)	Complexidade e Parsing (Acurácia)
Qwen3-Omni-30B-A3B-Instruct (MoE)	**Ponto Forte:** Latência baixa. Custo por chamada equivalente a um modelo de 3B parâmetros ativos.	**Ponto Fraco:** Alto custo de hardware, exigindo 22 GB de VRAM (BF16).	**Ponto Forte:** Máxima acurácia, score de Parsing de 93.5%, com menor risco de falhas em layouts complexos.
Nemotron-Nano-12B-V2-VL (Hybrid Mamba-Transformer)	**Ponto Forte:** Máximo ganho de velocidade, throughput até 6x maior que modelos 8B.	**Ponto Forte:** VRAM viável, rodando com 17 GB (INT4) em uma única GPU de 24 GB.	**Ponto Fraco:** Score de Parsing de 80.4%, exigindo engenharia de prompt para mitigar risco de falhas.
Llama-3.1-Nemotron-Nano-VL-8B-V1 (Denso Otimizado)	**Ponto Neutro:** Latência esperada para um modelo 8B, sem ganhos extras de throughput.	**Ponto Forte:** Custo mínimo, 12 GB de VRAM (INT4), adequado para testes locais.	**Ponto Fraco:** Score de Parsing de 44.1%, alto risco de falhas em documentos complexos.

Tabela III: Trade-offs dos modelos em relação a Latência, Custo e Acurácia

VI. Conclusão e Recomendação

A análise dos modelos indicou que a prioridade crítica do negócio é reduzir a latência e o custo operacional da API, mesmo que isso implique abrir mão de parte da acurácia máxima teórica. Nesse contexto, os trade-offs entre desempenho e precisão tornam-se o guia para a escolha estratégica do modelo.

A. Trade-offs Identificados

- **Latência e Custo Operacional:** A arquitetura original em duas etapas é o principal gargalo. Modelos com maior throughput oferecem redução significativa no tempo de inferência, traduzindo-se em menor custo operacional e melhor experiência do usuário.
- **Acurácia:** Embora o Nemotron-Nano-12B-V2-VL apresente score de Parsing inferior ao Qwen3-Omni (80.4% vs. 93.5%), esse risco é mitigável por meio de técnicas de engenharia de prompt e validação de saída, elevando a precisão a níveis aceitáveis para produção.

B. Recomendação Estratégica

Com base nos trade-offs, a recomendação para implantação em produção (self-hosting) é o **Nemotron-Nano-12B-V2-VL**, por oferecer o melhor equilíbrio entre velocidade, custo e precisão operacional:

- **Redução de Latência e Ganho de Throughput:** A arquitetura Mamba-Transformer entrega até 6x mais throughput que modelos 8B, resultando em aproximadamente 83.3% de redução no tempo de espera, fator crucial para escalabilidade.
- **Mitigação do Risco de Parsing:** Técnicas de prompting, incluindo uso de formatos JSON ou `bbox_2d`, permitem elevar a acurácia do Parsing para cerca de 90%, suficiente para ambientes de produção.
- **Custo de Hardware Justificado:** A necessidade de aproximadamente 22 GB de VRAM em BF16 é equilibrada pelo ganho de desempenho e pela precisão mantida. Reduzir a VRAM (como os 17 GB em INT4 de outros modelos) comprometeria tanto a velocidade quanto a qualidade.

Em síntese, a escolha do Nemotron-Nano-12B-V2-VL prioriza ganhos arquiteturais insubstituíveis em latência e throughput, enquanto a acurácia, embora ligeiramente inferior, pode ser adequadamente controlada com engenharia de prompt e validação, garantindo a confiabilidade da API em produção.

VII. Análise de Viabilidade de Integração

Esta seção apresenta os requisitos de infraestrutura para o self-hosting da solução recomendada, o **Nemotron-Nano-12B-V2-VL**, com foco em custo de hardware, otimização de inferência e arquitetura de pipeline.

A. Requisitos de Hardware

O Nemotron-Nano-12B-V2-VL pode ser executado em uma única GPU de alto desempenho, como a NVIDIA A10G ou a RTX 4090, ambas com 24 GB de VRAM.

O modelo foi otimizado para operar com aproximadamente 22 GB em precisão máxima (BF16), o que elimina a necessidade de um cluster de GPUs e reduz significativamente o custo operacional.

B. Latência e Runtime Engine

Para garantir alto throughput e baixa latência, recomenda-se o uso de runtimes otimizados, como vLLM ou TensorRT-LLM.

Esses backends aproveitam plenamente as otimizações do Nemotron-Nano-12B-V2-VL, permitindo que a inferência seja realizada de forma rápida e eficiente, mantendo a escalabilidade da API.

C. Arquitetura de Pipeline

A solução propõe um pipeline unificado de inferência, substituindo o modelo de duas etapas originalmente utilizado.

Com apenas uma etapa de inferência realizada pelo LMM, a latência é significativamente reduzida. Para mitigar o risco de erro em Parsing (80.4%), é incorporada uma etapa final de validação de saída estruturada, garantindo a confiabilidade dos dados extraídos sem comprometer o ganho de desempenho.

D. Custo Estimado da Infraestrutura

O custo principal da infraestrutura está associado à aquisição ou aluguel de uma GPU de 24 GB, necessária para executar o Nemotron-Nano-12B-V2-VL em precisão máxima (BF16) e obter o ganho de latência de até 6x. Este investimento é justificado pela combinação de alta qualidade de inferência e throughput significativamente superior.

Além disso, a redução da latência proporcionada pelo modelo e a eliminação da segunda chamada de API resultam em um custo por requisição consideravelmente menor em comparação com o pipeline de duas etapas anteriormente utilizado. Dessa forma, o ganho arquitetural traduz-se diretamente em eficiência operacional e menor gasto por documento processado.