Extração de Esquemas de Documentos JSON: O que há de Novo?

Natália Banhara[1], **Denio Duarte**[1], **Geomar Schreiner**[2]

[1]Universidade Federal da Fronteira Sul
Campus Chapecó
Chapecó – SC – Brasil

[2]Universidade do Oeste de Santa Catarina
Chapecó – SC – Brasil

natalia.banhara@outlook.com, duarte@uffs.edu.br, geomarschreiner@gmail.com

Abstract. *JSON is one of the most widely used formats for publishing and interchanging data. It combines the flexibility of semistructured data models to store data, so schemas are not mandatory. This paper extends the systematic review proposed by [Imhof et al. 2017] to show the new approaches and techniques used to extract schemas from JSON documents. We analysed seven works and concluded that JSON schema extraction is a very active study area; the new studies are using new concepts for schema extraction, e.g. machine learning.*

Resumo. *JavaScript Object Notation (JSON) tem se difundido de forma rápida e universal sendo um dos formatos mais utilizados para trocar e publicar dados. Ele é caracterizado por não ter um esquema rígido associado aos dados armazenados. Assim, neste sentido, este trabalho tem como objetivo estender o mapeamento sistemático realizado em [Imhof et al. 2017] para apresentar o que há de novo em termos de abordagens e técnicas para extrair esquemas de documentos JSON. Foram analisados sete trabalhos e verificado que a área de extração de esquemas JSON é bastante ativa e novos conceitos de extração tem surgido e.g, abordagens baseadas em aprendizado de máquina.*

1. Introdução

Um esquema descreve a estrutura dos dados armazenados em um banco de dados. Os sistemas gerenciadores de banco de dados (SGBD) tradicionais (*e.g.*, relacionais) utilizam esquemas para garantir que os dados armazenados sejam consistentes e válidos. Quando os dados respeitam as restrições impostas por um esquema, várias operações sobre os dados podem ser otimizadas [Bouchou and Duarte 2007], tais como (i) definição de uma interface para programas e usuários consultar os dados e (ii) determinar como os dados são gerenciados fisicamente nos discos. Em ambientes em que os dados necessitam ser analisados, é natural que esquemas estejam associados aos mesmos.

A produção acelerada de dados atualmente fez com que várias empresas se apoiassem em *data lakes* para o armazenamento. *Data lake* é um repositório em que dados de diversas fontes, possivelmente heterogêneas, são armazenados. A coleção de dados armazenadas neste tipo de repositório é geralmente volumosa, contendo terabytes. Também, além dos dados serem heterogêneos e volumosos, os *data lakes* são caracterizados por um

rápido tráfego de chegada de dados, tornando um desafio o pré-processamento antes do armazenamento [Zouari et al. 2021]. Essas características fazem com que os bancos de dados dos *data lakes* sejam semi (ou não) estruturados, ou seja, sem um esquema rígido associado aos dados armazenados.

Neste contexto, vários trabalhos estão sendo propostos para extrair esquemas de banco de dados. O formato de entrada mais utilizado para este tipo de extração é o *JavaScript Object Notation* (JSON). Esse formato tem se difundido de forma rápida e universal, sendo um dos formatos mais utilizados para publicar e trocar dados [Baazizi et al. 2019b].

Imhof *et al.* [Imhof et al. 2017] apresentaram uma revisão sistemática de abordagens de extração de esquemas a partir de banco de dados no formato JSON. O trabalho disserta sobre abordagens que foram publicadas entre 2012 e 2016. Com o crescente número de abordagens propostas nos últimos anos, este trabalho tem como objetivo estender a revisão feita em [Imhof et al. 2017] acrescentando trabalhos publicados entre os anos de 2018 e 2022. Esta extensão visa identificar como as abordagens evoluíram entre as duas fatias de tempo citadas.

Esta nova revisão sistemática identificou que algumas abordagens já se apoiam em aprendizado de máquina para criar modelos mais inteligentes de extração, técnicas para encontrar metadados nos dados, além de abordagens focadas em identificar variação de esquemas em coleções similares. Ou seja, as pesquisas em extração de esquemas a partir de dados semiestruturados estão ainda bem ativas e utilizando novas técnicas.

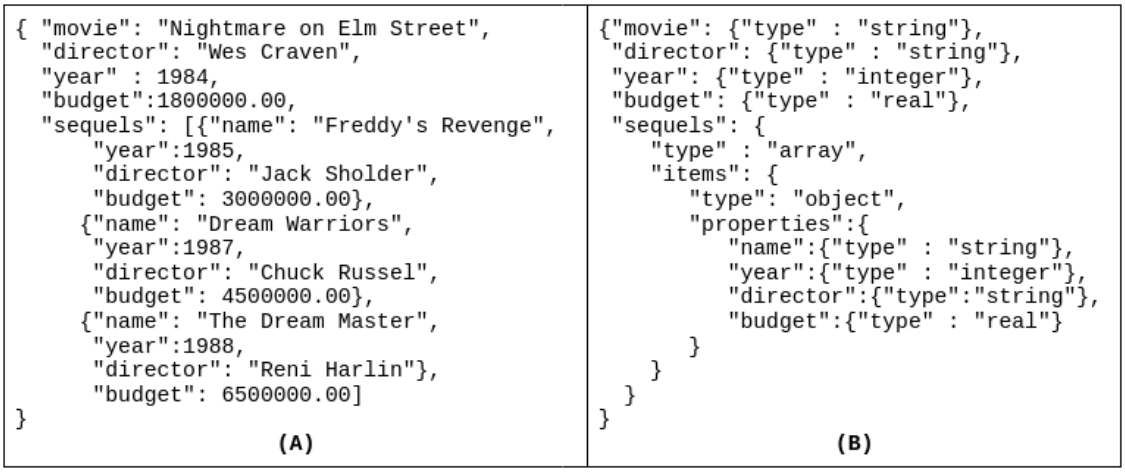
O restante do trabalho está organizado da seguinte forma: a próxima seção apresenta brevemente o formato JSON. A Seção 3 apresenta uma discussão do mapeamento sistemática realizado. Em seguida, na Seção 4, uma discussão dos trabalhos analisados e uma comparação com o estudo anterior é apresentada e a Seção 5 conclui este trabalho.

2. JSON

Esta seção apresenta brevemente o formato JSON, alvo deste estudo. JSON vem se tornando o formato universal para publicar e trocar dados entre aplicações [Baazizi et al. 2019b]. Além disso, é o formato mais empregado em SGBDs NoSQL orientados a documentos. Neste trabalho, um arquivo com dados estruturados no formato JSON será chamado de documento.

O formato JSON é simples: os dados são armazenados no formato *chave:valor* e são imbricados, ou seja, o *valor* de uma *chave* pode ser outro par *chave:valor*. Assim, um valor pode se agrupar como uma lista de valores (conhecido como *array*) ou outro documento JSON (conhecido como objeto). Finalmente, os valores podem ser atômicos, assim possuem tipos associados: inteiro, sequência de caracteres, real ou data [Bray 2014].

O formato JSON é livre, podendo o autor criar seus conjuntos chave-valor da maneira que julga necessário respeitando a imbricação, ou seja, todo o valor não atômico deve ser aberto e fechado para respeitar o aninhamento. Opcionalmente, um documento JSON pode estar associado a um esquema. Isso pode ser necessário para facilitar as consultas e armazenamentos de documentos JSON, principalmente em SGBD NoSQL. A Figura 1 apresenta um extrato de um documento JSON (A) e um possível esquema que este documento respeita (B). Perceba que existem chaves atômicas (*e.g.*, *movie*, *year* e *budget*) e uma chave do tipo *array* que armazena objetos, *i.e.*, *sequels*. Assim, uma abor-



```
{ "movie": "Nightmare on Elm Street",
  "director": "Wes Craven",
  "year" : 1984,
  "budget":1800000.00,
  "sequels": [{"name": "Freddy's Revenge",
      "year":1985,
      "director": "Jack Sholder",
      "budget": 3000000.00},
     {"name": "Dream Warriors",
      "year":1987,
      "director": "Chuck Russel",
      "budget": 4500000.00},
     {"name": "The Dream Master",
      "year":1988,
      "director": "Reni Harlin"},
      "budget": 6500000.00]
}
```

(A)

```
{"movie": {"type" : "string"},
 "director": {"type" : "string"},
 "year": {"type" : "integer"},
 "budget": {"type" : "real"},
 "sequels": {
    "type" : "array",
    "items": {
       "type": "object",
       "properties":{
          "name":{"type" : "string"},
          "year":{"type" : "integer"},
          "director":{"type":"string"},
          "budget":{"type" : "real"}
       }
    }
  }
}
```

(B)

Figura 1. Um documento JSON D e um possível esquema que D respeita.

dagem de extração de esquemas de documentos JSON deve ler o documento de entrada e encontrar um esquema que descreva adequadamente a entrada, ou seja, o esquema extraído deve ser flexível o suficiente para validar o documento de entrada e não permissivo ao ponto de validar qualquer documento.

3. Trabalhos Analisados

A seleção dos trabalhos apresentados nesta seção foi realizada por meio da plataforma *Google Scholar* com a seguinte *string* de busca *JSON AND ("schema extraction" OR "schema inference" OR "schema discovery")* com publicações entre 2017 e 2022, retornando 494 trabalhos. Sete artigos foram selecionados baseado nos seguintes critérios: (*i*) trabalhos mais relevantes com o tema da análise (leitura dos resumos), (*ii*) qualidade do veículo de publicação (*h-index*), e (*iii*) número de citações. Dos sete artigos selecionados, três trabalhos de 2021, dois de 2022, um de 2019 e um de 2018.

O primeiro trabalho apresenta a ferramenta *JSON Schema Discovery* [Frozza et al. 2018]. Esta ferramenta gera um esquema JSON para coleções de documentos JSON ou *Extended* JSON do MongoDB. O processo de extração é dividido em quatro etapas: geração do esquema bruto dos documentos, agrupamento dos esquemas brutos, unificação dos esquemas brutos e geração do *JSON Schema*. Inicialmente, os valores atômicos são substituídos por seus domínios, em seguida, chaves iguais são agregadas e ordenadas. Baseado nos esquemas brutos gerados, aqueles duplicados são removidos (etapas 1 e 2).

Na terceira etapa, uma estrutura de dados baseada em árvore é definida. Ela armazena informações hierárquicas dos esquemas brutos, chamada de *Raw Schema Unified Structure* (RSUS). Essa é manipulada por um processo baseado em um *Model Driven Engineering* (MDE). As informações são as chaves (objetos ou *arrays*) com seus domínios. A hierarquia construída permite armazenar os caminhos da raiz até as chaves (atributos) do esquema e seus domínios. Na quarta etapa, a abordagem transforma a RSUS em um esquema JSON.

Diferentemente de [Frozza et al. 2018], [Abdelhedi et al. 2021] propõem a ferramenta *ToNoSQLSchema* que é baseada em regras de transformação para a extração do

esquema. As coleções estão armazenadas no MongoDB e o processo é baseado na *Model Driven Architecture* (MDA).

A primeira regra cria uma coleção com o esquema (*DB_Schema*) para cada coleção de entrada e, em seguida, os *DB_Schemas* são unidos no que os autores chamam de *CollectionSchema*. A próxima regra codifica os valores atômicos das chaves nos *DB_Schemas*. A Regra 4 é aplicada em estruturas complexas, caso dentro destas estruturas existam valores atômicos, a Regra 3 é novamente aplicada. Por fim, as Regras 5 e 6 são aplicadas para criar as estruturas das chaves mono-valoradas e multivaloradas, respectivamente. Após a aplicação das seis regras, o esquema correspondente às coleções de entrada é gerado.

Já a ferramenta *ClustVariants* [Thilagam et al. 2022] tem como objetivo identificar as variações nos esquemas extraídos de uma coleção JSON do MongoDB. No primeiro passo, a coleção é transformada em sub-esquemas em um formato próprio. Por exemplo, o documento {*"pessoa" : [{ "nome":"Michael"}], "idade": 28*} é transformado em {*pessoa#nome.str, idade.int* }, sendo que # indica aninhamentos com *arrays*. Uma estrutura hierárquica é construída (similar aquela de [Frozza et al. 2018] e documentos que possuam caminhos similares são agrupados e analisados por meio de clusterização (utilizando o método *Formal Concept Analysis* - FCA). O uso do FCA permite identificar as variações dos esquemas similares permitindo lidar com as estruturas heterogêneas de grande volume de dados no formato JSON.

Ultimamente, o aprendizado de máquina (AM) vem sendo utilizado em várias aplicações, Namba e Mior [Namba 2021] aplicaram AM para distinguir os metadados de uma coleção JSON. Argumentam que a maioria dos extratores não considera que parte de uma chave representa dados, mas é extraída como metadado, ou seja, algumas chaves representam, na realidade, valores. Chaves desta categoria são chamadas de dinâmicas em contrapartida das estáticas (que não apresentam tal característica).

A abordagem extrai atributos numéricos a partir de coleções JSON criando um conjunto de dados correspondente. São propostos e extraídos seis atributos: (*i*) Domínio das Características Intrínsecas, (*ii*) Domínio da Tendência Central, (*iii*) Domínio da Dispersão Estatística, (*iv*) Domínio do Formato, (*v*) Domínio da Semelhança Semântica e Contextual, e (*vi*) Domínio da Semelhança Estrutural. Tais atributos representam numericamente algumas características descobertas pelos autores que podem identificar uma chave como estática ou dinâmica. Domínio das Características Intrínsecas, por exemplo, apresenta dois atributos intrínsecos, a porcentagem e o nível de alinhamento das chaves. A porcentagem refere-se à quantidade de vezes que, em todos os documentos, uma chave aparece, relativa ao número total. O nível de alinhamento, como o nome indica, é a profundidade de uma chave em um documento JSON.

Todos os atributos extraídos são armazenados para gerar o conjunto de dados alvo para a aplicação de algoritmos de aprendizado de máquina. Segundo os autores, o conjunto de dados resultante é desbalanceado, possuindo mais exemplos de atributos estáticos do que dinâmicos. Para solucionar esse problema, exemplos dinâmicos são gerados sinteticamente. Finalmente, o conjunto de dados é normalizado subtraindo de cada valor a sua média e dividindo pelo desvio padrão. Assim, todos os valores ficarão na mesma magnitude, sendo que a média e o desvio padrão resultantes ficarão em 0 e 1, respectivamente.

O conjunto de dados gerado é utilizado em três algoritmos de classificação: regressão logística, *random forest*, e *support vector machines* (SVM). *Random forest* foi o melhor classificador para identificar chaves dinâmicas e estáticas a partir de uma coleção de documentos JSON de entrada.

Klessinger et al. [Klessinger et al. 2022] apresentaram uma pesquisa voltada a detecção de *tagged unions* com base em coleções JSON, dependências entre o valor de um atributo e o esquema interno similar subsequentes. *Tagged unions* é uma maneira de definir um tipo de dado que tem estruturas diferentes com diferentes propósitos.

A abordagem não tem foco na extração do esquema e sim na detecção de *tagged unions* em um esquema existente. Assim, dado um esquema pré-extraído, todos os valores JSON são colocados em uma árvore para posterior determinação dos caminhos da raiz até um determinado objeto. Os caminhos são codificados em *JSONPath* e, a partir dele, uma codificação entre as relações dos caminhos é criada (*relational encoding*). A partir do *relational encoding*, dependências funcionais entre as chaves. Como resultado, dependências funcionais condicionais são criadas sendo utilizadas como base para declarar as *tagged unions*.

O trabalho de Baazizi *et al.* [Baazizi et al. 2019a] é similar aos trabalhos de [Frozza et al. 2018, Abdelhedi et al. 2021], porém utilizando um algoritmo de inferência paramétrica e a linguagem funcional *Scala* para extração. *Scala* permite que coleções JSON volumosas possam ser utilizadas como entrada para a extração. Basicamente, a abordagem utiliza o caminho estrutural do esquema extraído desconsiderando objetos com chaves repetidas para obter o esquema final.

Há duas fases: inferência do tipo individual e *type reduction*. A primeira fase é implementada com a operação *Map*, nela são inferidos os tipos para cada um dos atributos e, a partir da análise de tipos distintos, é produzido um conjunto desses que, na fase seguinte, serão combinados.

Na segunda, utilizam a operação *Reduce*, iterativamente agrupando os tipos inferidos na fase anterior. Essa operação possui as seguintes propriedades: inclusão, associatividade e comutatividade. Seguindo uma relação de equivalência, os tipos são comparados e fundidos caso sejam suficientemente parecidos e caso contrário, são unidos. Por exemplo, se chave *idade* tem dois tipos definidos (*e.g.*, numérico e string) há a união deles (idade: (STR + NUM)). Ao contrário, se os valores de *idade* fossem apenas numéricos, ocorreria a fusão (idade: NUM).

São propostos dois tipos de equivalência: *kind equivalence* quando as chaves possuem tipos iguais (*e.g.*, atômicos, arrays ou objetos); e *label equivalence* que utilizando condições da *kind equivalence*, fusiona objetos com o mesmo conjunto de chaves. Utilizando as estratégias *kind equivalence* e *label equivalence*, o esquema final das coleções de entrada é gerado.

Finalmente, a ferramenta *Jxplain* [Spoth et al. 2021] procura reduzir, utilizando heurísticas, três ambiguidades na inferência de esquemas JSON: identificação de *arrays* que contém uma coleção, objetos que representam tuplas e coleções de entidade múltiplas.

Para identificar as duas primeiras ambiguidades, como heurística base, foi observado que chaves variam mais em coleções e os tipos de alinhamento são mais consistentes.

Assim, são calculados *Key-Space Entropy* e Entropia dos Tipos (*Type Entropy*).

Key-Space Entropy é a variação do número de chaves em um objeto ou a distribuição do tamanho dos *arrays*. Já, a Entropia dos Tipos descreve a variação dos tipos de um atributo. Assim, *arrays* e objetos são tuplas quando há tipos diferentes em dois valores alinhados ou a *Key-Space Entropy* está abaixo do valor 1.

Com o intuito de identificar coleções de entidade múltiplas, foi utilizada uma técnica de *bi-clustering* (agrupar registros com co-ocorrência e co-aparição). Essa técnica, chamada de *Bimax*, agrupa subconjuntos/superconjuntos utilizando um algoritmo guloso. Assim, o maior conjunto de chaves é gulosamente selecionado e os outros registros são particionados em: (i) subconjuntos estritos; (ii) conjuntos de chaves sobrepostos; e (iii) conjuntos de chaves disjuntos.

Aplicando o algoritmo, subconjuntos são classificados mais perto, o segundo grupo, um pouco mais longe e o último, seguindo o padrão, ainda mais distante. Uma implementação ingênua desenvolvida cria entidades utilizando o primeiro grupo construído, sendo cada conjunto retornado como um *cluster*.

No entanto, esse algoritmo é limitado pelos atributos opcionais. Para isso, utilizam uma heurística gulosa que, como em um grafo não direcionado, há um nodo e uma aresta, respectivamente para cada entidade encontrada na implementação ingênua e para nodos que possuem um atributo igual. Como atributos opcionais representam regiões com várias interconexões, a heurística agrupa regiões densas identificando ciclos, que são agrupados em um nodo, levando em consideração o algoritmo *Bimax*, ocorrendo na ordem inversa de descoberta.

Nesta seção foram apresentadas as diferentes abordagens de extração de esquema elencadas. Cada uma das abordagens possui uma técnica única de extração, e percebe-se que há uma boa variedade de métodos sendo empregados. Na próxima seção é realizada uma comparação das abordagens e feita uma discussão sobre os principais pontos que as diferem das encontradas pelo trabalho de Imhof *et. al* [Imhof et al. 2017].

4. Discussão

Esta seção discute os trabalhos analisados fazendo uma comparação com os achados de [Imhof et al. 2017]. A Tabela 1 sumariza a comparação proposta neste trabalho. A seguir, são apresentadas as colunas que identificam cada trabalho quanto às seguintes características: origem do documento JSON, objetivo, abordagem, modelo intermediário e modelo de saída. Perceba que as colunas foram selecionadas conforme o trabalho comparado. Itens não encontrados nos trabalhos foram marcados com *NE*.

As origens dos documentos JSON em todos os trabalhos são identificadas da mesma forma, coleções de documentos JSON. Um trabalho também apresenta a entrada de um próprio esquema JSON, focando na detecção de *tagged unions* e deixando a extração para ferramentas de terceiros [Klessinger et al. 2022].

Três trabalhos intencionam a geração de um esquema JSON [Frozza et al. 2018, Baazizi et al. 2019a, Spoth et al. 2021]. Ainda, dois deles ampliam esse objetivo ao lidar com *datasets* volumosos e reduzir ambiguidades [Baazizi et al. 2019a, Spoth et al. 2021]. A ferramenta *ToNoSQLSchema* procura extrair o esquema NoSQL [Abdelhedi et al. 2021], enquanto outros enfatizam a descoberta de variações de esquema,

a distinção entre dados e metadados e a detecção de *tagged unions* [Thilagam et al. 2022, Namba 2021, Klessinger et al. 2022].

A escolha de abordagem é diversa, justificada pelos objetivos apontados. Heurísticas são utilizadas em dois trabalhos [Klessinger et al. 2022, Spoth et al. 2021]. As seguintes, todavia, são únicas: hierarquia e *Model Driven Engineering* (MDE), regras, clusterização e FCA, aprendizado de máquina (AM) e por fim, Scala e um algoritmo de inferência paramétrica [Frozza et al. 2018, Abdelhedi et al. 2021, Thilagam et al. 2022, Namba 2021, Baazizi et al. 2019a].

O *JSON Schema Discovery* define uma estrutura de dados baseada em árvore, a RSUS, como modelo intermediário [Frozza et al. 2018]. Regras são utilizadas em três pesquisas [Namba 2021, Baazizi et al. 2019a, Spoth et al. 2021], sendo em uma deles de inferência e em outra de AM, respectivamente [Baazizi et al. 2019a, Namba 2021].

O esquema JSON é inferido e representa a saída em quatro trabalhos [Frozza et al. 2018, Klessinger et al. 2022, Baazizi et al. 2019a, Spoth et al. 2021]. Além disso, é extraído o esquema NoSQL em uma das pesquisas, bem como as variações de esquema e a distinção entre dados e metadados [Abdelhedi et al. 2021, Thilagam et al. 2022, Namba 2021].

Ref	**Origem**	**Objetivo**	**Abordagem**	**Modelo Interm.**	**Modelo de Saída**
[Frozza et al. 2018]	MongoDB dataset	Geração de um esquema JSON	Sumarização Hierárquica e MDE	RSUS	JSON Schema
[Abdelhedi et al. 2021]	MongoDB dataset	Extrair o NoSQL Schema automaticamente após escanear o MongoDB	Baseado em regras	NE	NoSQL Schema
[Thilagam et al. 2022]	MongoDB dataset	Descobrir a variação de esquema	Clusterização e FCA	NE	Variações de Esquema
[Namba 2021]	JSON datasets	Distinguir dados de metadados	Aprendizado de Máquina	Regra de AM	Distinção entre dados e metadados
[Klessinger et al. 2022]	JSON datasets e JSON schema	Detecção de tagged unions	Heurística	NE	JSON Schema
[Baazizi et al. 2019a]	JSON datasets	Extrair o esquema JSON de datasets volumosos	Scala e algoritmo de inferência paramétrica	Regras de inferência	JSON Schema
[Spoth et al. 2021]	JSON datasets	Reduzir ambiguidades na inferência de esquema	Heurística	Regras	JSON Schema

Tabela 1. Tabela comparativa dos trabalhos analisados.

A Figura 2 apresenta a tabela proposta em [Imhof et al. 2017] para comparar as abordagens discutidas naquele trabalho. Foram 8 trabalhos analisados de 2012 a 2016. Por razão de espaço, as referências foram excluídas, mas os leitores podem encontrá-las em [Imhof et al. 2017]. Ambas as tabelas possuem colunas similares, porém a tabela aqui

proposta suprimiu as colunas **Unif.** e **Etapas do processo**, pois ambas as características são discutidas no corpo do texto durante a apresentação da abordagem.

Em relação à origem, as fontes continuam as mesmas, coleções de documentos JSON. No entanto, alguns dos novos trabalhos analisados procuram descobrir informações nos esquemas que não foram apresentadas no estudo anterior. Por exemplo, a distinção entre dados e metadados, a redução de ambiguidades e a descoberta de *tagged unions* e variações de esquema [Namba 2021, Spoth et al. 2021, Klessinger et al. 2022, Thilagam et al. 2022].

Foi possível identificar o uso de novas abordagens em relação ao estudo anterior: regras, aprendizado de máquina, heurísticas, Scala, clusterização e FCA [Abdelhedi et al. 2021, Namba 2021, Klessinger et al. 2022, Spoth et al. 2021, Baazizi et al. 2019a, Thilagam et al. 2022]. Além disso, as regras empregadas em três trabalhos estudados [Namba 2021, Baazizi et al. 2019a, Spoth et al. 2021] como modelo intermediário, sendo uma baseada em AM e dois utilizando inferência [Namba 2021, Baazizi et al. 2019a], são novidades identificadas no presente estudo.

É relevante mencionar que enquanto a representação do esquema de alguma forma como modelo de saída continua nos trabalhos [Frozza et al. 2018, Abdelhedi et al. 2021, Klessinger et al. 2022, Baazizi et al. 2019a, Spoth et al. 2021], houve uma evolução nesse quesito ao encontrar variações de esquema e a distinção entre dados e metadados [Thilagam et al. 2022, Namba 2021].

Finalmente, é possível afirmar que a área de extração de esquemas é ainda bastante ativa e que o uso de JSON como formato de entrada está presente em vários estudos. Com novas formas de repositório de dados mais flexíveis (*e.g.*, *Data Lakes*), a extração de esquemas se torna cada mais imprescindível para estruturar os dados semi-estruturados e oferecer formas mais eficientes de consulta e armazenamento.

5. Conclusão

Este mapeamento sistemático apresentou trabalhos atuais (de 2018 a 2022) na área de extração de esquemas de documentos JSON. O objetivo foi identificar a evolução das abordagens comparado-as com o mapeamento sistemático realizado em [Imhof et al. 2017]. Esta identificação procurava verificar se houve ou não evolução nas técnicas de extração de esquemas.

Como esperado, as fontes de extração continuaram as mesmas, porém no quesito dos modelos intermediários, alguns dos trabalhos atuais utilizam regras de inferência como base. Isso se dá, pois as regras de inferência usam gramáticas para descrever esquemas, assemelhando-se às abordagens de compiladores em relação à estrutura das linguagens de programação. Também, pode ser ressaltado que um dos trabalhos se apoia em aprendizado de máquina, que é uma técnica aplicada amplamente em outras áreas de computação.

Este mapeamento mostrou que houve novidades na extração de esquema, principalmente nas abordagens utilizadas e nos modelos intermediários. Como direções futuras pode-se citar: (i) fazer um estudo mais aprofundado na utilização de aprendizado de máquina, (ii) comparar, em termos de métricas, o inter desempenho dos trabalhos selecionados, e (iii) verificar se o padrão de esquema (JSON Schema) resultante nos trabalhos

Origem	Objetivo	Abordagem	Modelo Interm.	Modelo de Saída	Unif.	Etapas do processo
APIs de redes sociais	Integrar perfis de usuário em redes sociais	Transformação de modelos	JSON *Schema*	Modelo de classes (ECORE)	SIM	a) Extração de dados b) Extração de esquemas c) Transformação d) Integração
APIs de serviços *web*	Criar visão de domínio para os serviços da API	Transformação de modelos	Meta-modelo JSON (ECORE)	Modelo de domínio da Aplicação (ECORE)	SIM	a) Pré-descoberta de documentos JSON b) Extração de esquema do serviço c) Criação do esquema de domínio
HBase	Integrar bancos de dados	Organização em ontologia	*HBase*	OWL	SIM	a) Criação de uma tabela de consulta sobre os documentos JSON com *MapReduce* b) Escolha do esquema mais apto via algoritmo genético c) Mapeamento do esquema para uma ontologia local OWL d) Combinação de ontologias para formar uma ontologia global (integração)
Dataset no *MongoDB*	Criar ferramentas para manipular e gerenciar esquemas	Organização hierárquica (grafo)	*Structure Identification Graph* (SG)	JSON *Schema*	SIM	a) Seleção de documentos JSON b) Extração estrutura dos documentos c) Construção do grafo SG d) Geração do esquema JSON
MongoDB, *CouchDB* e *HBase*	Criar utilitários para BD NoSQL	Transformação de modelos	Meta-modelo JSON	Meta-modelo de esquema NoSQL	NÃO	a) Extração de objetos JSON b) Transformação para esquemas JSON c) Transformação para esquema NoSQL
Datasets JSON	Consultar esquemas e integrar dados	Organização Hierárquica (árvore)	*eSiBu-Tree*	*eSiBu-Tree*	SIM*	a) Seleção dos documentos JSON b) Criação registro na *eSiBu-Tree* c) Visualização do esquema unificado (*skeleton*)
Dataset JSON ou CSV	Migrar para BD Relacional	Organização Hierárquica (grafo)	Grafo dirigido	Modelo relacional	SIM	a) Criação de uma árvore de atributos e mineração de dependências funcionais b) Identificação de entidades sobrepostas c) Geração do esquema físico relacional
Oracle JSON *column*	Consultar coleções de documentos JSON	Organização hierárquica (JSON DOM)	--x--	JSON *DataGuide*	SIM	a) Derivação dos caminhos (*paths*) estruturais hierárquicos de documentos JSON b) Armazenamento de informações estatísticas dos JSON *paths*

Figura 2. Tabela de comparação proposta por [Imhof et al. 2017]

.

segue o que é preconizado pela especificação oficial.

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. (2019a). Parametric schema inference for massive json datasets. *The VLDB Journal*, 28:497–521.

Baazizi, M.-A., Colazzo, D., Ghelli, G., and Sartiani, C. (2019b). Schemas and types for json data: From theory to practice. In *SIGMOD/PODS 2019*.

Bouchou, B. and Duarte, D. (2007). Assisting XML schema evolution that preserve validity. In *Simpósio Brasileiro de Banco de Dados - SBBD*, pages 270–284.

Bray, T. (2014). The javascript object notation (JSON) data interchange format. Technical report, Datatracker.

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.

Imhof, R., Frozza, A. A., and dos Santos Mello, R. (2017). Um survey sobre extração de esquemas de documentos json. In *Anais da XIII Escola Regional de Banco de Dados*. SBC.

Klessinger, S., Klettke, M., Störl, U., and Scherzinger, S. (2022). Extracting json schemas with tagged unions. *coordinates*, 30:10.

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

Spoth, W., Kennedy, O., Lu, Y., Hammerschmidt, B., and Liu, Z. H. (2021). Reducing ambiguity in json schema discovery. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1732–1744.

Thilagam, P. S. et al. (2022). Clustvariants: An approach for schema variants extraction from json document collections. In *2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, pages 515–520. IEEE.

Zouari, F., Kabachi, N., Boukadi, K., and Ghedira Guegan, C. (2021). Data management in the data lake: A systematic mapping. In *Proceedings of the 25th International Database Engineering & Applications Symposium*, pages 280–284.