Avaliação do Desempenho de LLMs Médios na Extração de Esquemas JSON: Um Estudo Empírico

Thiago C. Almeida [1], **Denio Duarte**[1], **Geomar A. Schreiner**[1], **Samuel Feitosa**[1]

[1]Universidade Federal da Fronteira Sul (UFFS) – Campus Chapecó
Caixa Postal 181 – 89815-899 – Chapecó – SC – Brazil

{duarte, gschreiner,samuel.feitosa}@uffs.edu.br, thiagochafado123@gmail.com

Abstract. *Large Language Models (LLMs) are widely used across various natural language processing tasks; however, the substantial computational overhead remains a primary constraint on their deployment. This study investigates the performance of mid-sized LLMs—specifically Gemma 3-4B and Qwen 2.5-14B—concerning parameter count in the context of schema extraction from JSON collections. Performance is evaluated based on the accuracy of the extracted schema (i.e., the validity of documents relative to the schema) and total processing latency. Experimental results demonstrate that these models successfully extracted JSON schemas with a success rate exceeding 99%. Nevertheless, even with mid-sized models, the high computational cost and processing time continue to pose significant challenges to the solution's scalability in resource-constrained environments.*

Resumo. *Modelos de Linguagem de Grande Escala (LLMs) são amplamente utilizados em várias tarefas de processamento de linguagem natural, porém com alto custo computacional que limita o seu uso. Este estudo investiga o desempenho de LLMs de tamanho médio em relação ao número de parâmetros, especificamente Gemma 3-4B e Qwen 2.5-14B, na extração de esquemas de coleções JSON. O desempenho é avaliado pela corretude do esquema extraído (i.e., os documentos utilizados são válidos em relação ao esquema extraído) e o tempo total do processamento. Os resultados dos experimentos mostram que os modelos tiveram sucesso (taxa superior a 99%) em extrair esquemas JSON. Contudo, mesmo com modelos médios, o elevado custo computacional e o tempo de processamento ainda representam desafios significativos para a escalabilidade da solução em ambientes com recursos limitados.*

1. Introdução

Atualmente, o JSON (JavaScript Object Notation) tornou-se um padrão para intercâmbio de dados no ambiente web, principalmente devido à sua simplicidade [Bourhis et al. 2017]. Além disso, muitos bancos de dados NoSQL utilizam JSON como um formato interno para armazenar seus dados (*e.g.*, MongoDB e HBase). Geralmente, os dados JSON não estão associados a um esquema rígido [Spoth et al. 2021, Maiwald et al. 2019]. Essa ausência de esquema rígido é vantajosa para o desenvolvimento rápido e a troca de dados; no entanto, um esquema desempenha um papel crucial na validação da estrutura e do conteúdo dos dados JSON [Baazizi et al. 2019]. Um esquema pode ajudar a mitigar erros de dados e é benéfico para otimizações de consulta e armazenamento [Bouchou and Duarte 2007].

Apesar dos avanços recentes, a tarefa de extração de esquemas em documentos JSON permanece desafiadora. A estrutura hierárquica do formato, aliada às possibilidades de aninhamento profundo, variações tipológicas, campos opcionais e *arrays* heterogêneos, aumenta a complexidade do problema [Pezoa et al. 2016]. Atualmente, as pesquisas na área concentram-se principalmente em algoritmos baseados em estruturas formais, como árvores e grafos, ou em abordagens estatísticas que inferem padrões a partir de múltiplos documentos. Contudo, tais métodos, embora robustos, frequentemente não capturam nuances semânticas — como relações implícitas entre atributos — e podem sofrer com inconsistências presentes em grandes coleções de dados reais.

Paralelamente à evolução dessas abordagens tradicionais, o campo da Inteligência Artificial testemunhou avanços com o surgimento dos *Modelos de Linguagem de Grande Escala* (*Large Language Models* – LLMs). Treinados com quantidades massivas de dados textuais e baseados na arquitetura *Transformer* [Vaswani et al. 2017], os LLMs demonstraram capacidades superiores às abordagens tradicionais de processamento natural de linguagens em tarefas relacionadas ao raciocínio, síntese e análise estrutural de informações textuais [Wei et al. 2022]. Essas capacidades os tornam candidatos para atuar além de sua função original de geração de linguagem, podendo ser explorados em domínios técnicos como extração de conhecimento, modelagem semântica e interpretação de dados semi-estruturados.

Este artigo então tem o objetivo de comparar e testar dois modelos LLM que executam em hardware tradicionais (i.e., capacidade de processador, memória e GPU equivalentes a um notebook) na tarefa de extrair/descobrir esquemas a partir de coleções de documentos JSON. Para este fim, foram escolhidos dois modelos: *Gemma3-4B* (chamado de Gemma a partir de agora) e *Qwen 2.5-14B* (chamado de Qwen a partir de agora). Ambos os modelos considerados médios em relação ao número de parâmetros, sendo que o primeiro possui 4 bilhões e o segundo 14 bilhões. Os modelos possuem janelas de contextos (número de *tokens* de entrada) de 128 mil. A proposta consiste em explorar a capacidade desses modelos para inferir estruturas e tipos de coleções heterogêneas de documentos, avaliando seu desempenho em cenários que variam em complexidade estrutural e tamanho das coleções. Para isso, foram definidas três coleções de documentos reais, além da criação de uma coleção sintética. Os documentos são classificados como de baixa e alta complexidade. Essa classificação considerou o número de campos, profundidade de aninhamento e tamanho (em bytes) da coleção. Após a inferência dos esquemas, os mesmos foram avaliados utilizando o validador de esquemas *Javascript Ajv*, utilizando os documentos fontes dos esquemas extraídos. Os resultados obtidos se mostraram satisfatórios, embora o tempo de extração (custo) tenha sido considerado elevado no cenário de hardware considerado.

Sumarizando, as contribuições deste trabalho são: um pipeline para extração de esquema JSON usando LLM médios e hardware limitado (contribuição $\mathbf{C}_1$), uma proposta de fragmentação automática de coleções JSON ($\mathbf{C}_2$), uma abordagem de reconstrução dos esquemas gerados a partir da fragmentação ($\mathbf{C}_3$), e uma discussão do desempenho dos modelos de LLM considerados ($\mathbf{C}_4$).

```
{
 "title": "The Lord of the Rings",
 "author": "J.R.R. Tolkien",
 "published_in": 1954,
 "books": [
   {
     "title": "The Fellowship of the Ring",
     "main_characters": ["Frodo","Sam","Gandalf","Aragorn"]
   },
   {
     "title": "The Two Towers",
     "main_characters": ["Frodo","Sam","Gollum","Legolas","Gimli"]
   },
   {
     "title": "The Return of the King",
     "main_characters": ["Frodo","Sam","Aragorn","Gandalf"]
   }
 ],
 "genre": "Fantasy",
 "adapted_to_film": true
}
```

(a)

```
{
 "type": "object",
 "properties": {
   "title": { "type": "string" },
   "author": { "type": "string" },
   "published_in": { "type": "integer" },
   "books": {
     "type": "array",
     "items": {
       "type": "object",
       "properties": {
         "title": { "type": "string" },
         "main_characters": {
           "type": "array",
           "items": { "type": "string" }
         }
       },
       "required": ["title", "main_characters"]
     }
   },
   "genre": { "type": "string" },
   "adapted_to_film": { "type": "boolean" }
 },
 "required": ["title","author","published_in","books",
              "genre","adapted_to_film"]
}
```

(b)

Figura 1. Exemplo de documento JSON (a) e seu esquema (b).

2. Referencial Teórico

A extração de esquemas, no contexto de bancos de dados, refere-se ao processo de identificação e recuperação da estrutura subjacente aos dados armazenados em uma determinada fonte [Wang and Liu 1997]. Essa fonte pode assumir diferentes formas, como arquivos de texto, coleções de documentos JSON ou XML, ou ainda sistemas gerenciadores de bancos de dados relacionais ou NoSQL. O objetivo principal desse processo é inferir, de forma automática ou semi-automática, as entidades, atributos, tipos de dados e suas respectivas relações, permitindo que o conteúdo seja compreendido, validado e manipulado com maior precisão e eficiência [Frozza et al. 2018].

Este trabalho considera coleções JSON para verificar o desempenho em extração de esquemas de LLM médios. A Figura 1 apresenta um extrato de um documento JSON (a) e um possível esquema (b) que descreve o extrato. Percebe-se, na figura, que um documento JSON é composto por conjunto de duplas *chave:conteúdo* em que a *chave* representa o atributo e o *conteúdo*, o valor do atributo. O conteúdo pode ser atômico como, por exemplo, um inteiro (1954) para a chave *published_in* (Linha 4) ou um array como para *books* (Linha 5) composta por objetos JSON (Linhas 6-17). Note que a Figura 1(b) apresenta um esquema representando o extrato da Figura 1(a). Por exemplo, a chave *published_in* é obrigatória (Linha 24) e o seu conteúdo deve ser representado como um inteiro (Linha 6).

Essa estrutura hierárquica e flexível torna o JSON um dos formatos mais utilizados para troca de dados na Web [Bourhis et al. 2017] e especialmente útil para representar informações complexas em diversos contextos, como APIs e bancos de dados NoSQL. O MongoDB, por exemplo, é um sistema gerenciador de banco de dados da classe NoSQL orientado a documentos que utiliza JSON como representação dos dados [MongoDB Inc. 2024].

Apesar de documentos JSON não exigirem um esquema para serem publicados, a extração ou definição de esquemas é fundamental para garantir padronização e interpretabilidade dos dados. Esquemas facilitam tarefas como validação, reutilização, integração e indexação por diferentes sistemas e usuários [Kellou-Menouer et al. 2022].

Por outro lado, LLMs têm sido utilizados em várias tarefas computacionais (*e.g.*, geração e transformação de conteúdo e análise de dados). Esses modelos são baseados em técnicas avançadas de aprendizado profundo, desenvolvidos para interpretar e gerar textos, de maneira humana ou estruturada. Estes modelos surgiram como evolução dos primeiros modelos estatísticos de linguagem, como os modelos *n-grams*, e posteriormente dos modelos baseados em redes neurais recorrentes (RNNs) e Long Short-Term Memory (LSTM). Com o avanço da arquitetura *Transformer* introduzida por [Vaswani et al. 2017], tornou-se possível escalar o treinamento de modelos de linguagem para níveis antes impraticáveis, tanto em termos de quantidade de dados quanto de complexidade de tarefas.

Com o crescimento no tamanho dos modelos, observou-se o surgimento de capacidades emergentes, como raciocínio multi-etapas, compreensão de instruções, geração de código, e até mesmo interações multimodais. Essas capacidades, não explicitamente programadas, desafiam a intuição tradicional sobre o funcionamento de sistemas computacionais e trazem novas perspectivas sobre a interface entre linguagem e inteligência artificial [Wei et al. 2022]. Apesar de seu bom desempenho em uma ampla gama de tarefas, os LLMs ainda enfrentam limitações relevantes, como alucinação de fatos, viés nos dados de treinamento e elevado custo computacional [Bender et al. 2021].

A extração de esquemas de documentos JSON é uma temática já abordada pela literatura, a Seção 5 apresenta algumas das abordagens já propostas. A maioria delas se apoia em estrutura de dados específicas para varrer e encontrar a estrutura que define os documentos, *e.g.*, grafos ou árvores; poucos trabalhos de extração de esquemas utilizam LLMs.

3. Metodologia

Esta seção aborda a metodologia do estudo, apresentando a estratégia utilizada para processar coleções volumosas no contexto de LLMs médios.

Coleções de documentos JSON costumam ultrapassar milhões de caracteres. Os tamanhos das janelas de contextos dos LLMs definem o tamanho da entrada para o modelo. O número de tokens pode variar de 128 mil a 100 milhões. Os modelos empregados neste trabalho possuem uma janela de contexto de 128 mil, assim as coleções de documentos devem ser fragmentadas para executar os experimentos. Este pipeline (contribuição $\mathbf{C}_1$) é definido a seguir.

Dada uma coleção de documentos C, o pipeline de processamento é definido pelas seguintes etapas de forma automática[1] (contribuições $\mathbf{C}_2$ e $\mathbf{C}_3$):

E1 - Fragmentação A coleção C é composta em um conjunto $D = \{d_1, d_2, \ldots, d_k\}$ de k documentos JSON individuais.

E2 - Amostragem Um subconjunto $S \subseteq D$ é selecionado aleatoriamente de D, resultando em uma amostra de n documentos (onde $n \leq k$).

E3 - Análise de Complexidade Cada documento $d_i \in S$ é classificado como sendo de *alta* ou *baixa* complexidade. Esta análise é detalhada a seguir.

E4 - Extração por LLM Para cada documento $d_i \in S$, um modelo de LLM correspondente (baseado na complexidade) é utilizado para extrair um esquema individual e_i. Este processo resulta em um conjunto $E_S = \{e_1, e_2, \ldots, e_n\}$ de n esquemas.

[1] Fontes disponíveis em `github.com/ThiagoChafado/SchemaDiscovery-LLM`

E5 - Fusão de Esquemas O conjunto de n esquemas E_S é submetido ao processo de fusão (detalhado a seguir), resultando em um único esquema mestre E_M consolidado.

Validação: O esquema mestre final E_M é então utilizado para validar a conformidade de todos os documentos do conjunto D utilizando o *Ajv*, um validador de esquemas JSON.

Para os experimentos, foram selecionadas três coleções de documentos JSON de dados reais e uma coleção sintética. A Tabela 1 apresenta algumas estatísticas de tamanhos das coleções que são explicadas a seguir.

1. **Russian election 2018 — twitter user activity:**[2] Coleção que contém dados sobre posts do twitter relacionados às eleições russas de 2018 (chamada de *Russian*).
2. **Prescription-based prediction:**[3] Coleção contendo dados sobre quantidade de vezes que um medicamento foi prescrito por um determinado médico, além de outras informações sobre o mesmo (chamada de *Prescriptions*).
3. **Airbnb listings London-UK**[4]**:** Coleção contendo dados sobre locações do Airbnb de Londres (chamada de AirBnB).
4. **Dados sintéticos:** Coleção criada de forma sintética utilizado para avaliar escalabilidade da proposta (chamada de *Synthetic*).

Tabela 1. Estatísticas descritivas dos datasets originais e das amostras processadas.

Feature	AirBnB	Prescriptions	Russian	Synthetic
Size (Mb)	42,45	156,39	10.740	100,01
# Docs	83.711	239.930	1.945.365	109600
# Sample	1.000	1.000	1.000	500
Min Keys	19	11	74	36
Max Keys	19	416	771	74
Avg Keys	19,0	30,1	186,9	53,2
Min Depth	3	3	4	5
Max Depth	3	3	11	5
Avg Depth	3,0	3,0	7,5	5,0

As coleções foram pré-processadas antes de serem enviadas para a extração dos respectivos esquemas. As etapas *E1*, *E2* e *E3* do pipeline.

1. **Amostragem automática e fragmentação:** A coleção de entrada passa por um processo de amostragem automática para criar uma sub-coleção representativa e de tamanho gerenciável. Este processo é controlado por um conjunto de parâmetros que definem as condições de parada. A amostragem termina assim que o primeiro dos seguintes limites é atingido:
    - **Limite de Documentos:** Um número máximo absoluto de documentos a serem extraídos.
    - **Limite de Tamanho Proporcional:** Um tamanho máximo total para a sub-coleção, definido como uma porcentagem do tamanho do arquivo original.

[2] `www.kaggle.com/datasets/borisch/russian-election-2018-twitter`

[3] `www.kaggle.com/datasets/roamresearch/prescriptionbasedprediction`

[4] `insideairbnb.com/london/`

Após a conclusão da amostragem, a sub-coleção resultante é fragmentada em múltiplos arquivos, de modo que cada um contém um único documento JSON. Essa etapa não apenas garante representatividade e viabilidade computacional, mas também busca contornar o limite de janela de contexto imposto pelos LLMs. No caso deste trabalho 128 mil.

2. **Análise de complexidade:** Cada documento JSON individual é submetido a uma análise de complexidade, sendo então classificado em baixa ou alta. Essa análise é útil para a extração, sendo assim possível utilizar um modelo de menor escala a fim de otimizar o tempo de extração. Esta complexidade é medida a partir dos seguintes critérios e *Thresholds*:
    - **Número de Chaves:** A contagem total de chaves (propriedades) únicas no documento. Usada para calcular o número mínimo, máximo e médio de chaves das amostras (*Min Keys*, *Max Keys* e *Avg Keys* na Tabela 1).
    - **Profundidade de Aninhamento:** O nível máximo de aninhamento de objetos ou arrays dentro da estrutura do documento. Usada para calcular o menor, maior e média das profundidades das amostras (*Min Depth*, *Max Depth* e *Avg Depth* na Tabela 1).
    - **Tamanho do Arquivo:** O tamanho total do documento em bytes.

    Com base nesses critérios e por meio de testes empíricos, um documento foi classificado como de **alta complexidade** se atender a qualquer uma das seguintes condições:
    - O número total de chaves for superior a 100.
    - A profundidade máxima de aninhamento for superior a 3.
    - O tamanho do documento amostra for superior a 30 kilobytes.

    Documentos que não se enquadram em nenhuma dessas condições são, por padrão, classificados como de **baixa complexidade** e podem ser direcionados a um modelo de LLM mais leve para a extração do esquema.

Observando a Tabela 1, percebe-se que as amostras das coleções *Russian* e *Synthetic* foram classificadas com alta complexidade, devido ao grau médio de profundidade ser 7,5 e 5,0 nas coleções bem como a média de chaves (186,4 e 53,2). Ambos os casos atendem ao perfil complexo da coleção. As outras coleções foram classificadas como baixa complexidade, com exceção de alguns poucos documentos de *Prescriptions*, que devido ao seu número de chaves ser maior que 100, foram classificados como alta complexidade. O experimento que tentou utilizar o modelo de menor custo para esses documentos falhou em inferir uma resposta correta, ou sequer estruturalmente ou semanticamente válida. Mais detalhes podem ser encontrados em **Limitações** da Seção 4.

Para a extração dos esquemas, foram definidos dois LLMs:

1. **Gemma3-4B**[5]**:** Gemma é uma família de modelos leves e de última geração do Google, criada a partir da mesma pesquisa e tecnologia utilizadas para desenvolver os modelos `Gemini`. Neste projeto, ele foi utilizado para os documentos classificados como baixa complexidade. Nos testes empíricos, este modelo não teve sucesso em extrair esquemas de documentos de alta complexidade: resultados com problemas de alucinações e travamento de hardware.

[5] `https://huggingface.co/google/gemma-3-4b-it`

Tabela 2. Tempo Médio e Total de Processamento de Extração.

Coleção	# Doc Analisados	Tempo Médio (s)	Total (h)	Pico de Mem.(GB)
AirBnB[1]	1000	69,52	19,31	3,0
Prescriptions[1,2]	1000	51,56	14,32	3,0
Synthetic[2]	500	113	15,68	9,1
Russian[2]	1000	604,32	167,87	13,0

[1] Gemma [2] Qwen

2. **Qwen 2.5-14B[6]:** Qwen é uma família de modelos desenvolvida pela Alibaba Cloud e voltada tanto para aplicações de linguagem natural quanto para raciocínio estrutural em dados. Neste projeto, ele foi utilizado para os documentos classificados como alta complexidade. Seu desempenho melhor se deve à maior quantidade de parâmetros, o que resultou em extrações válidas para os documentos, porém com um custo computacional consideravelmente maior.

Os experimentos foram conduzidos em um ambiente computacional utilizando hardware tradicional: um notebook com um processador Apple M4 com 16 GB RAM compartilhada. A aceleração de hardware foi provida pela GPU integrada do chip, por meio do framework de aprendizado de máquina MLX com suporte à API Metal da Apple. Testes empíricos também mostraram que 1.000 documentos para as coleções reais e 500 para a coleção sintética são adequados para os experimentos. Valores maiores apresentaram um tempo de processamento acima de dez horas para as coleções reais. Valores menores prejudicaram os esquemas resultantes. Outro ponto importante é que não foi usada nenhuma técnica adicional para os modelos como, por exemplo, *fine-tuning* ou Geração Automática por Recuperação (RAG em inglês). A decisão foi feita para obter-se um *pipeline* pronto para usar sem necessidade de configurações para os modelos. A etapa E4 do pipeline é implementada utilizando o seguinte *prompt*:

```
You are a data schema extraction expert. Generate only the JSON Schema
(in standard JSON Schema Draft 2020-12 format) for the
following JSON document.
- Include 'required' when it can be clearly inferred.
- Do not include 'description' for any field.
- Output only the schema, no explanations or extra text.
End your response after the final '}'.
Input JSON:
```

A Tabela 2 apresenta alguns dados sobre a extração destes esquemas e quais modelos foram utilizados para a tarefa. A análise dos resultados apresentados revela uma observação principal (contribuição $\mathbf{C}_4$):

- **Impacto do Tamanho do Modelo:** O dataset *Russian*, classificado como de alta complexidade pela análise de pré-processamento, exigiu o uso do Qwen. Como esperado, este modelo maior demandou um tempo de processamento médio por documento (604s) e um pico de uso de memória (13.0 GB) consideravelmente superiores aos demais. Esta alta demanda de VRAM, operando no limite da memória unificada de 16 GB do hardware, levou a falhas críticas (*crashes*) durante a execução, exigindo a reinicialização da máquina e evidenciando um gargalo computacional.

[6] `https://huggingface.co/Qwen/Qwen2.5-14B`

- **Influência da Densidade da Informação:** Mesmo utilizando o mesmo modelo (Gemma), notou-se que o tempo de processamento para a coleção *Prescriptions* foi consideravelmente maior que *AirBnB*. Essa diferença é atribuída à densidade informacional dos documentos reais. A presença de um número elevado de chaves, combinada com valores textuais longos (como descrições e comentários), aumenta a complexidade de leitura para o LLM, resultando em um tempo de processamento maior por documento.

Para cada coleção, foram gerados esquemas para cada uma das amostras. Assim, conforme definido na *E5*, é necessário fusionar os esquemas de cada amostra, gerando o esquema mestre para a coleção. A metodologia de fusão é dividida em duas fases principais: a verificação dos esquemas e a geração do esquema mestre. Na primeira fase é verificado se os aninhamentos estão corretos, se alguma sintaxe do esquema não foi obedecida e coletada algumas estatísticas para a geração do esquema final (contribuição $\mathbf{C}_3$).

- **Carregamento e Reparo de Formatação:** O sistema tenta carregar o arquivo JSON. Em caso de falha de decodificação — um problema comum em saídas de LLMs — um mecanismo de reparo é ativado. Este mecanismo extrai o conteúdo entre o primeiro caractere de abertura (‘{‘) e o último de fechamento (‘}‘), tratando a maioria dos erros de formatação.
- **Reparo Estrutural:** Uma vez que o JSON é carregado com sucesso, uma verificação de sanidade estrutural é realizada. Esta etapa corrige anomalias comuns na geração de schemas por LLMs, como a utilização de valores booleanos para a chave `"required"` em vez de uma lista, ou a omissão da chave `"properties"` em nós do tipo `"object"`.
- **Coleta de Estatísticas:** Com o esquema limpo e estruturalmente válido, suas características são coletadas e armazenadas em uma estrutura de dados hierárquica que modela uma *árvore de estatísticas* (implementada como dicionários aninhados). São armazenadas: a contagem total de aparições, a frequência com que foi marcada como obrigatória, e a contagem de cada tipo de dado observado (e.g., `string`, `integer` e `null`). Para cada nó, o nome do campo é armazenado com o número de ocorrências, a frequência com que a propriedade foi marcada como obrigatória e a distribuição dos tipos observados. As estatísticas extraídas nessa etapa são utilizadas posteriormente para definir a obrigatoriedade dos campos e os tipos aceitos durante a construção do esquema mestre.

Após a análise de todos os arquivos válidos, uma árvore de estatísticas consolidada é utilizada para construir o esquema mestre final. As decisões são tomadas com base em limiares (*thresholds*) pré-definidos:

- **Definição de Campos Obrigatórios:** Uma propriedade é incluída na lista `"required"` do esquema mestre se a sua taxa de obrigatoriedade for maior ou igual ao `REQUIRED_THRESHOLD`. Para os experimentos finais, este limiar foi definido como **1,0**, ou seja, um campo será *required* no esquema mestre se aparecer em todos os documentos da amostra.
- **Definição do Tipo de Dado:** Para cada propriedade, o tipo de dado mais frequente (excluindo `null`) é identificado. Se a sua frequência relativa entre as ocorrências não-nulas for maior ou igual ao `TYPE_THRESHOLD` (definido como **0,75**), este

é eleito como o tipo definitivo. Caso contrário, é proposta uma lista de tipos possíveis para o campo.

O resultado desse processo são os esquemas mestres extraídos e prontos para serem avaliados[7].

A fim de avaliar a escalabilidade do estudo e mitigar a dependência exclusiva de coleções reais, foi construída uma coleção sintética de documentos JSON. A geração dos dados foi realizada com a biblioteca *Faker*[8], permitindo definir previamente a estrutura dos documentos e simular um cenário heterogêneo semelhante a aplicações reais. A coleção sintética é composta por três entidades principais: *Pessoa*, *Produto* e *Transação*. Cada entidade contém um conjunto fixo de propriedades obrigatórias, além de propriedades opcionais. A entidade *Transação*, por sua vez, incorpora outras estruturas, incluindo objetos e *arrays*, possibilitando avaliar a capacidade dos LLMs em inferir esquemas em contextos aninhados. O esquema desta coleção tem a seguinte estrutura (os atributos em itálico são opcionais):

customer(id:int, name:str, *email*:str, age:int, *phone*:str, city:str, state:str, registered_at:date)
product(id:int, name:str, category:str, price:float, in_stock:bool, rating:(1-5), created_at:date)
transaction(id:int, person_id:int, date:date, status:str, payment_method:str, transaction_items[])
  transaction_items(id:int, product_id:int, quantity:int, price:float)

Esse esquema resulta em uma coleção com diversidade estrutural: documentos de tamanhos distintos, presença de campos opcionais, múltiplos domínios categóricos, objetos aninhados e `arrays`. Tais características tornam o conjunto sintético apropriado para testar a robustez dos LLMs na inferência de esquemas e permitem observar o impacto da complexidade estrutural no desempenho dos modelos.

Para avaliar a qualidade dos esquemas mestres gerados, utilizou-se o validador de esquemas *JavaScript Ajv*[9]. Esse validador recebe o esquema mestre como entrada e o aplica a cada documento da coleção, verificando se o documento está em conformidade com as regras estruturais e semânticas definidas no esquema. Assim, é possível mensurar de forma objetiva o grau de generalização e precisão do esquema inferido. Além disso, o estudo de [Lima et al. 2024], que emprega LLMs para testar validadores de esquemas JSON em JavaScript e outras linguagens, identificou que os validadores em JavaScript apresentam poucas inconsistências, reforçando a confiabilidade da ferramenta escolhida para este trabalho.

4. Resultados e Discussões

As Tabelas 2 e 3 apresentam os resultados do tempo de execução e da validação dos esquemas mestres gerados pelo processo de fusão orientado por LLMs. Esta etapa avalia a capacidade da abordagem de extrair os esquemas em tempo computacional aceitável e a capacidade dos modelos de linguagem em generalizar corretamente os padrões estruturais presentes em coleções de dados reais de diferentes domínios (contribuição $\mathbf{C}_4$).

Os resultados mostram que ambos os modelos exibiram desempenho robusto na tarefa de inferência e validação de esquemas, mas os tempos de extração foram menos satisfatórios. Gemma atingiu próximo de 100% de sucesso, sugerindo que, em cenários

[7] Todos os limiares utilizados foram calculados por meio de experimentos empíricos iniciais.
[8] `https://faker.readthedocs.io/en/master/` [9] `ajv.js.org/`

Tabela 3. Resultados da Validação do Schema Mestre (LLM) contra as Coleções de Dados Originais Completas.

Dataset	# de Docs	Aprovados	Reprovados	Sucesso (%)
AirBnB	83.711	83.686	25	99.97
Prescriptions	239.930	238.507	1.423	99.41
Russian	1.945.365	1.945.365	0	100.00
Synthetic_100	109.600	109.600	0	100.00
Synthetic_300	109.600	109.600	0	100.00
Synthetic_500	109.600	109.600	0	100.00

menos complexos, modelos de menor porte são suficientes para capturar a variação estrutural dos dados. O tempo de processamento para o *AirBnB* foi superior a 19 horas, com pico de memória de três gigabytes. Por outro lado, o Qwen apresentou 100% de acurácia, validando corretamente todos os documentos dos datasets associados à alta complexidade. Essa diferença evidencia que modelos maiores apresentam maior capacidade de abstração e generalização. Para os experimentos utilizando dados sintéticos com 100, 300 e 500 amostras, a validação teve 100% de sucesso. Porém, o custo computacional foi mais alto em relação ao Gemma: a coleção *Russian* que possui muitos textos atingiu quase 168 horas de processamento, com pico de memória de 13 gigabytes. Isso indica que coleções maiores (e mais complexas) devem ser processadas com modelos mais robustos e hardware mais otimizado. A qualidade do esquema gerado evidencia que a abordagem de fragmentação automática e a fusão ($\mathbf{C}_2$) foi satisfatória dentro do *pipeline* proposto. Apesar de todos os documentos terem sido utilizados na validação, acredita-se que os documentos de extração (representam menos de 1%) não influenciaram na métrica de desempenho dos modelos.

A maioria dos documentos reprovados nas coleções *AirBnB* e *Prescriptions* foram por motivo de tipagem. Em alguns documentos, por exemplo, a chave `name` possui valor do tipo `null`, mas a maioria são *string*. Entretanto, como tais instâncias representam menos de 0,5% da coleção completa, espera-se que a amostra contenha uma proporção ainda menor desses valores nulos (inferior ao limiar de 25% definido para a fusão). Dessa forma, o esquema mestre inferiu que o tipo predominante para a chave `name` é `string`. Contudo, durante a validação, documentos contendo `name = null` são reprovados, já que não atendem ao tipo definido no esquema mestre. Nota-se, neste caso, que o limiar pode ser alterado para valores mais próximos da necessidade do usuário.

Observando a Tabela 3, nota-se também que todos os datasets obtiveram taxas de validação superiores a 99%, incluindo o dataset *Russian*, que apresenta o maior volume absoluto e o maior grau de heterogeneidade textual. Os experimentos com os dados sintéticos tiveram como objetivo também de servir como uma prova de conceito para tipos e opcionalidade dos esquemas, além da escalabilidade do modelo Gemma neste cenário.

A Figura 2 apresenta um extrato do esquema (entidade *transaction*) gerado a partir da coleção *Synthetic*. Perceba que os campos marcados como *required* são aqueles obrigatórios propostos no esquema original (Linhas 24-29 para *transaction* e Linhas 17-20 para *transaction_items*). Também é possível verificar que *transaction_items* foi transformado corretamente em um array (Linhas 9-16). Os esquemas gerados foram iguais para todos os cenários.

Figura 2. Esquema para a entidade *transaction*

```
"transaction": {
  "type": "object",
  "properties": {
    "id": {"type": "integer"},
    "person_id": {"type": "integer"},
    "date": {"type": "string"},
    "status": {"type": "string"},
    "payment_method": {"type": "string"},
    "transaction_items": {
      "type": "array",
      "items": {"type": "object",
                "properties": {"id": {"type": "integer"},
                "product_id": {"type": "integer"},
                "quantity": {"type": "integer"},
                "price": {"type": "number"}
               },
               "required": ["id",
                            "price",
                            "product_id",
                            "quantity"]
      }
    }
  },
  "required": ["date",
               "id",
               "payment_method",
               "person_id",
               "status",
               "transaction_items"] }
```

Os experimentos utilizando a coleção *Synthetic* com 100, 300 e 500 amostras tiveram como objetivo também mostrar a escalabilidade do pipeline. Os tempos foram: 100 amostras 3,21h, 300 amostras 9,13h e 500 amostras 15,68h. Perceba que a proporção entre as amostras e o tempo de execução são constantes. Por exemplo, a proporção entre 100 e 300 é 3, e a proporção entre 3,21 e 9,13 é 2,84. Isso indica que o crescimento dos tempos de execução é linear. Importante salientar que, devido ao custo de execução (tempo), todos os cenários foram executados apenas uma vez.

Limitações: (contribuição $\mathbf{C}_4$) Uma das limitações comuns para o uso de LLMs para extração de esquemas é seu custo computacional e o limite da janela de contexto. Ambas as limitações podem ser contornadas fragmentando a coleção em documentos menores e selecionando uma amostra representativa (etapas *E1* e *E2* do pipeline). Porém, essa amostra não pode ser pequena demais (que ocasionaria em um esquema mestre incapaz de generalizar adequadamente as variações estruturais presentes no dataset) nem grande demais, pois isso geraria um custo de tempo e processamento elevado. Pode-se citar, então, três situações que comprometem a extração de esquemas JSON utilizando LLM médios: (*i*) a necessidade das amostras representarem estruturalmente um possível esquema para a coleção, (*ii*) modelos menores, como os utilizados para documentos considerados de baixa complexidade, podem não conseguir extrair um esquema satisfatório, ou modelos maiores podem ter seus tempos de execução inaceitáveis, e (*iii*) extrair semânticas dos dados como, por exemplo, enumerações e identificadores.

5. Trabalhos Relacionados

A tarefa da extração ou descoberta de esquemas é explorada em várias pesquisas [Yun et al. 2024, Banhara et al. 2024, Klessinger et al. 2023, Spoth et al. 2021, Namba 2021, Abdelhedi et al. 2021, Baazizi et al. 2019, Frozza et al. 2018]. Porém, a maioria delas se apoia em estruturas tradicionais (e.g., grafos e dicionários) para extrair os esquemas. Já os trabalhos de [Dagdelen et al. 2024] e [Mior 2024] aplicam LLMs de

forma similar à proposta aqui apresentada. Sendo que o primeiro extrai a estrutura de textos científicos e o segundo enriquece semanticamente esquemas JSON extraídos de forma tradicional.

Esquemas JSON podem ser complexos contendo enumerações, *tagged unions*, objetos complexos e cardinalidades. Para esses casos, apenas os trabalhos que aplicam técnicas tradicionais implementam esses conceitos. Por exemplo, apenas os trabalhos de [Banhara et al. 2024] e [Klessinger et al. 2023] extraem *tagged unions*; e enumerações são tratadas apenas por [Banhara et al. 2024]. A maioria extrai apenas os tipos básicos e objetos complexos. O mesmo que mostrou os experimentos realizados neste trabalho.

O trabalho de [Dagdelen et al. 2024] propõe uma abordagem para a extração de informações estruturadas a partir de textos científicos, utilizando técnicas de ajuste fino (*fine-tuning*) em grandes modelos de linguagem (LLMs). Já [Mior 2024], utiliza LLMs para acrescentar informações em esquemas extraídos. Foi realizado um *fine-tuning* com pares de documentos, onde o primeiro era o esquema extraído e o segundo o mesmo esquema com semânticas e anotações adicionais.

Também, pode-se citar a pesquisa de [Silva et al. 2025] que estuda os desempenhos de vários tamanhos do modelo Qwen2.5 para a tarefa de *Text-to-SQL*. Os autores, como a proposta aqui apresentada, avaliam tamanhos de LLMs para identificar a mais viável na tarefa selecionada. Porém, a tarefa de *Text-to-SQL* é menos complexa em relação ao tamanho da entrada, não sendo necessário fragmentá-la.

6. Conclusão

Este artigo verificou a viabilidade de LLM médios em extrair esquemas JSON. O foco foi em modelos médios (número de parâmetros) executando em *hardware* não especializado. Os resultados indicam que ainda tais modelos possuem um desempenho inferior à abordagens tradicionais, tanto nas facetas não triviais de um esquema JSON quanto no tempo de execução. A avaliação aqui realizada mostrou também que coleções com mais dados textuais tendem a demorar mais tempo para serem processadas, resultado esperado considerando as características do funcionamento dos LLMs.

De forma geral, os resultados indicam que LLMs representam uma direção promissora para esquematização automática de dados semiestruturados, especialmente em domínios onde a variabilidade estrutural e semântica dos documentos é elevada. Contudo, sua adoção plena depende de avanços futuros que reduzam os requisitos computacionais e ampliem a capacidade de processamento contextual, tornando essa abordagem mais acessível e sustentável em cenários práticos. Também foi possível verificar que as contribuições propostas neste trabalho (*i.e.*, $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$ e $\mathbf{C}_4$) foram atendidas, destacadas ao longo das Seções 3 e 4.

Trabalhos futuros podem concentrar esforços na redução do custo computacional da extração de esquemas via LLMs, mantendo ou ampliando a qualidade dos resultados. Entre as possíveis direções de pesquisa destacam-se: (*i*) investigar o uso de modelos generalistas menores, complementada por *fine-tuning* específico para a tarefa de extração de esquemas e (*ii*) desenvolvimento de modelos treinados do zero, projetados exclusivamente para compreender e inferir a estrutura de documentos JSON.

Referências

Abdelhedi, F., Brahim, A. A., Rajhi, H., Ferhat, R. T., and Zurfluh, G. (2021). Automatic extraction of a document-oriented nosql schema. In *ICEIS (1)*, pages 192–199.

Baazizi, M.-A., Colazzo, D., Ghelli, G., and Sartiani, C. (2019). Parametric schema inference for massive JSON datasets. *The VLDB Journal*, 28:497–521.

Banhara, N., Schreiner, G. A., da Silva Feitosa, S., and Duarte, D. (2024). Enumeration, tagged unions, tuples, and collections: A novel approach to extracting json schema. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 234–246. SBC.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Bouchou, B. and Duarte, D. (2007). Assisting XML schema evolution that preserve validity. In *Brazilian Database Symposium*, pages 270–284.

Bourhis, P., Reutter, J. L., Suárez, F., and Vrgoč, D. (2017). Json: data model, query languages and schema specification. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pages 123–135.

Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., Persson, K. A., and Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Frozza, A. A., dos Santos Mello, R., and da Costa, F. d. S. (2018). An approach for schema extraction of json and extended json document collections. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 356–363. IEEE.

Kellou-Menouer, K., Kardoulakis, N., Troullinou, G., Kedad, Z., Plexousakis, D., and Kondylakis, H. (2022). A survey on semantic schema discovery. *The VLDB Journal*, 31(4):675–710.

Klessinger, S., Klettke, M., Störl, U., and Scherzinger, S. (2023). Extracting JSON schemas with tagged unions. *arXiv preprint arXiv:2306.07085*.

Lima, M., Duarte, D., Schreiner, G., Salton, G., and Feitosa, S. (2024). Automated edge-case discovery in JSON schema validation: A differential testing approach powered by LLMs. TCC - UFFS.

Maiwald, B., Riedle, B., and Scherzinger, S. (2019). What are real json schemas like? In *International Conference on Conceptual Modeling*, pages 95–105. Springer.

Mior, M. J. (2024). Large language models for json schema discovery. *arXiv preprint arXiv:2407.03286*.

MongoDB Inc. (2024). Mongodb documentation. `https://www.mongodb.com/docs/manual/`. Acesso em: 13 maio 2025.

Namba, J. (2021). Enhancing json schema discovery by uncovering hidden data. In *PhD@ VLDB*.

Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., and Vrgoč, D. (2016). Foundations of json schema. In *Proceedings of the 25th international conference on World Wide Web*, pages 263–273.

Silva, L. O., Silva, P. H., and Silva, F. A. (2025). Leis de escala para text-to-sql: Um estudo sobre a relação entre tamanho e desempenho de modelos de linguagem. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 140–153. SBC.

Spoth, W., Kennedy, O., Lu, Y., Hammerschmidt, B., and Liu, Z. H. (2021). Reducing ambiguity in JSON schema discovery. In *Proceedings of the 2021 SIGMOD*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, K. and Liu, H. (1997). Schema discovery for semistructured data. In *KDD*, volume 97, pages 271–274.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Yun, J., Tak, B., and Han, W.-S. (2024). Recg: Bottom-up json schema discovery using a repetitive cluster-and-generalize framework. *Proc. VLDB Endow.*, 17(11):3538–3550.