TECH4HUMANS

DESAFIO TÉCNICO: PESQUISADOR DE IA GENERATIVA (P&D)

Dossiê de Pesquisa

Matheus Gonçalves de Souza

mathsouzag@gmail.com

RESUMO
Neste dossiê de pesquisa, está sendo apresentado um estudo para validar uma abordagem utilizando OCR, SLMs e RAG, em detrimento de melhorias para uma API com problemas de latência e custo, mantendo a acurácia. Por meio de levantamentos bibliográficos em repositórios acadêmicos e execução de benchmarks práticos para provar a efetividade da solução, ampliando também a visão sobre arquiteturas híbridas.

1 Introdução

A extração de documentos textuais e físicos sempre foi, além de uma prioridade, um desafio, sendo sua automatização uma prioridade para equipes de tecnologia. Foi estudada, assim, neste Dossiê de Pesquisa, uma API com duas etapas envolvendo LLM:

1. Interpreta e extrai conteúdos do documento;
2. Com uma inferência, converte os dados extraídos no formato JSON.

Dado o cenário de alta latência, principalmente em documentos longos (que exigem mais tempo e processamento) ou com layouts complexos, juntamente com o custo operacional de mais de uma chamada ao modelo, a solução disponível pode ser limitada, trazendo problemas para os usuários em processos que exigem resposta rápida, ou no processamento de grandes lotes de faturas.

A partir destes obstáculos, nota-se a necessidade de validar, identificar e transcrever diferentes abordagens para a redução de latências, custos e complexidade, e aumentar assim as possibilidades de resultados, mantendo a acurácia mínima. É também de grande importância a facilidade e utilização de ferramentas *open-source*. Tendo isso em vista, serão avaliadas soluções híbridas com a utilização baseada em OCR e Modelos de Linguagem, além de técnicas como o RAG, para documentos longos e complexos, diante do cenário de que cada tipo de documento pode funcionar melhor em um tipo de solução, trazendo detalhes como comparações métricas, juntamente com uma PoC executável (Prova de Conceito).

2 Metodologia

Os esforços deste Dossiê foram especialmente exploratórios e de caráter comparativo, identificando o maior número possível de informações e técnicas envolvendo a redução de latência e a extração de informações documentais. O Dossiê contém etapas bem definidas como o levantamento bibliográfico, análises e triagens de ferramentas e técnicas possíveis e, ainda, o uso de *benchmarks* e documentações técnicas.

Para a escrita e formatação deste dossiê foi utilizada a plataforma Overleaf, ferramenta com diversas opções de uso para documentos científicos em LaTeX, atendendo à preocupação com a consistência na entrega. Além disso, esta ferramenta facilitou a referenciação correta dos estudos.

2.1 Pesquisas e Repositórios acadêmicos

Primeiramente, o levantamento de repositórios técnicos e acadêmicos foi de extrema importância, trazendo alternativas como Google Scholar, arXiv, IEEE Xplore, Hugging Face, TechTarget e Github, combinando *strings* de busca específicas para abranger o maior número possível de artigos relacionados à otimização de SLMs/LLMs, as técnicas utilizadas para extração de informações e a formatação de saídas.

Com uma ampla gama de artigos retornados, o critério para seleção foi, a primeira vista, a leitura de seus títulos e resumos, observando-se assim quais realmente fariam sentido e seriam relevantes para resolver os problemas do escopo, priorizando-se, além disso, pesquisas com resultados comparativos, e eliminando-se artigos que fugiam logicamente do contexto de documentos, além de ferramentas que não são públicas.

2.2 Uso de Ferramentas de IA

Utilizei algumas ferramentas como Chat GPT e Gemini para correções de ortografias e coesões textuais, além da criação de strings de busca utilizando palavras chaves, que foram utilizados em repositórios de artigos. Para desenvolvimento, utilizei novamente o Chat GPT juntamente do Deepseek combinado a minhas experiências passadas na utilização de modelos de linguagem e RAG, garantindo que todo o fluxo de acontecimentos no código foram com base em minhas pesquisas e julgamentos.

3 Técnicas e Modelos Avaliados

Serão abordadas nas próximas seções algumas técnicas que podem ser promissoras na extração, juntamente com ideias descartadas.

3.1 Optical Character Recognition (OCR)

O Optical Character Recognition (OCR) tem-se mostrado como uma solução confiável e robusta para a automação de documentos, especialmente em tarefas específicas de amplo volume. Com base no artigo de Jones (2025), o OCR consegue extrair textos de imagens, mantendo sua estrutura e tornando-os editáveis para possíveis padronizações.

Além disso, o artigo Flatworld Solutions (2025) afirma a eficiência do processamento de dados, com reduções em erros humanos e com uma boa experiência do usuário final, garantindo uma solução promissora quando se trata do problema de conseguir as informações de um documento.

3.2 Retrieval-Augmented Generation (RAG)

Modelos de linguagem apresentam, frequentemente, um fenômeno conhecido como "alucinação", no qual retornam informações incorretas ou imprecisas sobre determinado assunto, extrapolando os dados nos quais foram treinados. Esse comportamento é comum a todos os modelos, variando de acordo com seu desenvolvimento, treinamento e arquitetura. Para mitigar esse problema, além das otimizações no próprio modelo, podem ser aplicadas técnicas como o RAG (Retrieval-Augmented Generation) Gupta (2025).

O RAG foi desenvolvido com o objetivo de melhorar a qualidade das respostas geradas por modelos de linguagem, tornando-as mais precisas e fundamentadas. Para isso, combina técnicas de geração de texto com mecanismos de recuperação de informações a partir de bases de dados ou documentos externos. Dessa forma, o modelo é capaz de acessar informações atualizadas e relevantes, resultando em respostas mais robustas e precisas. Essa abordagem tem se tornado amplamente utilizada em assistentes virtuais, chatbots e sistemas de perguntas e respostas Lewis et al. (2020).

Para a implementação do RAG, foram utilizadas algumas ferramentas durante a Prova de Conceito, como o framework LangChain empregado para manipular a integração com diversas fontes de dados, no caso deste projeto, o ChromaDB (banco de dados vetorial responsável por armazenar e buscar vetores de embeddings). Para a geração desses embeddings, foi utilizada a biblioteca reconhecida Hugging Face, que oferece modelos pré-treinados especialmente desenvolvidos para transformar textos em vetores.

3.3 Soluções desconsideradas

O fluxo com diversas chamadas de modelos, algo semelhante à solução atual do desafio técnico, foi evitado, tendo em vista seu alto tempo de resposta (devido às diversas chamadas do modelo), assim como o alto custo computacional. Além disso, seu custo está muito ligado ao tamanho de um arquivo, tornando-se inviável quando se trata de informações extensas e consultas.

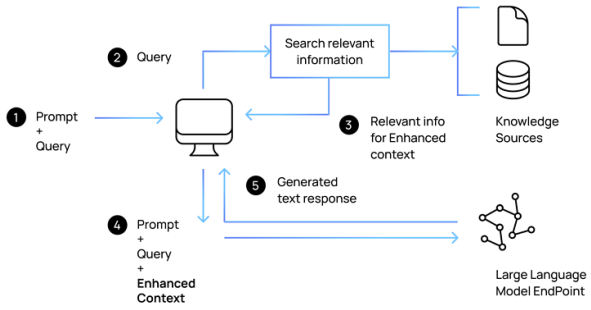

Figura 1: Exemplo de fluxograma RAG. (Fonte: https://eagerworks.com/blog/retrieval-augmented-generation).

4 Resultados e Experimentos

Nesta seção serão apresentados resultados obtidos através de testes de benchmark realizados localmente, juntamente com dados de artigos, com o objetivo de analisar sua viabilidade em uma aplicação real. O intuito aqui é mostrar opções de OCR e comprovar que é uma solução funcional. Paralelo a isso, serão testados diversos modelos, avaliando sua acurácia e tempo.

4.1 OCR

Existem diversas ferramentas de OCR, entre elas estão algumas como Amazon Textract, Google Document AI, Tesseract, Claude Sonnet 3.7 e Mistral OCR. Para uma avaliação precisa entre elas, foram pesquisados artigos com dados visuais para uma melhor interpretação, com destaque principal para o Tesseract, sendo o único OCR *open-source*. O artigo de Heggham-mer (2022) traz métricas como acurácia de extração, custo, modelo de implantação e complexidade, especificamente do Tesseract, Amazon Textract e Google Document AI, que são amplamente conhecidos.

Este gráfico da Figura 2 demonstra de forma clara como se saiu a efetividade dos OCRs e como eles podem reagir conforme os documentos estejam em baixa qualidade, o chamado "ruído". O eixo X representa a taxa de erro por palavra, logo, quanto mais próximo da esquerda, melhor é sua acurácia.

O Google Document AI (azul) mostra-se o mais preciso em todos os cenários; o Tesseract (vermelho) se destaca em documentos "limpos", apresentando pioras drásticas com o aumento dos ruídos, e o Inglês se mostra mais eficiente que o arábico. Já o Amazon Textract apresenta-se como um OCR equilibrado, com base no gráfico.

Já este gráfico da figura 3 retrata qual tipo de ruído afeta cada ferramenta de OCR, evidenciando grandes pontos fracos do Tesseract, desde arquivos com cores a *blur* no documento, mas retrata também seu grande ponto forte: arquivos pretos e brancos e binarizados. Isso evidencia uma etapa muito importante ao utilizá-lo: fazer um pré-processamento para escala de cinza, juntamente com um filtro de binarização para pixels pretos e brancos, e assim enviar ao Tesseract. Pode-se, assim, converter seu pior cenário (todo colorido) em melhor cenário.

4.2 Benchmark RAG

Como avaliação de modelos na extração e busca de informações, foi utilizado um RAG, com possibilidades para altera-

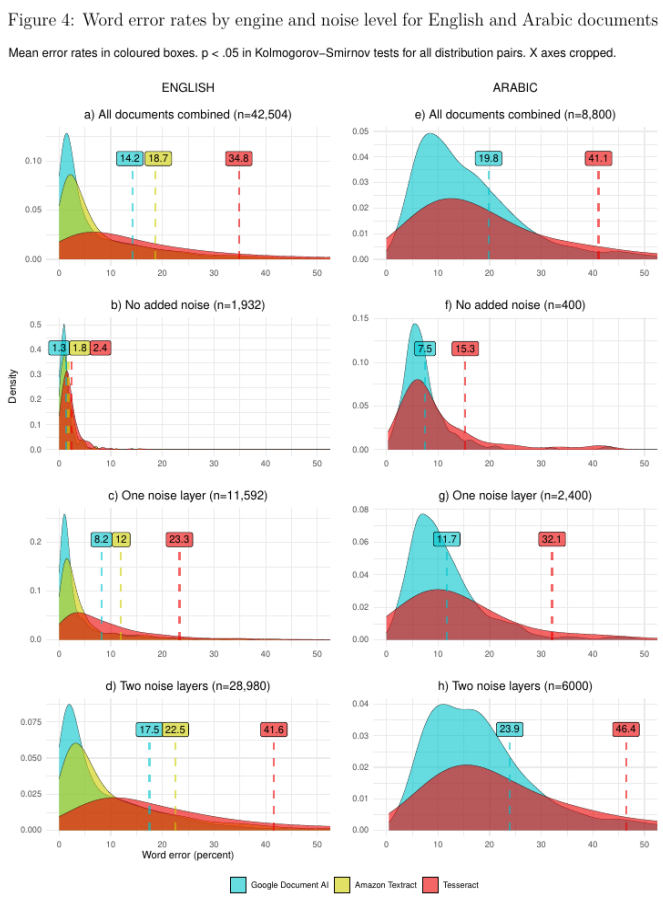

Figura 2: Taxas de erro de palavras por mecanismo de busca e nível de "sujeira"para documentos em inglês e árabe. (Fonte: Hegghammer (2022)

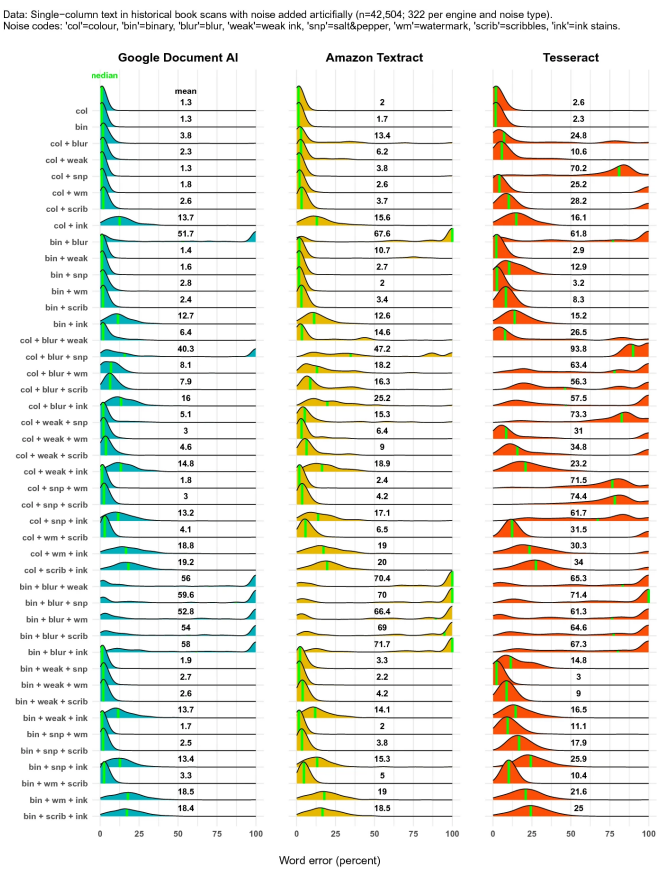

Figura 3: Taxas de erro de palavras por mecanismo de busca e tipo de ruído em documentos em inglês. (Fonte: Hegghammer (2022)

ções de parâmetros, medindo-se métricas como tempo, custos computacionais e acurácia. Como o objetivo é apenas testar os modelos, os seguintes parâmetros foram padronizados:

- Chunk Size: 500, Overlap: 100 (otimização da recuperação de texto).
- Modelo de embedding *mxbai-embed-large-v1*, através do *Hugging Face*.
- *Retriever* MMR, devido ao seu equilíbrio semântico, com *k=1* (chunk mais relevante, maior agilidade), e *scoreth-reshold=0.25*, para não ter contextos com baixa relevância.

Com a entrada do arquivo Excel, contendo 20 perguntas do artigo "The Claude 3 Model Family: Opus, Sonnet, Haiku"disponibilizado no Caso 2, cada uma delas contendo quatro alternativas e, na coluna ao lado, sua alternativa correta, que será utilizada para medir a taxa de acerto de cada um dos modelos.

O benchmark terá como prioridade SLMs de até 8 bilhões de parâmetros e com atualizações nesse último ano, visto que, através dos testes, eles se mostraram eficazes na extração, busca e formatação de informações. A implementação utiliza a infraestrutura LangChain, que faz as etapas de recuperação e geração, enquanto o Chroma é o banco vetorial, onde armazena e busca as informações semânticas.

Foram selecionadas três famílias com grandes reputações, todos eles executados via **Ollama**. Os modelos incluídos são:

- Llama 3.2:1B, Llama 3.2:3B e Llama 3.1:8B (Meta).
- Gemma 3:1B, Gemma 3:4B e Gemma 2:2B (Google).
- Phi 3:3.8B e Phi 3.5:3.8B (Microsoft).

Com a execução, para cada uma das perguntas uma série de métricas são medidas, como tempo em segundos, porcentagem de uso do CPU e memória RAM (antes e depois de cada pergunta), com a biblioteca psutil. Após a resposta do modelo, para avaliação é utilizada a similaridade cosseno, que é calculada através dos embeddings das frases obtidos com o sentence-transformer "all-MiniLM-L6-v2", o que mostra a proximidade entre as respostas com um valor entre 0 e 1.

Com a variação dos modelos, é possível analisar e identificar até mesmo padrões entre as famílias, o que define assim a melhor escolha com base no objetivo do projeto. Interpretando os gráficos 4 e 5, podemos observar um *trade-off* em relação a latência e acurácia das respostas.

4.2.1 Análise de Tempo de Resposta (Latência)

O gráfico 4, mostra quantos segundos, em média, os modelos levam para gerar uma resposta neste cenário.

Como esperado, quanto menos parâmetros, maior foi a velocidade, sendo o `gemma3:1b` o mais ágil (9,49 segundos), seguido pelo `llama3.2:1b` (10,5 segundos), enquanto entre os mais lentos estão o `llama3.1:8b` e `phi3.5:3.8b`, respectivamente.

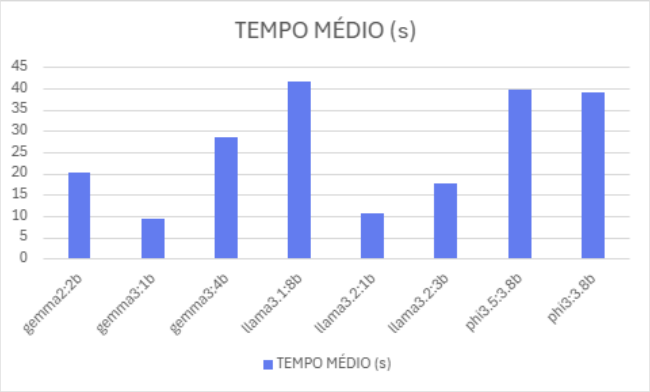

Figura 4: Benchmark SLMs - Tempo Médio (s)

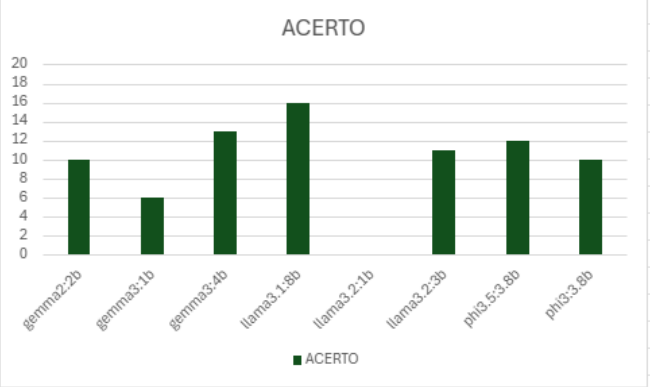

Figura 5: Benchmark SLMs - Acerto (20 perguntas)

4.2.2 Análise de Acurácia

O gráfico 5 exibe como os modelos reagem ao mesmo RAG (documento do caso 3), respondendo a 20 perguntas com 4 alternativas.

Conseguimos notar que o `llama3.1:8b` teve maior efetividade, com 16 das 20 perguntas corretas, com `gemma3:4b` e `phi3.5:3.8b` como intermediários com 13 acertos, seguidos pelos `phi3.5:3.8b` e `llama3.2:3b` com 12 e 11 acertos, respectivamente; o `gemma3:1b`, que obteve 6 acertos, só não foi pior que o modelo `llama3.2:1b`, que não foi eficaz em acertar nenhuma das perguntas.

4.2.3 Conclusão do Benchmark

Diante dos dados exibidos, a hipótese inicial do *trade-off* se confirma, já que os modelos mais precisos são também os mais lentos, enquanto os menos assertivos são os mais rápidos. Logo, a escolha ideal seria o equilíbrio entre estas duas métricas, com acurácia razoável (acima de 50%), com tempo de resposta consideravelmente menor em relação ao modelo de 8B.

5 Conclusão e Recomendação

Durante esta conclusão, será mostrado o principal fluxo recomendado, com base no processamento de documentos, considerando o problema central de latência, acurácia e custo operacional com o uso de ferramentas *open-source* e *self-hosted* para maior controle. Diante dos experimentos e dados coletados, as soluções híbridas mostram-se mais estratégicas, unindo assim OCR com SLMs/LLMs de uma forma robusta, conforme as análises realizadas.

5.1 Arquitetura Recomendada

Diante do contexto da diminuição de chamadas ao modelo, é proposta uma solução híbrida, abrangendo tecnologias consolidadas juntamente com modelos de linguagem atuais *open-source*, que podem variar conforme o tipo de tarefa (extração ou consulta de conteúdo):

- No contexto de **OCR**: Pode ser utilizado para Imagens e PDFs Digitalizados, com a utilização do Tesseract, pois, como mostrado pelo gráfico da Figura 2, seus pontos fracos (cores e ruídos) podem ser mitigados, além de ser uma ferramenta *open-source*.
- Para **Formatação e Compreensão**: Recomenda-se a utilização de `llama3.2:3b` ou `gemma3:4b`, por meio do Ollama. Ele foi selecionado com base no benchmark feito localmente e em artigos próprios do Ollama (2025).
- Para **Documentos Extensos**: A partir da classificação, o RAG pode ser um grande aliado na extração precisa de informações do documento e, aliado a bibliotecas para entendimento de tabelas e imagens, consegue abordar assim todo o conteúdo dentro de um documento.

Para o uso correto do SLM, a separação do *prompt* é altamente recomendada, já que quanto maior a quantidade de informações dentro dele, maior a chance de confundir o modelo, sendo assim a classificação um forte aliado na formatação final da resposta, na PoC por exemplo, foi separado em "CNH", "FATURA"e "DOCUMENTO", mantendo a precisão esperada.

5.2 Justificativa e Análise de Trade-offs

Tendo em vista os problemas centrais abordados anteriormente, esta arquitetura também abre espaço para otimizações futuras, com novos modelos sendo desenvolvidos diariamente e melhorias no OCR, e, com base nos *trade-offs*, é importante ressaltar os seguintes pontos:

- A escolha do modelo foi feita com base no balanceamento das métricas, sendo também *open-source* e de fácil utilização, permitindo assim o uso em nuvens de baixo custo, ou na própria infraestrutura. A prioridade aqui foi a diminuição do tempo de resposta, com a maior acurácia possível, o que não exige modelos de ponta.
- A respeito dos OCRs, com a escolha do Tesseract, que é gratuito, em vez do Google Document AI, que é pago, são necessárias etapas extras para o pré-processamento. A PoC utiliza filtros em escala cinza, por exemplo, para a transição para seu melhor cenário, podendo também ser implementadas outras ferramentas para melhorias da imagem, como a remoção de *blur*.
- Sobre a alta complexidade textual, o RAG foi priorizado devido à sua agilidade na recuperação de informações e devido à sua vetorização ser feita apenas uma única vez, o que confirma assim sua velocidade e otimização.

5.2.1 Sugestões de melhorias

Como observado anteriormente, afim de simplificar e garantir uma boa qualidade de código, podemos classificar os documentos de maneira externa, como a nomeação dos arquivos enviados de acordo com o campo requisitado, eliminando assim uma das etapas da PoC. Outra questão seria a melhoria da visualização da imagem, utilizando bibliotecas como *open-cv*, assegurando a exata informação.

Como observado anteriormente, a fim de simplificar e garantir uma boa qualidade de código, podemos classificar os documentos de maneira externa, como a nomeação dos arquivos enviados de acordo com o campo requisitado, "cnh-123.png"por exemplo, o que elimina assim uma das etapas da PoC. Outra questão seria a melhoria da visualização da imagem, pelo uso de bibliotecas como *OpenCV*, o que assegura a exata informação dos documentos.

Finalmente, o OCR deve também ser integrado juntamente na extração dos PDFs, por meio de bibliotecas de conversão, fundindo assim a técnica de RAG e extração de imagens. Algo importante a ressaltar sobre o futuro é a existência de modelos multimodais *open-source*, que conseguem processar texto e imagem, sendo também soluções robustas que podem se sobressair com atualizações futuras.

6 Análise de Viabilidade de Integração:

O uso do modelo Llama 3.2:3B em infraestrutura local é viável. Seus 3 bilhões de parâmetros permitem a execução em GPU com resultados satisfatórios, exigindo cerca de 16 GB de VRAM, e funcionando em 8 GB de VRAM (como a utilizada neste computador). O aumento da VRAM (ou seja, o uso de GPUs mais potentes) pode melhorar sua velocidade.

Na nuvem, é possível alugar GPU equivalente a 24 GB VRAM variando entre US $ 0,38 e US $ 14,24 por hora na AWS, considerando execução por 8 horas diárias, trata-se de aproximadamente US $ 94,24 a US $ 3.531,52 mensais, segundo o site GetDeploying (2025).

Agora, falando a respeito do OCR Tesseract, por ser *open-source* e gratuito, seu custo seria apenas o de processamento, o qual se mostrou bem ágil e pouco custoso nos testes da PoC. No total, em hardware local o custo seria praticamente nulo, enquanto na nuvem seria necessária a contratação para seu processamento.

Referências

Flatworld Solutions (2025). 9 key advantages of ocr-based data entry.

GetDeploying (2025). Gpu price comparison [2025]. https://getdeploying.com/reference/cloud-gpu.

Gupta, S. (2025). Retrieval-augmented generation and hallucination in large language models: A scholarly overview. *Scholars Journal of Engineering and Technology*.

Hegghammer, T. (2022). Ocr with tesseract, amazon textract, and google document ai: a benchmarking experiment. *Journal of Computational Social Science*, 5(1).

Jones, J. (2025). 7 advantages of ocr for data entry.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuksa, P., Min, S., Yih, W.-t., Richardson, M., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Ollama (2025). Ollama library.