JSONGlue: A hybrid matcher for JSON schema matching

Vitor Marini Blaselbauer[1], **João Marcelo Borovina Josko**[1]

[1]Center of Mathematics, Computing and Cognition – Federal University of ABC (UFABC)
Av. dos Estados, 5001 – Santo Andre – SP – Brazil

vitor.blaselbauer@aluno.ufabc.edu.br, marcelo.josko@ufabc.edu.br

Abstract. *The JSON-based databases are the dominant cloud-oriented approach to handle structured-variable data. Its schemaless nature makes writing operations quick at the expense of integrating data with heterogeneous schemas. Schema matching literature collects various contributions to identify similarities in the XML documents, but very few for JSON. This work introduces a hybrid supervised matcher (named JSONGlue) that executes linguistic, semantic, and instance-based methods in parallel to match multiple heterogeneous JSON schemas. It also reports JSONGlue's first results and presents its next evolution steps.*

1. Introduction

The JSON-based databases are the dominant cloud-oriented approach to handle structured-variable data in Big Data or Internet contexts. Besides their scalable feature, their no rigid schema before writing into the database made application development more straightforward. Conversely, since the database ignores the structure of their persisted data, applications may store semantically equivalent documents using heterogeneous schemas. This heterogeneity increases the complexity of integrated data access by analytical procedures and makes database evolution hard.

Schema matching denotes the methods of identifying a possible correspondence between elements of different schemas that contain semantically related data [Gal 2011]. Its literature presents important methods for various data representations, including structured and XML. Several studies discussed approaches that focus on schema characteristics (e.g., attributes names) or instance-level similarity (e.g., data distribution) to determine the schemas correspondence [Gal 2011]. Other works adopt various methods (hybrid approach) to identify similarities, whereas some studies add the semantic distinction capacity of human beings into the matching process [Gal 2011]. However, few researchers have addressed schema matching for the JSON data format. Indeed, we have found two works [Padilha 2020, Waghray 2020] whose purpose or design differ from the ones used by the present work.

This work introduces a hybrid matcher (named JSONGlue) that uses linguistic, semantic, and data distribution approaches to match several JSON schemas of a given dataset. Our supervised matcher maps all matches and corresponding similarity measures to a graph structure to enable future human supervision and analysis mediated by visualizations.

This paper is structured as follows: In Section 2, we briefly discuss the main differences between JSON and XML. Next, we characterize the architecture and components of our matcher in Section 3 and report its preliminary results in Section 4. Finally, we review related works in Section 5 and present conclusions and future works in Section 6.

2. Characterizing JSON and XML

JSON (JavaScript Object Notation) and XML (Extensible Markup Language) are both popular notations to store and transfer data. Although they share several aspects, the specificity of JSON makes its matching process slightly different from XML.

JSON has an array data type that might include any of the JSON value types (e.g., string, numeric), embedded attributes, or even another array (nested array). Moreover, JSON allows the same element label to occur in the same schema as it does not have a namespaces feature. This situation may lead to label collision. Finally, the unordered and arbitrary levels of nesting key-value pairs of JSON documents contrast with the essential role of the order of XML elements.

3. JSONGlue Characteristics

Figure 1 exhibits the JSONGlue architecture style and its modules' communication flow. These modules code uses Python language due to its extensive support libraries that facilitate string operations, manipulation of massive amounts of data, parallel processing, among other possibilities. Except for the normalization module that builds the graph, the remaining modules run parallelly using all core available. For each distinct pair of JSON schemas, our matcher spawns a process to run the three schema matching modules.

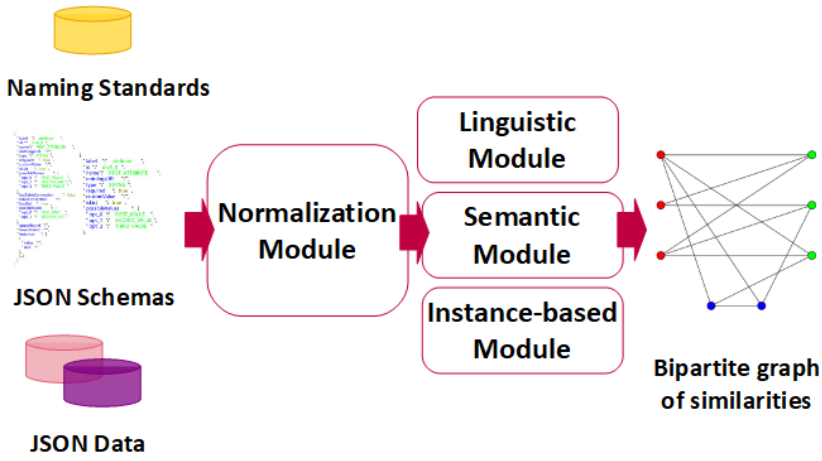


Figure 1. JSONGlue architecture style and components (Source: The authors)

The *normalization module* has three steps executed in sequence. The first is in charge of removing irrelevant characters, including numbers, extra spaces between tokens, special symbols, punctuation symbols, and stop words based on WordNet [Miller 1998]. In turn, the second transforms elements' text to lowercase and convert abbreviations to the corresponding business name. This last procedure occurs if there is a file that represents this organization's naming standard convention. The third step builds a disconnected graph $\mathcal{G}$ composed of subgraphs $\mathcal{S}_i$, $\mathcal{S}_i = \mathcal{S}_j$ iff $i = j$, that correspond to each input scheme $\mathcal{SC}_i$, $i \geq 2$. Moreover, this step breaks arrays of embedded attributes into individual elements for handling for the next steps.

The *linguistic module* is in charge of measuring the string similarity between all the JSON elements of all previous schemes loaded. In other words, this module applies a similarity function $ling : (a, b) \rightarrow [0, 1]$ (for each pair of nodes (a_k, b_m), $k = |\mathcal{S}_i|$,

$a_k \in \mathcal{S}_i$, $m = |\mathcal{S}_j|$, $b_m \in \mathcal{S}_j$, $i \neq j$. Afterward, it creates an edge for each pair (a_k, b_m) whose linguistic property contains this function's result. This work uses the Jaro-Winkler algorithm as it is best suited for comparing short strings [Peng et al. 2012], though it produces favorable ratings when strings beginnings are the same.

The *semantic module* uses a lexical function to measure the similarity between all pairs of nodes. This function uses the WordNet monolingual database because of its widespread adoption for linguistics processing tasks [Miller 1998]. The current JSONGlue version assumes no semantic available about data (e.g., a partial ontology or data dictionary). Hence, its lexical function $sema : (a, b) \rightarrow_{max} [0, 1]$ transverses all WordNet synset (set of synonyms denoting the same concept) until it reaches the one of maximum similarity for each pair of nodes (a_k, b_m). When comparing the JSON elements with a different number of words, this function reconciles their length using the ancestor (when available) or assume no similarity between head nouns. Subsequently, it returns the average between the individual similarity of each modifier and head noun. In the current version, our tool does not handle the JSON elements with three or more words. Analogous to the previous module, our system connects each pair of nodes and assign to its similarity property the lexical function's result. We use the Wu-Palmer measure in which similarity is inverse to the path distance between two concepts [Miller 1998].

Lastly, the instance-based module applies three functions to measure how similar data values are. The first $diffAVG : (a, b) \rightarrow R_{\geq 0}$ calculates the difference between each pair of nodes by considering the average length of their data values, while the second $diffSTD : (a, b) \rightarrow R_{\geq 0}$ does the same for the standard deviation. The third function $distHIST : (a, b) \rightarrow R_{\geq 0}$ measures the distance between two histograms H_a and H_b that describe the characters' frequency within the data values of a given pair of nodes. The calculation of this distance considers both the overlapping and non-overlapping parts of the histograms (D_1 algorithm by [Cha and Srihari 2002]).

4. Case Study with JsonGlue

4.1. Data settings

This case study used artificial and real data. For the former, our algorithm considered a reduced customer invoice business domain (outlined below) to generate three datasets of increasing sizes (Figure 4). Each dataset (D_1, D_2, D_3) has a distinct JSON schema (S_1, S_2, S_3, respectively). It randomly introduced schema variations in the number of attributes (15 on average) and their grouping (e.g., array, embedded), nomenclature (e.g., names with and without abbreviations), and position of attributes. Such an algorithm also randomly generated the values of the attributes with several differences (e.g., size, range of values).

Customer (CustomerID, Name, Address, Phone, Email)
Invoice (InvoiceID, CustomerID, InvoiceDate, DeliveryAddress, TotalAmount)
Invoice Item (InvoiceID, InvoideItemID, ProductName, ItemQuantity, ItemTotal)

For the latter, we extract two schemas (25 attributes on average) from a COVID-19 dataset[1] with the latest numbers from every US territory. Is it worth noting that we used the real dataset for validation purpose and the algorithm-generated datasets for validation and discussion of isolated methods results.

[1]The COVID Tracking Project

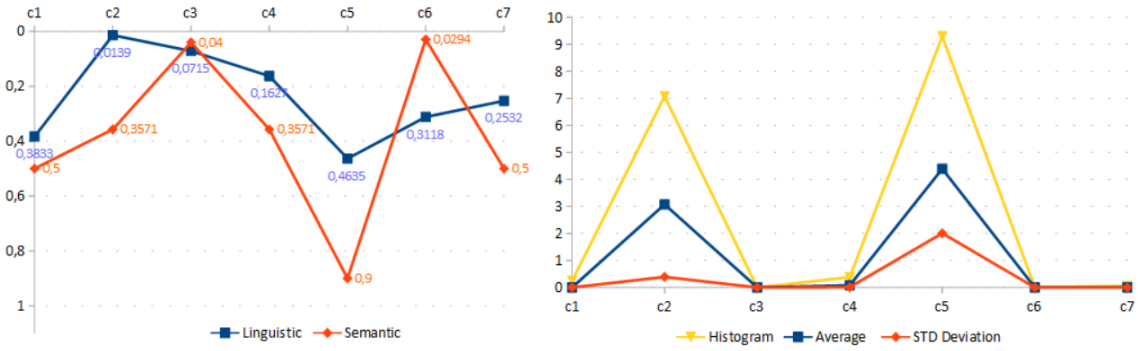
4.2. Methods Results and Validation

Table 1 shows some selected mapping cases of the JSONGlue outcome to highlight the characteristics of its methods. Each pair of lines represent a single case preceded by an identification code (c_n, $n \geq 1$) to make its reference easy. In turn, the columns present the schemas compared, its original elements names, and the names of the elements used on linguistic and semantic comparisons after normalization and ancestor identification. It is worth noting that linguistic and semantic measures (Figure 2a) use *zero* to denote full similarity and *one* otherwise.

Table 1. Selected Mapping Cases - Algorithm-based datasets

	Sch	Schema Element	Linguistic Element	Semantic Element
c_1 :	S_1 ·	[customer_phonenumber] ·	[customer phonenumber] ·	[customer phonenumber] ·
	S_2	[phone]	[phone]	[customer phone]
c_2 :	S_1 ·	[total_amount] ·	[total amount] ·	[total amount] ·
	S_2	[total_mount]	[total mount]	[total mount]
c_3 :	S_1 ·	[product_name] ·	[product name] ·	[product name] ·
	S_2	[product_title]	[product title]	[product title]
c_4 :	S_1 ·	[customer_name] ·	[customer name] ·	[customer name] ·
	S_3	[cus_name]	[cus name]	[cus name]
c_5 :	S_2 ·	[product_title] ·	[product title] ·	[product title] ·
	S_3	[inv_prod1]	[inv prod]	[inv prod]
c_6 :	S_1 ·	[invoice_id] ·	[invoice identifier] ·	[invoice identifier] ·
	S_2	[stmt_id]	[statement identifier]	[statement identifier]
c_7 :	S_1 ·	[quantity] ·	[quantity] ·	[invoice items quantity] ·
	S_2	[item_qty]	[item quantity]	[item quantity]

Semantic matching provides relevant results when schemas elements follow a naming standard or share correlated business terms (c_3, c_6 in Figure 2a). However, the outcome of this matching method is negatively affected by the absence of such concerns (c_4, c_5 in Figure 2a, respectively) or irregular nomenclature, including misspelling (c_2 in Figure 2a), excessive concatenation (c_1 in Figure 2a), or proper nouns.



(a) Linguistic and Semantic similarities **(b)** Instance-level distance

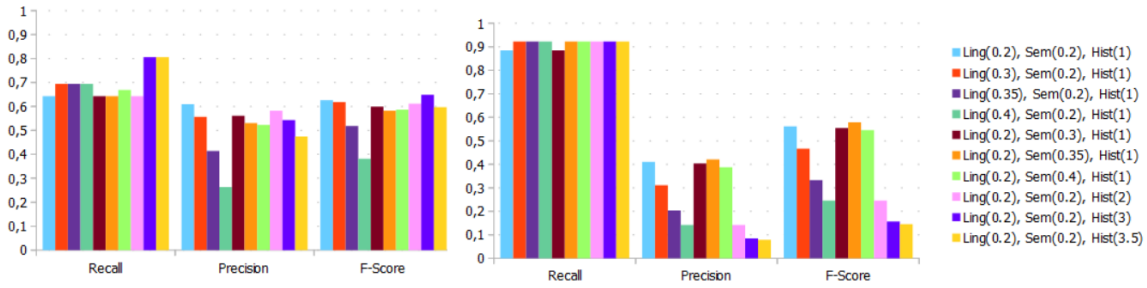
Figure 2. Similarity Results per Selected Case (Source: The authors)

Linguistic method outcomes can complement semantic similarity measurement to support matching analysis, although it also benefits from name homogeneity (c_3). For the

cases c_2 and c_4 (Figure 2a), the former provides a more robust indication of similarity than the latter. However, there are situations (c_6 in Figure 2a) that linguistic outcome is worse, or (c_1, c_7 in Figure 2a) both methods' measurements are uncertain for a proper decision.

Finally, instance-based methods can provide additional support for the previous analysis. In several cases, these methods reveal the little variance between the values of the elements compared, as observed in c_1, c_3, c_4, c_6, c_7 in Figure 2b. However, as the JSON elements may have a marked heterogeneity in terms of length, instance-based methods that use frequency of characters or string length distance decrease its certainty (c_2, c_5 in Figure 2b).

We validate JSONGlue matching results using a gold standard manually built for all datasets (Section 4.1). We also manually select some thresholds for each measure (the legend in Figure 3) as JSONGlue does not have an automatic threshold estimation yet.



(a) Algorithm-based $60K$ dataset (Average)

(b) Covid Dataset

Figure 3. Matching Results per Set of Thresholds (Source: The authors)

Figure 3a reveals that the instanced-based method (histogram distance) tend to provide better results for datasets with high or moderate heterogeneity attribute nomenclature. The increase in its recall ($\simeq 25\%$) and f-score ($\simeq 5\%$) measures somehow followed its threshold growth. In contrast, Figure 3b illustrates that semantic methods tend to offer a superior results when schemas attributes align with business terms. Its f-score and precision measures are close as the threshold increases. Figure 3b also shows that the precision measure of the instance-based methods decreases dramatically for datasets whose attributes share very close data ranges.

The parallel matching support reduced 27% (on average) the comparison time for the datasets from $1k$ to $60k$. Such reduction increases a little more ($\simeq 42\%$ on average) for the datasets bigger than $60K$, as illustrated by Figure 4. The overall matching time reduction using parallel processing support was 35% on average. We used an Intel $i5-8500$ computer with six cores, 16GB of RAM, and Ubuntu 20.04 of 64 bits to gather all matching results.

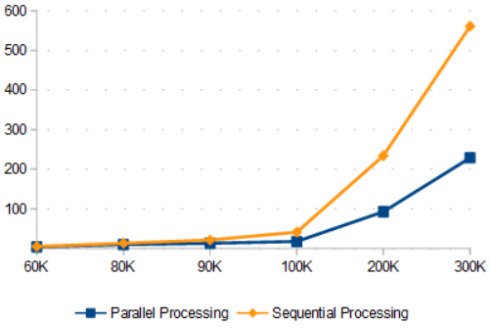


Figure 4. Matching time consumption (in min) per dataset size (Source: The authors)

5. Related Works

The literature regarding handling heterogeneous schema has a broad set of contributions that apply different perspectives to diverse data formats. Due to space restrictions, this work only discusses papers focusing on integrating heterogeneous JSON schemas.

Some studies apply one integration method to combine JSON documents on query time [Gallinucci et al. 2018] or to transform them into a relational representation [DiScala and Abadi 2016]. In another perspective, [Waghray 2020] shows that XML schema matching approaches do not readily support JSON schema matching.

The closest work to ours also combines linguistic, semantic, and instance-level methods to match heterogeneous schemas [Padilha 2020]. However, our work differs in the design approach and some of the methods adopted. For example, [Padilha 2020] does not use graphs to represent similarities among schemas, parallel matching, or histograms to compare data values. Conversely, our system does not calculate a summary similarity measure because we intend to integrate progressive matching visualization features.

6. Conclusion

This work reports the design approach and components of the JSONGlue system that matches JSON schemas using parallel processing. This matching process applies linguistic, semantic (single and compound noun), and instance-based methods to identify JSON elements' similarities represented in a graph form.

Nevertheless, JSONGlue neither consider semantic information nor uses human semantic distinction on the matching process. As future works, we intend to integrate human supervision and knowledge representations (e.g., expert rules, ontologies) to increase the JSONGlue matching performance.

References

Cha, S.-H. and Srihari, S. N. (2002). On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370.

DiScala, M. and Abadi, D. J. (2016). Automatic generation of normalized relational schemas from nested key-value data. In *Proceedings of the 2016 International Conference on Management of Data*, pages 295–310.

Gal, A. (2011). *Uncertain schema matching*. Morgan & Claypool Publishers, 1st edition.

Gallinucci, E., Golfarelli, M., and Rizzi, S. (2018). Variety-aware olap of document-oriented databases. In *DOLAP*.

Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

Padilha, R. J. (2020). Um processo para casamento de esquemas de documentos json baseado na estrutura e nas instâncias. Master's thesis, Federal University of Santa Maria, Brazil.

Peng, T., Li, L., and Kennedy, J. (2012). A comparison of techniques for name matching. *GSTF Journal on Computing (JoC)*, 2(1):55–61.

Waghray, K. (2020). Json schema matching: Empirical observations. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2887–2889.