UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THIAGO CHAFADO ALMEIDA

EXTRAÇÃO DE ESQUEMAS DE DADOS SEMI-ESTRUTURADOS (JSON) UTILIZANDO LARGE LANGUAGE MODEL (LLM)

CHAPECÓ
2025

THIAGO CHAFADO ALMEIDA

EXTRAÇÃO DE ESQUEMAS DE DADOS SEMI-ESTRUTURADOS (JSON) UTILIZANDO LARGE LANGUAGE MODEL (LLM)

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS), como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Denio Duarte

Este Trabalho de Conclusão de Curso foi avaliado e aprovado pela banca avaliadora em: 10/12/2025.

BANCA AVALIADORA

Denio Duarte - UFFS

Geomar Andre Schreiner - UFFS

Samuel da Silva Feitosa - UFFS

SBC Computing Reviews, 202X, XX:1
ISSN: 2966-3938 • doi: 10.5753/rocs.202X.XXXXXX

Revisão da Literatura

Extração de Esquemas de Dados Semi-estruturados(JSON) Utilizando Large Language Model (LLM)

Extracting Schemas from Semi-Structured Data (JSON) Using Large Language Model (LLM)

Thiago Chafado Almeida [Universidade Federal da Fronteira Sul | *thiagochafado123@gmail.com*]
Denio Duarte [Universidade Federal da Fronteira Sul | *duarte@uffs.edu.br*]

Universidade Federal da Fronteira Sul - Campus Chapecó,Rodovia SC 484 - Km 02, Fronteira Sul Chapecó, SC - Brasil, CEP 89815-899

Resumo. Este estudo investiga a aplicação de Modelos de Linguagem de Grande Escala (LLMs), especificamente Gemma 3-4B e Qwen 2.5-14B, na extração automática de esquemas de dados a partir de arquivos JSON. Considerando os desafios impostos pela heterogeneidade e natureza dinâmica dos dados semiestruturados, propõe-se uma abordagem baseada em um pipeline de fragmentação, inferência e fusão probabilística de esquemas. A validação, realizada através do validador Ajv em coleções reais e sintéticas, demonstrou taxas de sucesso superiores a 99%, comprovando a capacidade de generalização da abordagem. Contudo, os experimentos evidenciaram que o elevado custo computacional e o tempo de processamento dos modelos maiores ainda representam desafios significativos para a escalabilidade da solução em ambientes com recursos limitados.

Abstract. This study investigates the application of Large Language Models (LLMs), specifically Gemma 3-4B and Qwen 2.5-14B, for the automatic extraction of data schemas from JSON files. Considering the challenges imposed by the heterogeneity and dynamic nature of semi-structured data, an approach based on a pipeline of fragmentation, inference, and probabilistic schema fusion is proposed. The validation, performed using the Ajv validator on real-world and synthetic datasets, demonstrated success rates exceeding 99%, confirming the generalization capability of the approach. However, the experiments highlighted that the high computational cost and processing time of larger models still pose significant challenges to the scalability of the solution in resource-constrained environments.

Palavras-chave: Extração de Esquemas, Large Language Models, JSON, Aprendizado de Máquina, Extração de Dados

Keywords: Schema Discovery, Large Language Models, Json, Machine Learning, Data Extraction

Recebido/Received: DD Month YYYY • **Aceito/Accepted:** DD Month YYYY • **Publicado/Published:** DD Month YYYY

1 Introdução

O crescimento acelerado na geração e circulação de dados nas últimas décadas transformou a forma como sistemas computacionais são projetados, integrados e explorados. Nesse cenário, formatos flexíveis e de fácil intercâmbio tornaram-se indispensáveis para representar informações estruturadas e semi-estruturadas. Entre esses formatos, o *JavaScript Object Notation* (JSON) se consolidou como um padrão amplamente adotado em aplicações Web, serviços em nuvem, plataformas de *data analytics* e sistemas de bancos de dados NoSQL. Sua maleabilidade, embora vantajosa do ponto de vista operacional, introduz desafios importantes relacionados à padronização, validação e compreensão estrutural dos dados que seguem sendo desafio até os dias atuais [Pezoa *et al*., 2016].

Diferentemente dos bancos de dados relacionais, em que a estrutura dos dados é rigidamente definida por esquemas previamente estabelecidos, os documentos JSON podem ser publicados e consumidos sem qualquer especificação estrutural explícita. Embora essa característica facilite a interoperabilidade entre sistemas heterogêneos, ela compromete atividades essenciais como validação automática, integração, recuperação de informações, documentação e controle de qualidade. Nesse contexto, torna-se necessária a aplicação de técnicas de *extração de esquemas*, que buscam identificar, de forma automática ou semi-automática, a organização intrínseca dos dados, incluindo entidades, atributos, tipos e relações [Wang and Liu, 1997; Frozza *et al*., 2018]. Essa etapa contribui para reduzir ambiguidade semântica, aumentar a confiabilidade dos dados e melhorar a eficiência de sistemas analíticos e transacionais.

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 disruptivos com o surgimento dos *Modelos de Linguagem de Grande Escala* (*Large Language Models* – LLMs). Treina-

dos com quantidades massivas de dados textuais e baseados na arquitetura *Transformer* [Vaswani *et al*., 2017], os LLMs demonstraram capacidades emergentes em tarefas relacionadas ao raciocínio, síntese e análise estrutural de informações textuais [Wei *et al*., 2022]. Essas capacidades os tornam candidatos promissores 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.

Diante desse panorama, este trabalho investiga o uso de LLMs como ferramenta para extração automática de esquemas de documentos JSON. A proposta consiste em explorar a capacidade desses modelos para inferir estruturas, tipos e relações presentes em coleções heterogêneas de documentos, avaliando seu desempenho em cenários que variam em complexidade estrutural e nível de ruído nos dados. Para isso, foram definidos três coleções de documentos reais, além da criação de uma coleção sintética. Para a extração dos esquemas, foram utilizados dois modelos, sendo eles o *Gemma3-4B* e o *Qwen 2.5-14B*, um deles sendo utilizados para documentos classificados como de baixa complexidade, e o outro para documentos classificados como alta complexidade. Após a inferência dos esquemas, os mesmos foram fusionados em seus respectivos esquemas mestres e avaliados utilizando o validador de esquemas *Javascript Ajv* utilizando os documentos fontes dos esquemas extraídos. Os resultados obtidos se mostraram satisfatórios, embora o custo para a extração foi considerada elevada para cenários com hardware limitado.

A estrutura deste trabalho está organizada da seguinte forma: a próxima seção apresenta o referencial teórico sobre extração de esquemas, com ênfase em documentos JSON. Na Seção 3, são discutidos conceitos fundamentais sobre LLMs e suas capacidades. Em seguida, na Seção 4, descreve a metodologia adotada, contemplando desde seu pipeline, coleta e preparação das coleções, extração dos esquemas utilizando os modelos, abordagem para a fusão dos esquemas e o mecanismo de avaliação. A Seção 5 apresenta e discute os resultados obtidos. Após, a Seção 6 aborda alguns trabalhos relacionados. Por fim, a Seção 7 resume os resultados e finaliza o trabalho.

2 Extração de Esquemas

A extração de esquemas, no contexto da ciência da computação e, mais especificamente, 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].

Nesta seção, serão apresentados, brevemente, os principais conceitos que fundamentam a extração de esquemas, com ênfase especial em dados no formato JSON pois é o foco deste trabalho, dada sua ampla adoção em aplicações modernas. Inicialmente, será discutida a estrutura e as particularidades do JSON, seguida por uma análise do que caracteriza um esquema de dados nesse contexto.

2.1 JSON

JSON, ou *Java Script Object Notation*, é um formato de dados semi-estruturado utilizado para armazenar informações entre sistemas, comumente usado em sistemas Web. É baseado em texto e utiliza pares de chave-valor para representar os dados, podendo ser simples (atômicos), listas (*arrays*) ou estruturas mais complexas (objetos aninhados). A Figura 1 exemplifica um documento JSON fictício.

- **Atributos atômicos**: são pares chave-valor em que o valor é um tipo de dado primitivo, como uma *string*, número ou valor booleano. Por exemplo:
  - Na linha 2, `"title": "The Lord of the Rings"` é um atributo atômico com valor textual.
  - Na linha 3, `"published_in": 1954` representa um valor numérico.
  - Na linha 4, `"adapted_to_film": true` representa um valor booleano.
- **Arrays**: são coleções ordenadas de valores, delimitadas por colchetes (`[]`). Os elementos podem ser atômicos ou objetos. Por exemplo:
  - Na linha 8, `"main_characters": ["Frodo", "Sam", "Gandalf", "Aragorn"]` é um array de strings.
- **Objetos**: são estruturas delimitadas por chaves (`{}`) compostas por um ou mais pares chave-valor. Podem estar aninhados dentro de outros objetos ou arrays. Exemplo:
  - Na linha 5, a chave `books` contém um array de objetos, onde cada objeto representa um dos três volumes da trilogia de Senhor dos Anéis, com seu próprio título e personagens, estes podendo ser atributos atômicos, arrays, ou até mesmo outros objetos.

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].

2.2 Esquemas

Um esquema representa a estrutura lógica de dados, especificando os tipos de elementos contidos, suas relações e restrições de organização. Trata-se de uma descrição formal que serve como um modelo de referência para interpretar os dados presentes em uma determinada fonte, como arquivos, bancos de dados ou APIs. Para banco de dados relacionais, o esquema é pré-definido pois os dados contidos são estruturados, mas em banco de dados NoSQL, por exemplo, os dados são geralmente não estruturados ou semi-estruturados

Figura 1. Exemplo de objeto JSON

```
1  {
2    "title": "The Lord of the Rings",
3    "author": "J.R.R. Tolkien",
4    "published_in": 1954,
5    "books": [
6      {
7        "title": "The Fellowship of the Ring",
8        "main_characters": ["Frodo", "Sam", "
             Gandalf", "Aragorn"]
9      },
10     {
11       "title": "The Two Towers",
12       "main_characters": ["Frodo", "Sam", "
             Gollum", "Legolas", "Gimli"]
13     },
14     {
15       "title": "The Return of the King",
16       "main_characters": ["Frodo", "Sam", "
             Aragorn", "Gandalf"]
17     }
18   ],
19   "genre": "Fantasy",
20   "adapted_to_film": true
21 }
```

Figura 2. Esquema correspondente ao objeto JSON da **Figura 1**

```
1  {
2    "type": "object",
3    "properties": {
4      "title": { "type": "string" },
5      "author": { "type": "string" },
6      "published_in": { "type": "integer" },
7      "books": {
8        "type": "array",
9        "items": {
10         "type": "object",
11         "properties": {
12           "title": { "type": "string" },
13           "main_characters": {
14             "type": "array",
15             "items": { "type": "string" }
16           }
17         },
18         "required": ["title", "
               main_characters"]
19       }
20     },
21     "genre": { "type": "string" },
22     "adapted_to_film": { "type": "boolean" }
23   },
24   "required": ["title", "author", "
         published_in", "books", "genre", "
         adapted_to_film"]
25 }
```

[Namba, 2021]. A definição de um esquema auxilia na detecção de erros, facilita a documentação automática e é fundamental para tarefas como integração de dados, extração de informações e geração de consultas.

Como mencionado na Seção 2.1, o formato JSON é semi-estruturado, o que pode dificultar uma consulta ou extração de informações. Criar um esquema para documentos JSON é um trabalho que segue em progresso até os dias atuais [Pezoa *et al.*, 2016].

A Figura 2 apresenta uma proposta de esquema para o documento JSON apresentado na Figura 1. A partir do esquema apresentado na Figura 2, observa-se que o tipo principal do documento é um objeto (*object*), o que indica que os dados estão organizados por meio de pares chave-valor. Dentro deste objeto, encontra-se diversos atributos atômicos, presentes a partir da linha 3, como `title`, `author`, `published_in`, `genre` e `adapted_to_film`, sendo cada um associado a um tipo primitivo, como *string*, *integer* ou *boolean*. Esses atributos representam informações básicas e indivisíveis sobre a obra descrita.

Além disso, na linha 7, o campo `books` é caracterizado como um *array* de objetos. Cada objeto neste *array* possui sua própria estrutura interna, contendo um título e uma lista de personagens principais. Essa organização indica a presença de hierarquia e aninhamento dentro do esquema, possibilitando a representação de relações mais complexas entre os dados. Nota-se ainda na linha 13 que o campo `main_characters` é definido como um *array* de *strings*, representando uma lista simples de nomes.

O uso do elemento `type` para cada campo permite que sistemas de validação reconheçam automaticamente o tipo esperado para cada valor, enquanto a inclusão da chave `required` presente nas linhas 18 e 24 especifica quais atributos são obrigatórios. Isso é fundamental para garantir a consistência e a integridade dos dados armazenados ou transmitidos.

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. Além disso, promovem maior confiabilidade e transparência, atributos essenciais especialmente em contextos colaborativos ou científicos, onde a consistência estrutural dos dados impacta diretamente sua qualidade e utilidade.

Existem várias abordagens de extração de esquemas de documentos JSON na literatura. A maioria delas se apoia em estrutura de dados específicas para encontrar a estrutura que define os documentos, por exemplo, grafos ou árvores. Porém, este trabalho utiliza LLM para extrair tais estruturas (esquemas), assim, o próximo capítulo apresenta brevemente conceitos sobre LLM.

3 Modelos de Linguagem de Grande Escala (LLMs)

Modelos de Linguagem de Grande Escala, comumente chamado pelo seu termo em inglês LLM, são modelos computacionais baseados em técnicas avançadas de aprendizado profundo, desenvolvidos para interpretar e gerar textos, de maneira humana ou estruturada.

3.1 Contextualização dos Modelos de Linguagem

Estes modelos surgiram como evolução dos primeiros modelos estatísticos de linguagem, como os modelos n-gramas, e posteriormente dos modelos baseados em redes neurais re-

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

Dataset	Tamanho Original	Docs Originais	Docs na Amostra	Tamanho da Amostra	Min Chaves	Max Chaves	Média Chaves	Min Prof.	Max Prof.	Média Prof.
air-bnb-listings	42.45 MB	83711	83711	49.70 MB	19	19	19.0	3	3	3.0
roam_prescription	156.39 MB	239930	119965	105.96 MB	11	416	30.1	3	3	3.0
dataset_sintético	5.03 MB	5600	5600	5.03 MB	26	26	26.0	3	3	3.0
twitter	10.74 GB	1945365	14200	127.32 MB	74	771	186.9	4	11	7.5

correntes (RNNs) e LSTMs[1]. 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.

Os LLMs são treinados com tarefas de predição de próxima palavra (*next token prediction*) ou preenchimento de lacunas, o que os capacita a capturar não apenas padrões linguísticos superficiais, mas também aspectos semânticos, pragmáticos e até mesmo conhecimentos enciclopédicos contidos nos dados de treinamento [Devlin *et al*., 2019].

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].

4 Metodologia

Esta seção aborda a metodologia do estudo, mostrando seu pipeline, coleções escolhidas, pré-processamento destas coleções, extração utilizando as LLMs e fusionamento dos esquemas.

4.1 Pipeline

Dada uma coleção de documentos C, o pipeline de processamento é definido pelas seguintes etapas:

1. **Fragmentação:** A coleção C é composta em um conjunto $D = \{d_1, d_2, \ldots, d_k\}$ de k documentos JSON individuais.
2. **Amostragem:** Um subconjunto $S \subseteq D$ é selecionado aleatoriamente de D, resultando em uma amostra de n documentos (onde $n \leq k$).
3. **Análise de Complexidade:** Cada documento d_i pertencente à amostra S é classificado como sendo de *alta* ou *baixa* complexidade. Esta análise é detalhada na Seção 4.4.
4. **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.
5. **Fusão de Esquemas:** O conjunto de n esquemas E_S é submetido ao processo de fusão (detalhado na Seção 4.6), resultando em um único esquema mestre E_M consolidado.
6. **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.

4.2 Coleta e pré-processamento das coleções JSON

Esta seção aborda a coleta e processamentos necessários das coleções JSON escolhidas para a realização do trabalho, bem como as motivações para tais escolhas.

4.3 Coleções

Para o trabalho, foram selecionadas três coleções de documentos JSON de dados reais e uma coleção sintética, sendo estas:

1. **Russian election 2018 — twitter user activity:**[2] Coleção que contém dados sobre posts do twitter relacionados as eleições russas de 2018.
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.
3. **Airbnb listings London-UK**[4]**:** Coleção contendo dados sobre locações do Airbnb de Londres.
4. **Dados sintéticos:** Coleção criada de forma sintética utilizado para avaliar escalabilidade da proposta. Detalhada na Seção 4.7.

4.4 Pré-processamento

Inicialmente, as coleções passam por uma série de pré-processamentos:

1. **Amostragem Inteligente e Fragmentação:** A coleção de entrada passa por um processo de amostragem inteligente 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.

[1] LSTM, ou *Long Short-Term Memory*, é um tipo de arquitetura de rede neural recorrente (RNN) projetada para lidar com o problema da dependência de longo prazo em dados sequenciais, como séries temporais

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

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

[4] `https://insideairbnb.com/london/`

- **Limite de Tamanho Proporcional:** Um tamanho máximo total para a sub-coleção, definido como uma porcentagem do tamanho do arquivo original.

A amostragem é controlada por um desses parâmetros isoladamente ou pela combinação de ambos.
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, que só conseguem processar um número fixo de tokens por requisição [Vaswani *et al*., 2017].

2. **Análise de complexidade:** Cada documento JSON individual é submetido a uma análise de complexidade, sendo então classificado em baixa complexidade ou alta complexidade. Essa análise será útil para a extração, sendo assim possível utilizar um modelo de menor escala a fim de otimizar o tempo de extração. Esta complexidade é feita a partir dos seguintes critérios e *Thresholds*:

   - **Número de Chaves:** A contagem total de chaves (propriedades) no documento.
   - **Profundidade de Aninhamento:** O nível máximo de aninhamento de objetos ou arrays dentro da estrutura do documento.
   - **Tamanho do Arquivo:** O tamanho total do documento em bytes.

   Com base nesses critérios, por meio de testes empiricos analisando a Tabela 1 e utilizando como base experimentos em que o modelo alucinou ou falhou em resolver a tarefa (veja Apêndice 8.2), 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 for superior a 30.000 bytes.

   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.

A Tabela 1 mostra alguns dados das coleções usadas no estudo. Note que todos os documentos da coleção `twitter` foram classificados com alta complexidade, devido ao grau de profundidade ser no mínimo 4 e grande maioria de seus documentos ter o número de chaves acima de 100. As outras coleções foram classificadas como baixa complexidade, com exceção de alguns poucos documentos de `roam_prescription`, 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 na Seção 5.1

Após a etapa de pré-processamento, foi iniciada a etapa de extração dos esquemas.

4.5 Extração dos esquemas

Para a extração dos esquemas, foram definidos duas 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. Para documentos de complexidade alta, o mesmo alucinou em boa parte dos casos.
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 específico, utilizando um notebook Macbook Air equipado com um processador Apple M4 com 16 GB de memória RAM compartilhada. A aceleração de hardware para as tarefas de inferência foi provida pela GPU integrada do chip, por meio do framework de aprendizado de máquina MLX com suporte à API Metal da Apple.

Os parâmetros para a amostragem apresentado na Seção 4.4 foram definidos como isoladamente, com o **limite de documentos** sendo 1000 documentos.

O Prompt de extração foi o mesmo para ambos os modelos, sendo:

```
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. A análise dos resultados apresentados revela uma observação principal:

- **Impacto do Tamanho do Modelo:** O dataset `twitter`, classificado como de alta complexidade pela análise de pré-processamento, exigiu o uso do modelo Qwen 2.5-14B. 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.

[5] `https://huggingface.co/google/gemma-3-4b-it`
[6] `https://huggingface.co/Qwen/Qwen2.5-14B`

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

Coleção	Documentos Analisados	Tempo Médio (s)	Tempo Total (h)	Pico de Memória (GB)
roam_prescription	1000	51.56	14.32	3.0
air-bnb-listings	1000	69.52	19.31	3.0
dataset_sintético	500	24.09	3.35	3.0
twitter	1000	604.32	167.87	13.0

Figura 3. Exemplo de Json recebido pelo modelo

```
        ........
        "translator_type": "regular",
        "protected": false,
        "verified": false,
        "followers_count": 7248,
        "friends_count": 7025,
        "listed_count": 72,
        "favourites_count": 1441,
        "statuses_count": 70099,
        "created_at": "Fri May 11 08:53:38 +
            0000 2012",
        ........
```

Figura 4. Extração feita pelo modelo referente à Figura 3

```
        ........
        "translator_type": {
          "type": "string"
        },
        "protected": {
          "type": "boolean"
        },
        "verified": {
          "type": "boolean"
        },
        "followers_count": {
          "type": "integer"
        },
        "friends_count": {
          "type": "integer"
        },
        "listed_count": {
          "type": "integer"
        },
        "favourites_count": {
          "type": "integer"
        },
        "statuses_count": {
          "type": "integer"
        },
        "created_at": {
          "type": "string"
        },
        ........
```

- **Influência da Densidade da Informação:** Mesmo utilizando o mesmo modelo (Gemma 3-4B), notou-se que os tempos de processamento para as coleções reais (`air-bnb-listings` e `roam-prescription`) foram consideravelmente maiores do que para o `dataset-sintético`. 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 a LLM, resultando em um tempo de processamento maior por documento.

A Figura 3 mostra um trecho de um documento da coleção `twitter`, enquanto a Figura 4 apresenta a estrutura inferida pela LLM a partir desse mesmo conteúdo. Nota-se que o modelo reconhece as chaves e identifica com precisão os tipos correspondentes, gerando uma descrição estruturada do documento.

Após realizada a extração dos esquemas, é feita a fusão dos mesmos. Desta forma, cada coleção JSON, que anteriormente foi separada em documentos individuais, é consolidada em seus respectivos esquemas mestres. Este processo é executado por diretório, garantindo que cada coleção de dados tenha seu próprio esquema mestre representativo.

4.6 Fusão dos Esquemas

A metodologia de fusão é dividida em duas fases principais:

1. **Fase de Análise e Reparo:** Nesta primeira etapa, o algoritmo itera sobre cada schema individual gerado pelo LLM dentro de um diretório específico. Para cada arquivo, são executadas as seguintes sub-etapas:
    - **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). Para cada propriedade (nó) nesta árvore, 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`, `null`). A Figura 5 ilustra uma *árvore de estatísticas* fictícia, na qual cada propriedade do esquema é representada como um nó preservando a hierarquia do docu-

mento. Para cada nó, o campo `_stats` armazena o número de aparições (`appearances`), a frequência com que a propriedade foi marcada como obrigatória (`required_count`) e a distribuição dos tipos observados (`type_counts`). Essa representação permite observar simultaneamente a estrutura do dataset e a consistência ou variação de cada propriedade ao longo dos documentos. As estatísticas extraídas nessa etapa são utilizadas posteriormente para definir a obrigatoriedade dos campos e os tipos aceitos durante a construção do schema mestre.

2. **Fase de Geração do Esquema Mestre:** Após a análise de todos os arquivos válidos do diretório, a árvore de estatísticas consolidada é utilizada para construir o schema 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 schema mestre se a sua taxa de obrigatoriedade for maior ou igual ao `REQUIRED_THRESHOLD`. Esta taxa é calculada dividindo a contagem de obrigatoriedade de uma propriedade (o número de vezes que o LLM a marcou como `'required'`) pelo **número total de documentos válidos** analisados na amostra (e.g., 1.000). Para os experimentos finais, este limiar foi definido como **1.0**, aplicando uma regra estrita de que um campo deve ser marcado como obrigatório em 100% de todos os documentos da amostra para ser considerado como tal no esquema mestre.
    - **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, para garantir a compatibilidade com estruturas de dados heterogêneas, uma lista de todos os tipos observados é utilizada (e.g., `["string", "null"]`).

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

4.7 Coleção Sintética

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*[7], 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 destinadas a introduzir variabilidade e "ruído" controlado. A entidade *Transação*, por sua vez, incorpora outras estruturas, incluindo objetos e *arrays*, possibilitando avaliar a capacidade das LLMs em inferir esquemas em contextos aninhados.

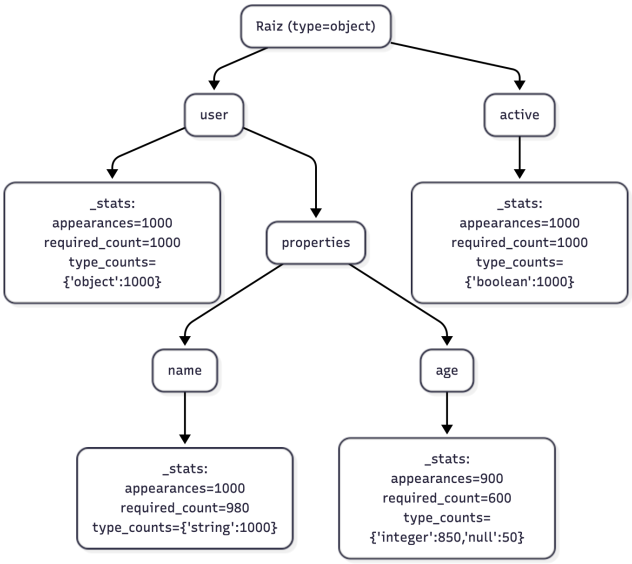


Figura 5. Exemplo de árvore de estatísticas fictícia

A estrutura da coleção foi definida da seguinte forma:

- **Pessoa** — documento composto pelos seguintes campos:
    - `id` (integer)
    - `name` (string)
    - `email` (string | null)
    - `age` (integer)
    - `phone` (string | null)
    - `city` (string)
    - `state` (string)
    - `registered_at` (string — formato ISO 8601)

  Os campos `email` e `phone` são gerados apenas de forma probabilística, a fim de simular valores ausentes comuns em bases reais.
- **Produto** — documento com as chaves:
    - `id` (integer)
    - `name` (string)
    - `category` (string)
    - `price` (number)
    - `in_stock` (boolean)
    - `rating` (number, faixa 1.0–5.0)
    - `created_at` (string — ISO 8601)
- **Transação** — documento contendo:
    - `id` (integer)
    - `person_id` (integer)
    - `date` (string — ISO 8601)
    - `status` (string)
    - `payment_method` (string)

  Além disso, cada transação inclui um `array` chamado `items`, no qual cada item é representado por um objeto com os campos:
    - `id` (integer)
    - `transaction_id` (integer)
    - `product_id` (integer)
    - `quantity` (integer)
    - `price` (number)

[7] `https://faker.readthedocs.io/en/master/`

Essa modelagem 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 das LLMs na inferência de esquemas e permitem observar o impacto da complexidade estrutural no desempenho dos modelos. O modelo lógico completo da coleção pode ser consultado no Apêndice 8.1.

4.8 Avaliação dos Esquemas Mestres

Para avaliar a qualidade dos esquemas mestres gerados, utilizou-se o validador de esquemas *JavaScript Ajv*[8]. 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 JSON Schema em JavaScript e outras linguagens, identificou que os validadores em JavaScript apresentam poucas inconsistências, reforçando a confiabilidade da ferramenta escolhida para este trabalho.

5 Resultados e Discussões

A Tabela 3 apresenta os resultados da validação dos esquemas mestres gerados pelo processo de fusão orientado por LLMs. Nesta etapa, o objetivo principal foi avaliar a capacidade dos modelos de linguagem em generalizar corretamente os padrões estruturais presentes em coleções de dados reais de diferentes domínios.

Os resultados demonstram que ambos os modelos exibiram desempenho robusto na tarefa de inferência e validação de esquemas. O modelo Gemma3-4B atingiu próximo de 100% de sucesso, sugerindo que, em cenários menos complexos, modelos de menor porte são suficientes para capturar a variação estrutural dos dados. Por outro lado, o Qwen 2.5–14B 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.

As Figuras 6 e 7 ilustram um caso de reprovação observado na Tabela 3 para o dataset `air-bnb-listings`. Em alguns documentos, a chave `name` possui valor do tipo `null`. 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. Esse resultado evidencia que o threshold de 75% (valor mínimo de consenso para determinar o tipo predominante) não é infalível e pode levar à propagação de inconsistências quando a distribuição de valores na amostra difere da distribuição real da coleção.

Observando a Tabela 3, nota-se que todos os datasets

[8] `ajv.js.org/`

Figura 6. Exemplo documento reprovado da coleção `air-bnb-listings`

```
{
    "id": 8326597,
    "name": null,
    "host_id": 43664970,
    "neighbourhood": "Lambeth",
    "room_type": "Private room",
    "column_10": 50,
   .......
```

Figura 7. Trecho do esquema mestre para a coleção `air-bnb-listings`

```
{
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    },
    "name": {
      "type": "string"
    },
    "host_id": {
      "type": "integer"
    },
    "neighbourhood": {
      "type": "string"
    .......
```

obtiveram taxas de validação superiores a 99%, incluindo o dataset do Twitter, que apresenta o maior volume absoluto e o maior grau de heterogeneidade textual.

Além dos dados reais, foram conduzidos experimentos adicionais com um dataset sintético contendo documentos gerados com um pequeno nível de diversidade artificial. Para esse conjunto, foram selecionados diferentes tamanhos amostrais (100, 300 e 500 documentos) com o objetivo de avaliar não apenas a precisão dos modelos, mas também a escalabilidade da abordagem conforme o número de documentos da amostra aumenta. A Tabela 4 apresenta os resultados dessa análise. Todas as variações amostrais obtiveram taxa de sucesso de 100%, indicando que a capacidade de generalização do esquema mestre não sofre impacto significativo conforme o volume de documentos aumenta.

Ao analisar os campos marcados como `required` no esquema ilustrado na Figura 8, observa-se que o modelo inferiu corretamente os atributos obrigatórios quando comparado ao modelo conceitual apresentado no Apêndice 8.1. Além disso, algumas chaves adicionais também foram classificadas como `required`. Isso ocorreu porque essas chaves não apresentaram valores inconsistentes ou ausentes na coleção, ou seja, estiveram presentes em 100% dos documentos analisados, o que levou o algoritmo a interpretá-las como obrigatórias. Como consequência, quando determinados campos apresentam baixa variabilidade (mesmo em um conjunto de dados que contém chaves potencialmente sujas) o processo de inferência pode interpretar esses campos como obrigatórios e marcá-los como `required`, ainda que conceitualmente não o sejam.

A Figura 9 mostra um gráfico para o tempo das extrações, evidenciando que o tempo total de processamento aumenta conforme o número de documentos na amostra cresce.

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

Dataset	Total de Documentos	Aprovados	Reprovados	Taxa de Sucesso (%)
air-bnb-listings	83.711	83.686	25	99.97
roam_prescription	239.930	238.507	1.423	99.41
twitter	1.945.365	1.945.365	0	100.00

Tabela 4. Resultados da Validação do Schema Mestre (LLM) contra os Datasets Sintéticos.

Dataset	Total de Documentos	Aprovados	Reprovados	Taxa de Sucesso (%)
sintetico_100Esquemas	5.600	5.600	0	100.00
sintetico_300Esquemas	5.600	5.600	0	100.00
sintetico_500Esquemas	5.600	5.600	0	100.00

Figura 8. Trecho contendo campos `required`

```
{
    ........
    "person" :{
      "required": [
        "id",
        "name"
      ]
    },
    ........
    },
    "product":{
      "required": [
        "id",
        "name",
        "price"
      ]
    },
    .......
    "transaction":{
      "required": [
        "date",
        "id",
        "person_id",

        "status"
      ]
```

Embora a tendência geral seja linear, a inclinação não é constante entre todos os intervalos: o salto de 100 para 300 documentos (0.83h → 2.13h) é levemente maior do que o salto de 300 para 500 documentos (2.13h → 3.35h). Essa variação indica que o custo computacional depende não apenas do modelo, mas também da distribuição interna de estruturas presentes no documento. Ainda assim, o crescimento temporal permanece previsível e dentro de uma faixa esperada para processos de inferência e fusão em larga escala.

5.1 Limitações

Uma das limitações comuns para o uso das 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. 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.

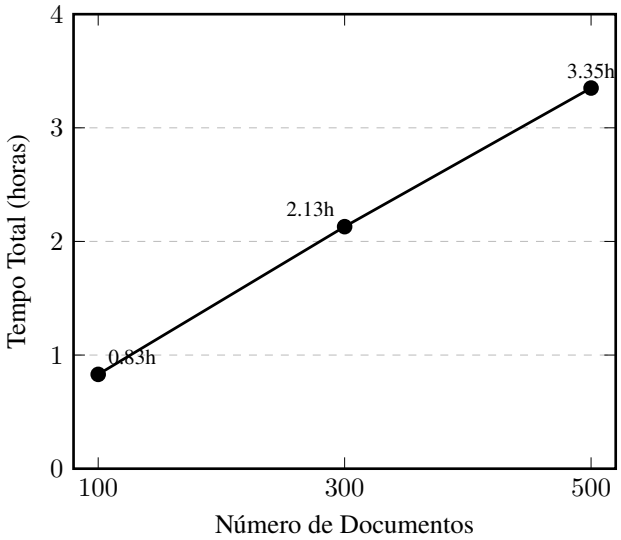


Figura 9. Tempo de extração no dataset sintético

Além dessa limitação inicial, outras restrições importantes foram identificadas durante o desenvolvimento deste trabalho:

1. **Dependência da representatividade da amostra.** O processo de fusão de esquemas assume que a amostra utilizada contém, ao menos uma vez, todas as estruturas relevantes do dataset. Caso campos raros, padrões incomuns ou estruturas aninhadas pouco frequentes não apareçam na amostra, o esquema mestre resultante tende a ser incompleto. Isso afeta diretamente a capacidade de generalização do modelo e pode levar a falsos positivos durante a validação em que documentos válidos são marcados como incorretos por ausência de chaves esperadas. Esse problema pode ser mitigado como feito com a amostra randômica, que busca aleatorizar as chaves e consequentemente conseguir um número menor de ausências.
2. **Dependência do modelo e de sua capacidade contextual.** Modelos menores, como os utilizados em documentos de baixa complexidade, podem não conseguir identificar padrões estruturais mais sutis. Isso pode gerar esquemas mais restritos, incompletos ou até mesmo, como pode ser visto no Apêndice 8.2, resultar um desvio completo do objetivo, gerando um conteúdo que não corresponde ao documento de entrada nem ao esquema esperado. Por outro lado, modelos maiores, embora mais precisos, possuem custos de inferência significati-

vamente superiores e exigem maior capacidade computacional, o que pode limitar o uso prático da abordagem em ambientes com restrições de hardware.

3. **Dificuldade em inferir enumerações.** Durante os experimentos, observou-se que, em alguns cenários, o modelo tenta inferir listas de valores possíveis (*enum*) a partir de um único documento por vez. Como não há visibilidade global do conjunto de documentos no momento da geração (devido a necessidade de quebrar o dataset em documentos e extrair um por vez), o modelo tende a assumir generalizações precipitadas, criando valores que não existem no dataset ou omitindo valores legítimos que não estiveram presentes no documento analisado. Esse comportamento resulta em erros tanto de tipo quanto de chave, com a inserção de atributos ou categorias inexistentes no dataset real. Esse efeito decorre diretamente da natureza amostral do processo e pode ser mitigado fornecendo contexto adicional ou aplicando validação posterior baseada em frequência de valores ao longo da coleção.

6 Trabalhos Relacionados

Este capítulo apresenta cinco trabalhos relacionados selecionados mais similares ao estudo proposto. O estudo de Frozza *et al*. [2018] destaca a crescente adoção do formato JSON, impulsionada pela proliferação dos bancos de dados não-relacionais (NoSQL). Os autores ressaltam que a ausência de um esquema predefinido nesses arquivos representa um desafio considerável para a recuperação, análise e consulta dos dados. Em resposta a essa problemática, o trabalho propõe uma abordagem que utiliza operações de agregação para extrair e consolidar o esquema de uma coleção de arquivos JSON, com o intuito de gerar um esquema global e representativo. O processo é definido em quatro etapas:

1. Geração de esquemas brutos de documentos: o primeiro passo é gerar um esquema bruto de cada documento e armazená-la em uma coleção *MongoDB* temporária, contendo a mesma estrutura do documento original, e substituindo cada valor primitivo por seu *JSON data type* (*e.g.*, string, bool, number).
2. Agrupamento de esquemas brutos de documentos: após a extração, é aplicado duas operações de agrupamento nos dados brutos. A primeira agrupa documentos que têm o mesmo esquema bruto, armazenando o resultado em uma coleção temporária, sendo utilizada pela segunda agregação após um ordenamento dos atributos em ordem alfabética, com o intuito de reduzir duplicatas. A agregação então é aplicada removendo estas duplicadas resultando no menor número possível de documentos válidos.
3. Unificação de esquemas brutos de documentos: após o agrupamento, é definida uma estrutura baseada em árvore chamada *Raw Schema Unified Structure* (RSUS) utilizada para armazenar informações hierárquicas dos esquemas brutos.
4. Geração de esquemas JSON: Após o resumo feito pelo RSUS, é aplicado um algoritmo que transforma estes dados brutos em um esquema JSON válido. Simplificadamente, para cada atributo da RSUS é gerado um esquema JSON respeitando sua estrutura hierárquica.

Após gerado o esquema JSON válido, os autores avaliaram o modelo utilizando diferentes coleções de documentos JSON armazenados em estruturas MongoDB, apresentando um resultado satisfatório.

Já o trabalho de Abdelhédi *et al*. [2021] propõe uma ferramenta baseada em regras de transformação para a extração de esquemas em documentos MongoDB, com o intuito de agilizar a criação de consultas. Para isso, é utilizada a Arquitetura Dirigida pelo Modelo (ADM), o qual fornece uma estrutura formal para a automação do processamento de esquemas [Hutchinson *et al*., 2011]. Com esta estrutura gerada, é aplicada então regras de transformação

Para a validação do modelo, foram analisados manualmente cada um dos esquemas extraídos. Os autores afirmam que o modelo gera um esquema satisfatório, e complementam que o tempo necessário para escrever consultas no banco de dados diminuiu em cerca de 50% utilizando o esquema de auxílio.

Diferentemente, aprendizado de máquina é aplicado no trabalho de Namba [2021] para distinguir os metadados de uma coleção JSON. A principal justificativa para isso se deve ao fato de que a maioria dos extratores não considera que uma parte de uma chave representa dados, mas a extrai como metadado. Estas chaves são denominadas dinâmicas.

É feita uma extração de seis atributos que descrevem diferentes aspectos das chaves JSON:

1. Domínio das Características Intrínsecas, que considera a frequência relativa (percentual de presença da chave nos documentos) e o nível de aninhamento da chave (profundidade na estrutura JSON);
2. Domínio da Tendência Central, que mede a média do número de subchaves sob uma determinada chave ao longo de todos os documentos, com a observação de que chaves dinâmicas tendem a ter mais subchaves;
3.) Domínio da Dispersão Estatística, que avalia a variação no número de subchaves por meio de métricas como amplitude, desvio padrão e entropia;
4. Domínio do Formato da Distribuição, que utiliza medidas de assimetria (skewness) e curtose (kurtosis) para analisar a forma da distribuição do número de subchaves entre os documentos;
5. Domínio da Semelhança Semântica e Contextual, que utiliza embeddings do *fastText* para calcular a distância média entre vetores de significado das subchaves, assumindo que chaves dinâmicas agrupam subchaves semanticamente mais próximas;
6. Domínio da Semelhança Estrutural, onde é aplicada a similaridade de conjuntos (Jaccard) para agrupar chaves com estruturas aninhadas semelhantes, o que reduz a chance de erros de classificação.

Esses atributos são utilizados como entrada para três algoritmos de classificação: regressão logística, *random forest* e *support vector machines* (SVM), sendo *random forest* o que obteve os melhores resultados.

Já Banhara *et al*. [2024] propõem a ferramenta JFUSE, que se diferencia de outras ao extrair *enumeration* e *tagged*

unions, além de outros dados atômicos frequentemente negligenciados por abordagens convencionais. Essa capacidade permite uma representação mais precisa e expressiva dos dados presentes em arquivos JSON, especialmente em contextos onde essas estruturas são utilizadas para definir restrições e variantes de tipos.

Para representar uma coleção de esquemas, foi escolhida uma estrutura baseada em grafos, onde cada vértice corresponde a uma tupla contendo informações relevantes: o rótulo do vértice (normalmente associado à chave JSON), o número de ocorrências daquela estrutura, um valor booleano indicando a presença de uma enumeração e outro booleano sinalizando se trata-se de uma *tagged union*. As arestas entre os vértices representam os relacionamentos hierárquicos ou de associação entre os componentes do esquema, permitindo visualizar a estrutura de forma conectada e navegável.

O algoritmo responsável por essa extração foi implementado na linguagem de programação C++, escolhida devido à sua eficiência em tarefas de processamento em larga escala, e executado em um ambiente de alto desempenho, utilizando um computador-servidor. Diversas instâncias do algoritmo foram executadas com diferentes valores de limiares, permitindo avaliar o impacto de cada parâmetro e identificar as configurações ideais para a execução final dos testes.

Dois *datasets* foram utilizados na avaliação empírica: um conjunto de exemplos oriundos de experimentos farmacêuticos e outro contendo *tweets* relacionados ao período eleitoral russo de 2018. Ambos os conjuntos de dados somam quase 12GB de informações e abrangem aproximadamente 430 milhões de chaves, o que demonstra a escalabilidade da ferramenta. Os autores concluem que os esquemas obtidos por meio da JFUSE são corretos e refletem com fidelidade as restrições estruturais implícitas nos dados originais das coleções JSON.

No mesmo contexto de utilização de aprendizado de máquina, 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). O objetivo principal da pesquisa foi treinar o modelo para, a partir de um texto científico complexo, produzir uma representação em formato JSON que encapsulasse as informações essenciais de maneira organizada e acessível.

Para alcançar tal propósito, os autores utilizaram a estratégia conhecida como sequência para sequência (*seq2seq*). Essa estratégia é amplamente utilizada no processamento de linguagem natural em que uma sequência de entrada — no caso, um trecho de artigo científico — é transformada em outra sequência — aqui, a estrutura JSON correspondente. Durante o treinamento, o foco foi moldar o comportamento do modelo para que ele conseguisse, de maneira autônoma, mapear informações não estruturadas de textos científicos para formatos estruturados que facilitam tanto o entendimento humano quanto a integração com sistemas computacionais.

Para a realização do ajuste fino, foi utilizada a arquitetura do GPT-3, sendo o modelo treinado com um conjunto de dados composto por aproximadamente 1000 pares de exemplos, onde cada par continha um trecho de texto e sua respectiva representação JSON esperada. Além disso, o autor destaca que, embora o JSON tenha sido escolhido como formato de saída neste estudo, outros formatos estruturados — como YAML ou mesmo representações em pseudocódigo — também são viáveis e podem ser explorados para aplicações distintas conforme a necessidade.

Em relação à avaliação dos resultados, o trabalho empregou uma abordagem mista: a avaliação manual, conduzida por especialistas, foi complementada por métricas estatísticas automatizadas, proporcionando uma visão abrangente da performance do modelo. Ambas as formas de avaliação indicaram que os resultados foram bastante satisfatórios, evidenciando que os LLMs, quando devidamente ajustados, possuem grande potencial para enfrentar o desafio da extração de informações estruturadas a partir de textos científicos densos e complexos.

Por fim, Mior [2024] utiliza LLMs para acrescentar informações em esquemas extraídos. Foi realizado um *fine-tunning* com pares de documentos, onde o primeiro era o esquema extraído e o segundo o mesmo esquema com semânticas e anotações adicionais. Esse treinamento supervisionado ensina o modelo a converter automaticamente representações estruturais básicas em versões anotadas, identificando relações implícitas, definindo tipos de dados, interpretando colunas e adicionando descrições semânticas que facilitam a compreensão humana e o processamento automatizado. Ao adotar esse formato par-a-par (esquema semântico $\rightarrow$ esquema anotado), o autor garante que o LLM aprenda padrões de enriquecimento semântico, em vez de simplesmente memorizar exemplos específicos.

O resultado é um modelo capaz de, ao receber um esquema extraído, gerar de forma autônoma uma versão enriquecida — anotada, descritiva e alinhada ao domínio — sem supervisão manual constante. Apesar de não realizar a extração do esquema mas sim enriquecer uma extração já realizada, essa capacidade melhora significativamente a eficácia das tarefas de documentação, integração de dados e validação de esquemas, pois reduz a intervenção humana e padroniza os metadados gerados.

Os estudos conduzidos por Frozza *et al*. [2018], Banhara *et al*. [2024] e Abdelhédi *et al*. [2021] evidenciam a crescente adoção do formato JSON em uma variedade de contextos de aplicação, destacando, contudo, que a ausência de um esquema predefinido ainda constitui um desafio significativo, persistente até os dias atuais. Esses trabalhos também indicam que, embora esse problema tenha sido objeto de investigação ao longo dos últimos anos, a busca por soluções eficientes continua em andamento.

Adicionalmente, as pesquisas de Namba [2021] e Dagdelen *et al*. [2024] apontam, respectiva mente, que o uso de aprendizado de máquina e de LLM's configura-se como uma abordagem promissora e, cada vez mais, necessária para a automatização de tarefas relacionadas à extração e estruturação de informações, contribuindo de forma efetiva para a superação das dificuldades associadas à ausência de esquemas em documentos JSON.

Diferentemente dos trabalhos de Frozza *et al*. [2018], Banhara *et al*. [2024] e Abdelhédi *et al*. [2021] que utilizam estruturas de dados como árvores e grafos e regras de transformação, a proposta deste trabalho é utilizar LLMs, semelhante aos trabalhos de Dagdelen *et al*. [2024] e de Mior [2024], mas para a tarefa específica de extrair esquemas de

documentos JSON.

7 Conclusão

Neste trabalho, foi conduzido um estudo sobre a aplicação de modelos de linguagem de grande porte (LLMs) para a tarefa de extração de esquemas a partir de dados semiestruturados, com foco no formato JSON. A proposta buscou investigar não apenas a viabilidade técnica da abordagem, mas também seu potencial como alternativa às técnicas tradicionais utilizadas atualmente para inferência de esquemas.

Para os experimentos, foram empregados dois modelos com diferentes capacidades: o *Gemma3-4B*, contendo 4 bilhões de parâmetros, e o *Qwen 2.5-14B*, com 14 bilhões de parâmetros. Essa escolha permitiu analisar o impacto da escala do modelo tanto na qualidade dos esquemas inferidos quanto no custo computacional associado ao processo. De maneira geral, ambos os modelos apresentaram desempenho sólido, alcançando taxas de validação próximas de 100% na geração de esquemas mestres compatíveis com os documentos originais, demonstrando alta capacidade de generalização estrutural mesmo em contextos com forte heterogeneidade de dados.

Além da avaliação de desempenho, este estudo buscou compreender os limites práticos do uso de LLMs para extração de esquemas. Observou-se que restrições inerentes aos modelos, especialmente o tamanho limitado da janela de contexto, podem comprometer a interpretação completa de coleções extensas. No entanto, estratégias como segmentação dos documentos, amostragem representativa e fusão posterior de esquemas parciais demonstraram ser eficazes para mitigar esse problema sem perda significativa de qualidade inferencial.

Apesar dos resultados satisfatórios, o estudo evidencia desafios relevantes. A dependência de modelos de maior porte, como o *Qwen 2.5-14B*, implica em custos computacionais significativamente elevados, tanto em termos de memória quanto de tempo de processamento. Essa limitação dificulta a adoção da abordagem em cenários reais com infraestrutura restrita, como instituições acadêmicas com hardware limitado ou sistemas corporativos que operam sob forte demanda de escalabilidade. Portanto, embora as LLMs tenham demonstrado alto potencial para a tarefa de extração de esquemas, o custo computacional atual ainda representa um obstáculo para que essa abordagem substitua plenamente algoritmos tradicionais de extração em ambientes de produção.

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.

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:

1. A investigação do uso de modelos generalistas menores, complementada por *fine-tuning* específico para a tarefa de extração de esquemas;
2. O desenvolvimento de modelos treinados do zero, projetados exclusivamente para compreender e inferir a estrutura de documentos JSON.

Disponibilidade de artefatos de pequisa

Os dados e outros materiais criados e/ou usados nesta revisão da literatura estão disponíveis em `https://github.com/ThiagoChafado/SchemaDiscovery-LLM`

Referências

Abdelhédi, 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.

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.

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.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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.

Hutchinson, J., Rouncefield, M., and Whittle, J. (2011). Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 633–642.

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. Trabalho de Conclusão de Curso aprovado, ainda não publicado.

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.

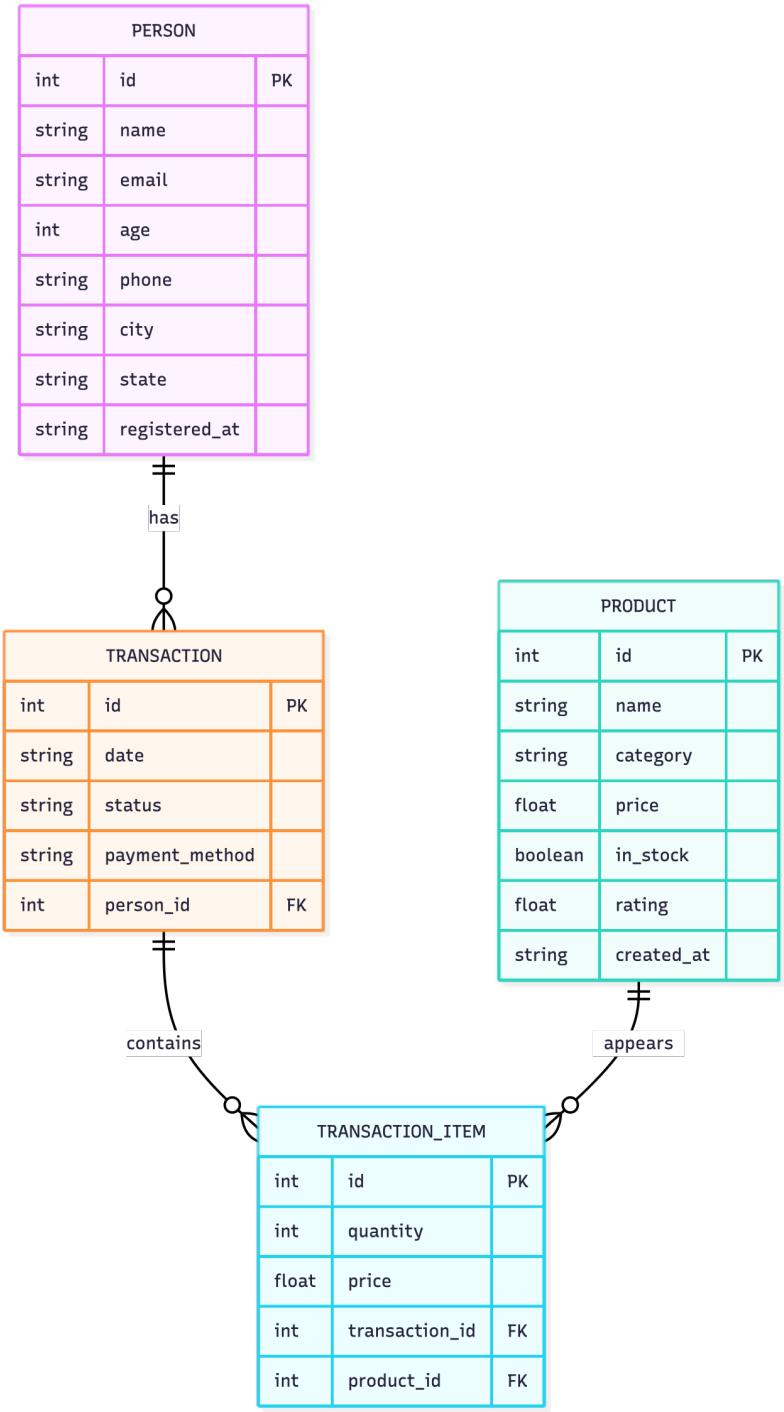
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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

Wei, J., Borgeaud, S., Byrne, W., Chollet, F., Chowdhery, A., Hessel, J., Le, Q. V., Lampinen, A., Lewkowycz, A., Mohiuddin, M., *et al*. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*.

8 Apêndice

8.1 Modelo Conceitual - Coleção Sintética



8.2 Exemplo de Alucinação do modelo Gemma3-4b


```
{
  "_sensitive": false,
  "filter_level": "low",
  "lang": "ru",
  "timestamp_ms": "1520780895235",
  "channels": {
    "putin": [
      "   "
    ]
  },
  "keywords": [
    "   "
  ]
}
```