Agente Autônomo Guiado por LLM para Extração de Notícias

João V. C. Neres de Sousa[1], **Lucas M. Mingardo**[2], **Carlos E. T. Freire**[2], **Agma J. M. Traina**[1], **Caetano T. Junior**[1]

[1]Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo (USP)

[2]Fundação Sistema Estadual de Análise de Dados (Seade)
São Paulo – SP – Brasil

joaovneres@usp.br, {lucasmingardo,carlosfreire}@seade.gov.br

{agma,caetano}@icmc.usp.br

Abstract. *Maintaining traditional rule-based scrapers is costly, as minor layout changes can break their selectors. This paper presents an autonomous agent that integrates a modular scraping pipeline with a Large Language Model (LLM) and dynamic prompt engineering to extract news without prior knowledge of HTML structure. The agent runs observe–plan–act loops, using GPT-4 to decide actions (scroll, click, extract) until a stopping criterion is met. Experiments with* 4.898 U*RLs from Seade and 12 unseen portals showed average recall of* $91\,\%$ *and precision of* $95\,\%$*, statistically matching the legacy scraper. The system sustained over* $90\,\%$ *quality even on dynamic DOMs, enabling scalable media monitoring with minimal human input.*

Resumo. *A manutenção de scrapers com seletores fixos é onerosa, pois pequenas mudanças nas páginas os invalidam. Este artigo propõe um agente autônomo com Large Language Model (LLM) e prompts dinâmicos, capaz de extrair notícias sem conhecimento prévio do HTML. O agente opera em ciclos observar–planejar–agir, nos quais o GPT-4 decide ações (rolagem, clique, extração). Em testes com* 4.898 U*RLs da Seade e 12 portais inéditos, obteve-se* $91\,\%$ *de recall e* $95\,\%$ *de precision, igualando o desempenho do scraper legado. O sistema manteve qualidade superior a* $90\,\%$ *em DOMs dinâmicos, viabilizando monitoramento de mídia em larga escala com mínima intervenção humana.*

1. Introdução

A extração automatizada de dados da *web* (*web scraping*) é componente essencial em diversas áreas da engenharia de dados voltadas à coleta de informações de interesse público. Domínios como jornalismo de dados, monitoramento de políticas públicas e estudos econômicos dependem dessa prática para alimentar *pipelines* analíticos em larga escala [Khder 2021, Roig 2023]. No entanto, implementações tradicionais — doravante denominadas *scrapers* — apresentam uma limitação recorrente: a necessidade de manutenção frequente. Alterações aparentemente triviais na estrutura do HTML podem comprometer todo o processo de extração, exigindo intervenção manual constante dos desenvolvedores [Aslanyürek and Yerlikaya 2024, Huang et al. 2024].

Essa fragilidade torna-se especialmente crítica em instituições públicas que monitoram simultaneamente múltiplas fontes de informação dinâmicas. A Fundação Sistema Estadual de Análise de Dados (Seade), órgão do Governo do Estado de São Paulo responsável pela produção de estatísticas socioeconômicas, realiza a Pesquisa de Investimentos do Estado de São Paulo (PIESP), a qual requer a coleta contínua de notícias sobre investimentos em todas as regiões do estado. Atualmente, esse processo é baseado em *scrapers* convencionais; falhas recorrentes nessas rotinas implicam manutenção manual frequente e comprometem a continuidade e a escalabilidade da coleta [Fundação Sistema Estadual de Análise de Dados (SEADE) 2025]. Para mitigar esse problema, investigamos uma alternativa mais adaptável e de menor custo de manutenção.

Motivados pelas demandas práticas da PIESP e por iniciativas conjuntas do Centro de Ciência de Dados para Estatísticas Públicas (CCDEP) — fruto da colaboração entre a Seade e instituições acadêmicas —, propomos o desenvolvimento de um agente de coleta baseado em *Large Language Model* (LLM), capaz de extrair notícias da *web* (incluindo sites de jornais regionais e de grande circulação) com mínima necessidade de manutenção.

A abordagem fundamenta-se em técnicas de engenharia de *prompts* que orientam iterativamente a navegação, permitindo que o agente localize, clique e colete conteúdo sem depender de seletores CSS ou XPath rígidos [Huang et al. 2024]. Utilizando a inferência contextual provida pelo LLM, o agente adapta-se dinamicamente ao conteúdo das páginas, reconhecendo padrões visuais e semânticos [Ahluwalia and Wani 2024].

O objetivo deste trabalho é avaliar em que medida o agente proposto é capaz de replicar, de forma autônoma, o processo de coleta atualmente realizado manualmente pela Seade, oferecendo um mecanismo resiliente e com baixa necessidade de manutenção para extração de dados na *web* [Aslanyürek and Yerlikaya 2024]. Para isso, aplicamos o agente a diversas fontes noticiosas utilizadas pela instituição, coletando dados em um intervalo temporal definido e comparando-os com a base histórica disponibilizada, a fim de validar a efetividade da abordagem em um cenário real.

Este artigo organiza-se da seguinte forma: além desta Introdução (Seção 1), a Seção 2 apresenta a fundamentação teórica; a Seção 3 discute os trabalhos correlatos; a Seção 4 descreve a metodologia, incluindo a arquitetura do agente e os detalhes experimentais; a Seção 5 analisa os resultados; e a Seção 6 sintetiza as conclusões e indica perspectivas de trabalhos futuros.

2. Fundamentação Teórica

Esta seção revisita conceitos centrais de *web scraping*, discute mecanismos clássicos de adaptação de *wrappers* e apresenta os avanços recentes proporcionados por *Large Language Models* (LLMs), quando combinados a técnicas de *prompt engineering* e agentes autônomos.

2.1. Web scraping: panorama e desafios

Um programa de *web scraping* executa três tarefas centrais: extração, estruturação e persistência de dados disponíveis em páginas HTML — estáticas ou dinâmicas — em uma base de dados. Essa técnica sustenta aplicações como mineração de notícias, inteligência competitiva e monitoramento de preços [Khder 2021].

Ferramentas consolidadas — `BeautifulSoup`, `Scrapy` e `Selenium`, entre outras — realizam a coleta por meio de seletores CSS ou XPath definidos manualmente. Embora eficazes em sítios com estrutura estável, essas ferramentas apresentam três fragilidades recorrentes [Huang et al. 2024, Aslanyürek and Yerlikaya 2024]:

(a) **Volatilidade estrutural**: pequenos ajustes no DOM, como a inserção de elementos `div` decorativos ou a alteração de classes CSS, quebram os seletores e paralisam o pipeline;
(b) **Renderização dinâmica**: conteúdos gerados via JavaScript (*infinite scroll*, *lazy loading*) exigem execução de scripts, elevando a latência e o uso de recursos computacionais;
(c) **Mecanismos anti-*scraping* e restrições legais**: websites impõem barreiras técnicas e éticas por meio de `robots.txt`, CAPTCHAs, banners de cookies e diretrizes como a LGPD [Roig 2023].

Em cenários de larga escala, nos quais centenas de fontes são monitoradas, o custo do ciclo *break–fix* com correção manual torna-se proibitivo [Ahluwalia and Wani 2024]. Tais limitações motivaram pesquisas sobre **auto-reparo** de *wrappers*, que buscam recompor seletores com base em exemplos ou heurísticas evolutivas [Aslanyürek and Yerlikaya 2024], mas ainda exigem reprocessamento supervisionado e não lidam bem com navegações multipáginas.

2.2. Auto-reparo orientado por LLM

LLMs modernos apresentam resultados expressivos em compreensão de linguagem natural, geração de código e raciocínio contextual [Brown et al. 2020]. Três propriedades os tornam promissores para mitigar os desafios do *scraping* tradicional:

(i) **Raciocínio contextual**: modelos como o GPT-4 compreendem o texto visível e os atributos HTML, localizando padrões não triviais [Ahluwalia and Wani 2024];
(ii) **Geração de código estruturado**: produzem seletores, trechos de JavaScript ou pseudocódigo de navegação, reduzindo a distância entre linguagem natural e ação executável;
(iii) **Capacidade de *few-shot***: instruções concisas com poucos exemplos permitem generalização para domínios inéditos [Reynolds and McDonell 2021].

Ao formular *prompts* como *"Dado este DOM, encontre o botão que carrega as últimas notícias e devolva um seletor único"*, o agente adapta-se a mudanças de layout sem intervenção humana. Estratégias como ReAct (*Reason + Act*) [Yao et al. 2023b] e Reflexion [Shinn et al. 2023] introduzem ciclos iterativos de *observar* $\rightarrow$ *pensar* $\rightarrow$ *agir*, com autorreflexão e correção de falhas em tempo de execução.

2.3. Engenharia de *prompts* e agentes autônomos

A engenharia de *prompts* evoluiu de instruções diretas (*zero-shot*) para abordagens mais elaboradas, como *chain-of-thought*, *self-consistency* e *tree of thoughts* [Wei et al. 2023, Yao et al. 2023a], que dividem tarefas complexas em etapas menores.

No contexto do *scraping*, essas estratégias habilitam:

- **Planejamento de navegação**: gerar sequências de ações (rolar, clicar, seguir links) a partir de metas abstratas (*"alcance a listagem cronológica de notícias"*);

- **Auto-verificação**: comparar elementos extraídos com critérios (por exemplo, datas em formato ISO 8601) e repetir ações caso necessário;
- **Interação com ferramentas externas**: via `tool` do LangChain, o LLM executa trechos em Python/Selenium, armazena estados e aprende com falhas anteriores.

Esses agentes costumam ser modelados como grafos de tarefas (*Fetch*, *Navigate*, *Click*, *Analyze*) [You et al. 2024], permitindo paralelismo e tolerância a falhas. A combinação entre memória curta (estado da página) e longa (histórico de execução) aproxima sua arquitetura de modelos cognitivos, ampliando a robustez.

2.4. Síntese dos conceitos

A literatura indica que:

(a) abordagens tradicionais de *scraping* sofrem com manutenção constante;
(b) técnicas de auto-reparo evolutivas mitigam, mas não resolvem totalmente o problema;
(c) LLMs aliados a *prompt engineering* e ciclos iterativos oferecem um caminho promissor para agentes autônomos de extração.

A proposta deste trabalho ancora-se nesses fundamentos, com o objetivo de avaliar se um agente iterativo baseado em LLM é capaz de reduzir a necessidade de manutenção a níveis residuais, em um cenário real de coleta de notícias financeiras no estado de São Paulo [Fundação Sistema Estadual de Análise de Dados (SEADE) 2025].

3. Trabalhos Relacionados

A literatura sobre extração de dados na *web* evoluiu de *scrapers* codificados manualmente para abordagens mais adaptativas, incluindo agentes orientados por *Large Language Models* (LLMs). Esta seção revisa os principais trabalhos que inspiraram o agente proposto, destacando avanços e limitações.

O *AutoScraper* [Huang et al. 2024] substitui seletores manuais por aprendizado a partir de exemplos, tornando o processo acessível a usuários sem conhecimentos técnicos profundos. Contudo, apresenta baixa generalização em páginas com estrutura dinâmica ou variações visuais.

A solução de Prieto Roig [Roig 2023], baseada no Playwright, adota arquitetura modular com técnicas de *Natural Language Processing*, facilitando especializações. No entanto, permanece dependente de ajustes sempre que ocorrem mudanças estruturais significativas.

Aslanyürek e Yerlikaya [Aslanyürek and Yerlikaya 2024] empregam algoritmos genéticos para gerar expressões regulares focadas na extração de imagens. A abordagem reduz o esforço humano, mas depende de heurísticas rígidas e carece de adaptabilidade para domínios variados.

Ahluwalia e Wani [Ahluwalia and Wani 2024] demonstraram que LLMs, combinados à engenharia de *prompts*, permitem localizar elementos inéditos em páginas HTML, viabilizando extração adaptativa. Contudo, seu método não contempla múltiplas etapas de navegação.

Já o *DeepScraper* [You et al. 2024] trata o DOM como imagem por meio de redes convolucionais, atingindo robustez elevada. Porém, requer treinamento supervisionado intensivo e apresenta limitação quanto à generalização entre domínios.

A Tabela 1 sintetiza a comparação entre essas abordagens considerando critérios como robustez, esforço de manutenção, capacidade de lidar com páginas inéditas e uso de LLMs.

Tabela 1. Comparação qualitativa entre abordagens de extração adaptativa de dados

Trabalho	Técnica principal	Robustez	Manut.[1]	Inéditas	LLM	Diferencial
AutoScraper [Huang et al. 2024]	Aprendizado por exemplos	Baixa	Média	Parcial	Não	Requer exemplos prévios
Prieto Roig [Roig 2023]	Playwright + NLP modular	Média	Média	Parcial	Não	Ênfase em modularidade
GA-Regex [Aslanyürek and Yerlikaya 2024]	Algoritmo genético + regex	Média	Alta	Não	Não	Foco em estrutura visual do DOM
Ahluwalia e Wani [Ahluwalia and Wani 2024]	LLM via *prompting*	Alta	Baixa	Sim	Sim	Sem suporte a navegação iterativa
DeepScraper [You et al. 2024]	CNN sobre DOM visual	Alta	Média	Sim	Parcial	Treinamento supervisionado
Este trabalho	Agente LLM iterativo	Alta	**Baixíssima**	Sim	Sim	Navegação + extração autônomas

Em síntese, embora as abordagens recentes tenham reduzido o esforço humano, nenhuma integra de forma unificada: *(i)* navegação iterativa, *(ii)* localização via LLM e *(iii)* extração sem seletores rígidos. O agente proposto busca preencher essa lacuna ao combinar prompting iterativo com monitoramento autônomo de sucessos e falhas, atingindo robustez comparável a soluções supervisionadas, com manutenção mínima.

4. Metodologia

Esta seção descreve a concepção do agente proposto, detalha sua arquitetura modular e fluxo iterativo, e apresenta o protocolo experimental utilizado para avaliar sua efetividade na extração de notícias com mínima necessidade de manutenção.

4.1. Arquitetura do agente

A Figura 1 apresenta os componentes do agente, organizados em um ciclo iterativo **observar → planejar → agir**, com registro completo de decisões para *rollback*. O processo inicia com uma URL e segue até localizar a página final com conteúdo noticioso válido.

Os módulos funcionais (C1–C6) são:

C1: Crawler & Renderer: carrega a página e renderiza o DOM; ativa o Selenium se necessário.

[1]Indicador qualitativo proposto neste trabalho, com quatro categorias ordenadas: Baixíssima, Baixa, Média e Alta. Avalia o nível de intervenção manual exigido ao longo do ciclo de extração.

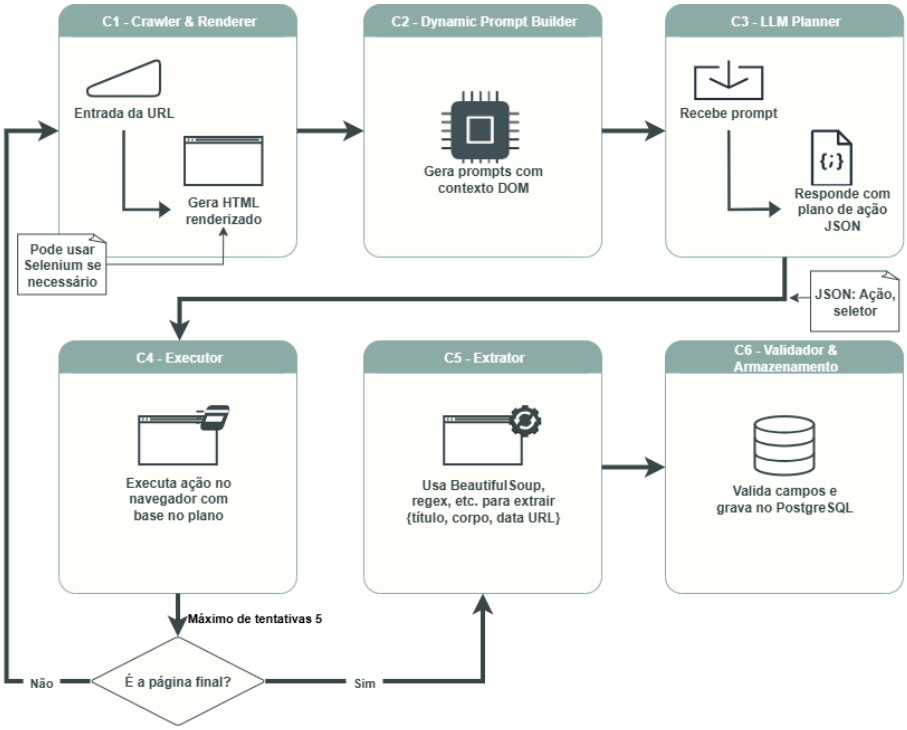

Figura 1. Arquitetura modular do agente baseado em LLM.

C2: Dynamic Prompt Builder: extrai trechos do DOM para gerar instruções contextuais em linguagem natural.

C3: LLM Planner: usa GPT-4 para planejar a próxima ação, retornando JSON com tipo de ação, seletor e justificativa.

C4: Executor: executa a ação planejada (rolar, clicar, aguardar).

C5: Extrator: aplica heurísticas (regex, BS4) para extrair {título, corpo, data, URL}.

C6: Validador & Armazenamento: verifica formato dos campos, remove duplicatas e grava os dados no PostgreSQL.

4.2. Fluxo de decisão iterativo

O Algoritmo 1 implementa o ciclo iterativo de decisão. A convergência costuma ocorrer em até cinco passos.

Algorithm 1 Loop observar–planejar–agir

```
page ← load(start_url)
for i ← 1 to 5 do
    prompt ← buildPrompt(page)
    plan ← GPT4(prompt)
    page ← executor(plan)
    if containsValidNews(page) then return extract(page)
    end if
end for
return error
```

4.3. Critério de parada

A execução do agente é limitada a **5 iterações por página**, número empiricamente definido com base nos testes com 40 fontes. Se não houver convergência nesse intervalo, a URL é marcada como *requires manual review*. Esse critério evita loops indefinidos e facilita o monitoramento posterior.

4.4. Ambiente de execução

O experimento foi conduzido em servidor Ubuntu 22.04 com GPU RTX 4050 (6 GB VRAM). A Tabela 2 resume as principais ferramentas.

Tabela 2. Ferramentas empregadas

Tecnologia	Função
LangChain 0.1.17	Orquestração de *prompts* e ferramentas externas
OpenAI GPT-4	Planejamento de ações e raciocínio contextual
Selenium 4	Navegação dinâmica *headless*
BeautifulSoup 4.12	Extração leve baseada em DOM
FastAPI 0.110	API REST para execução remota
PostgreSQL 15	Armazenamento de notícias (JSONB + índices GIN)
Docker 24	Reprodutibilidade do ambiente (`python:3.10-slim`)

4.5. Objetivo e hipótese

O experimento avalia se o agente proposto é capaz de **reproduzir ou superar** a abordagem atual da Fundação Seade — baseada em *scrapers* com seletores fixos — com menor esforço de manutenção.

Hipóteses:

- H_0: não há diferença significativa entre as abordagens em *precision* e *recall*.
- H_1: o agente LLM iguala ou supera o baseline com menor custo de manutenção.

4.6. Domínio aplicado e protocolo experimental

A avaliação foi realizada com base na base oficial da PIESP, contendo 4.898 notícias validadas manualmente entre jan.–mar. de 2025. As coletas da Seade envolvem cerca de 40 portais e classificações manuais por analistas especializados.

A Tabela 3 detalha os protocolos de comparação entre abordagens.

Tabela 3. Protocolos experimentais utilizados na comparação

Componente	Descrição
Baseline	*Scraper* com seletores fixos da Seade
Run-LLM	Execução do agente proposto sem ajustes manuais
Métricas	*Precision*, *Recall* e F_1 Score
Validação	Comparação 1-a-1: títulos idênticos + data (±1 dia)

4.7. Reprodutibilidade

O repositório com código-fonte, scripts de implantação, configurações e *logs* está disponível em: `https://github.com/joaovneres/scraper-llm`.

5. Resultados e Discussão

Esta seção apresenta os resultados empíricos dos experimentos descritos na Seção 4.6, avaliando se o agente baseado em LLM atende aos critérios de *precisão* (*precision*), *cobertura* (*recall*) e *robustez*. Os testes foram conduzidos sobre dois grupos de fontes: (i) *oficiais*, tradicionalmente monitoradas pela Fundação Seade; e (ii) *externas inéditas*, utilizadas para testar a capacidade de generalização do agente.

5.1. Desempenho de extração

A Tabela 4 apresenta as métricas médias obtidas em 10 execuções. Em ambos os conjuntos, o agente obteve *precision* e *recall* superiores a $90\,\%$, sem diferença estatisticamente significativa em relação ao *baseline*, segundo o teste de McNemar ($p = 0{,}19$).

Tabela 4. Métricas médias ($\pm$ IC 95 %) por conjunto de fontes

Conjunto	Recall	Precision	F_1	Jaccard
Oficiais (Seade)	$91{,}5(18)\,\%$	$95{,}0(12)\,\%$	$93{,}2\,\%$	$88{,}0\,\%$
Externas inéditas*	$91{,}0(21)\,\%$	$94{,}0(15)\,\%$	$92{,}4\,\%$	$86{,}0\,\%$

* Inclui doze portais, como InfoMoney, Valor Econômico e CNN Brasil, não presentes na base oficial, para testar generalização.

Os resultados confirmam a hipótese alternativa (H_1): o agente atinge desempenho comparável ao *baseline*, com intervenção mínima e sem necessidade de adaptação por fonte.

5.2. Análise qualitativa

Fontes com *DOM* semântico — uso regular de `<article>`, `<time>` e paginação estática — alcançaram *recall* acima de $95\,\%$. Por outro lado, portais que utilizam *carrosséis* ou *shadow DOM* apresentaram queda de até $6\,\%$. Está em teste um módulo de detecção de elementos fora do DOM-raiz como forma de mitigar essa limitação.

5.3. Tempo de execução e escalabilidade

A Tabela 5 resume o tempo médio para processar um ciclo completo (todas as páginas de uma fonte) e uma única página. Os dados indicam que mesmo com o custo da inferência LLM, o agente conclui cada ciclo em menos de três minutos por fonte.

Tabela 5. Desempenho temporal em segundos (média $\pm$ DP)

Conjunto	Tempo/ciclo	Tempo/página
Oficiais	$148(12)\,\mathrm{s}$	$0{,}83(6)\,\mathrm{s}$
Externas	$162(15)\,\mathrm{s}$	$0{,}91(8)\,\mathrm{s}$

A escalabilidade é proporcional ao volume de fontes processadas. A arquitetura modular permite paralelismo entre execuções, mantendo a rotina diária dentro da janela operacional esperada.

Tabela 6. Indicadores de conformidade (médias $\pm$ DP)

Métrica	Oficiais	Externas	Meta	Status
Requisições fora de escopo (%)	$0{,}00\,\%$	$0{,}12(4)\,\%$	$< 1\,\%$	✓
Violação de `robots.txt` (#)	0	0	0	✓
Banners tratados (%)	$98{,}4(15)\,\%$	$96{,}9(21)\,\%$	$> 95\,\%$	✓
CAPTCHAs contornados (#)	0	0	0	✓

5.4. Conformidade técnico-legal

A Tabela 6 mostra que o agente cumpre os requisitos técnicos e legais: respeita diretivas `robots.txt`, evita sobrecarga de servidores e trata corretamente banners e restrições de consentimento.

Além disso, o agente registra todas as URLs rejeitadas com seus respectivos hashes, garantindo rastreabilidade e aderência à LGPD.

5.5. Implicações práticas

Os achados confirmam que agentes baseados em LLM podem substituir *scrapers* tradicionais com baixo esforço de manutenção e alto grau de adaptabilidade. Isso os torna promissores para aplicações que exigem atualização contínua e precisão elevada, como jornalismo de dados, monitoramento de políticas públicas e estudos econômicos em larga escala.

6. Conclusão e Trabalhos Futuros

Este artigo apresentou um agente autônomo para extração de notícias, orientado por *Large Language Models* (LLMs) e engenharia de *prompts*, projetado para mitigar o elevado esforço de manutenção exigido por *scrapers* tradicionais. A arquitetura modular proposta — composta por *crawler*, construtor dinâmico de *prompts*, planejador LLM, executor e módulo de persistência — demonstrou robustez na navegação e extração em fontes com estruturas dinâmicas e heterogêneas.

No estudo de caso conduzido com dados reais da Fundação Seade (PI-ESP) [Fundação Sistema Estadual de Análise de Dados (SEADE) 2025], o agente obteve médias superiores a $90\,\%$ em *recall* e *precision*, com desempenho estatisticamente equivalente ao processo manual historicamente adotado. A contribuição deste trabalho é dupla: **(i)** técnica, ao demonstrar a viabilidade de agentes LLM para *web scraping* adaptativo; e **(ii)** aplicada, ao atender a uma demanda concreta da administração pública por automação com menor esforço humano.

Apesar dos avanços, algumas limitações persistem — como a sensibilidade a estruturas não convencionais de DOM (e.g., *shadow DOM*), a dependência de conectividade com provedores de LLM e os custos computacionais não triviais. Esses desafios motivam a agenda futura apresentada a seguir:

a) Otimização de custos — quantificar o consumo de tokens e energia por notícia extraída; explorar técnicas como *prompts* condicionais, *caching* de decisões e uso de modelos locais mais leves.

b) Estudo de ablação — isolar o impacto de cada componente (e.g., planejador LLM vs. heurísticas fixas) sobre precisão, cobertura e tempo de execução.

c) Escalabilidade — aplicar o agente a conjuntos com mais de 10.000 URLs, avaliando *throughput*, latência p95/p99 e resiliência a *rate limits*.

d) Domínios expandidos — testar o agente em novos contextos, como e-commerce (extração de preços), documentos governamentais e redes sociais.

e) Detecção de *shadow DOM* — incorporar um módulo específico para inspeção de estruturas encapsuladas (e.g., *Web Components*) e carrosséis dinâmicos.

f) Avaliação com outros modelos — testar o desempenho com diferentes LLMs.

Em síntese, este trabalho reforça o potencial da convergência entre modelos de linguagem e engenharia de dados como um caminho promissor para automatizar tarefas complexas de extração e integração de informações em larga escala. A abordagem proposta tem aplicabilidade imediata em contextos públicos e privados que demandam dados atualizados, confiáveis e com ampla cobertura — com significativa redução no custo humano de manutenção.

Agradecimentos

Este trabalho foi apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (Processos 2016/17078-0, 23/18026-8 e 24/13328-9) , Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Referências

Ahluwalia, A. and Wani, S. (2024). Leveraging large language models for web scraping. *arXiv preprint arXiv:2406.08246*. Acesso em: 15 jul. 2025.

Aslanyürek, C. and Yerlikaya, T. (2024). Automatic regular expression generation for extracting relevant image data from web pages using genetic algorithms. *IEEE Access*, 12:90660–90669.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. `https://arxiv.org/abs/2005.14165`. arXiv preprint arXiv:2005.14165. Acesso em: 15 jul. 2025.

Fundação Sistema Estadual de Análise de Dados (SEADE) (2025). Pesquisa de investimentos do estado de são paulo (piesp). `https://investimentos.seade.gov.br`. Acesso em: 28 abr. 2025.

Huang, W., Gu, Z., Peng, C., Li, Z., Liang, J., Xiao, Y., Wen, L., and Chen, Z. (2024). Autoscraper: A progressive understanding web agent for web scraper generation. *arXiv preprint arXiv:2404.12753*. Acesso em: 15 jul. 2025.

Khder, M. A. (2021). Web scraping or web crawling: State of the art, techniques, approaches and application. *International Journal of Advances in Soft Computing and Its Applications*, 13(3):145–168.

Reynolds, L. and McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. https://arxiv.org/abs/2102.07350. arXiv preprint arXiv:2102.07350. Acesso em: 15 jul. 2025.

Roig, A. P. (2023). *Web data scraper*. PhD thesis, Universitat Politècnica de València, València, ES. Tese de Doutorado. Acesso em: 15 jul. 2025.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. https://arxiv.org/abs/2303.11366. arXiv preprint arXiv:2303.11366. Acesso em: 15 jul. 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models. https://arxiv.org/abs/2201.11903. arXiv preprint arXiv:2201.11903. Acesso em: 15 jul. 2025.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023a). Tree of thoughts: Deliberate problem solving with large language models. https://arxiv.org/abs/2305.10601. arXiv preprint arXiv:2305.10601. Acesso em: 15 jul. 2025.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023b). React: Synergizing reasoning and acting in language models. https://arxiv.org/abs/2210.03629. arXiv preprint arXiv:2210.03629. Acesso em: 15 jul. 2025.

You, J., Lee, K., and Kwon, H. (2024). Deepscraper: A complete and efficient tweet scraping method using authenticated multiprocessing. *Data and Knowledge Engineering*, 149:102260.