JSON Schema Matching: Empirical Observations

Waghray, Kunal
kwaghray@wisc.edu, UW-Madison.

Quick Overview

- A major challenge in database integration is that of *schema matching* [10, 12], which seeks to determine the schema elements in different databases that correspond to the same real-world entity.
- There is interest in creating schema matching tools that can cope with a broad variety of schema definitions (e.g., relational, XML and OWL) [1, 9].
- Nowadays data is increasingly stored and exchanged in JSON, which is data-centric and often schema-less. To our knowledge, there is **no well-defined system for matching the schema of JSON files.**
- My contributions are: (1) create a novel (baseline) approach for matching JSON files using existing tools and methods, (2) evaluate the approach empirically, (3) formalize a few challenges specific to JSON schema matching. Experiments show that current tools are inadequate for JSON integration and work is needed to formally understand the underlying challenges and to create systems that natively support JSON

Schema Matching: Challenges

Missing, Misleading Info

Schema		Same Attribute?	Same Element?
Movies.year	~ Items.year	Yes	Yes
Movies.title	~ Items.name	No	Yes
Locations.name	≠ Items.name	Yes	No

Consider a matching system that looks at whether attributes are equivalent to determine if those attributes have the same real-world meaning. In the first example, this method is helpful, in the second it is not useful, and in the third it is misleading!

JSON Schema Matching: Additional Challenges

Schema	No Schema
{ “Director” : “Christopher Nolan”, “Comments”: [{ “ID” : 907, “Content” : “Love it” }, { “ID” : “909”, “Content” : “Too scary” }]}	{ “Christopher Nolan” : [[907, “Love it”], [909, “Too scary”]] }

Challenges of matching JSON files includes,

i) **Different Nesting Structures**: Path to 907 is Comments -> ID in schema one, there is no clear path in the second one.

ii) **Data Entries Lacking Names and Data Types:** What the data represents is not defined in “No Schema” example and neither states the types of data

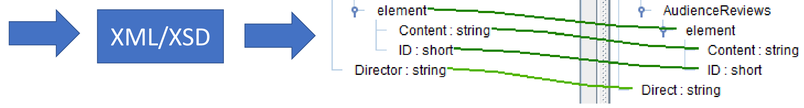
This is an issue as similar nesting structures and data types/names are valuable clues for identifying matches

Proposed (Baseline) Approach

```
{  "Director": "Nolan",
   "Comments": [{
     "ID": 907,
     "Content": "Love it"
}]}
```

```
{ "Direct" : "Mart",
  "Comments":{
      "AudienceReviews" : [{
          "ID": 907,
          "Content": “Inspired"
}]}}
```

JSON



COMA 3.0: Schema matching tool [3]

Empirical Results

(1) Schema Inference

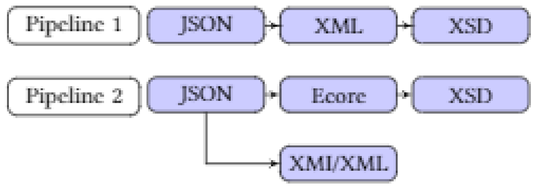


Figure 1: Evaluating Inference Tools

We create two pipelines to evaluate if a tool that aims to recover as much schema detail as possible improves matching. Both pipelines get similar results, so further work is needed to take advantage of schematic information in JSON files.

(2) Low Accuracy

F-Score Max: 0.43

The highest f-score is 0.43. It is obtained for the trial that uses pruning.

Thus, more work is needed to formally understand the underlying challenges and create systems that natively support JSON

(3) Pruning Improves

F-Score: 0.23 → 0.43

We create a testing scenario where we manually *prune* the files (i.e. remove parts that are not relevant for an example use case).

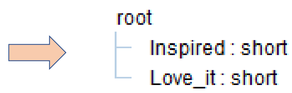
We see an improvement from 0.23 to 0.43!

Note: tests performed on three news sources of data with ~50-100 attributes each [6]

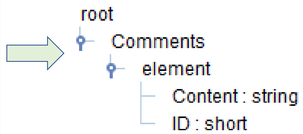
Formalized Challenges

(1) Key as Instance Data

```
{
   “Love it”: 907,
   “Inspired”: 908
}
```




```
{
  "Comments": [{
     "ID": 907,
     "Content": "Love it"
     }, {
     “ID”: 908,
     “Content” : “Inspired”
     }]
}
```



One of the issues we saw in testing data was that since converters assume JSON keys are part of the schema, it’s an issue when they contain instance data instead of attributes [2].

When instance data appears as a JSON key (as in the red example):

(i) It likely will not match with the corresponding attribute

(ii) It becomes unwieldy as there may be hundreds of instance data values for one attribute.

(2) Invalid Characters

Special characters, *null*, Integer overflow

JSON has fewer type restrictions in comparison to XML which can cause a variety of problems. For instance, the conversion of *null* results in undefined behavior as tools expect to be able to infer the type of each data element

References

1. Algergawy, Cheatham, Faria, et al.2018. Results of the Ontology Alignment Evaluation Initiative 2018. In13th International Work-shop on Ontology Matching co-located with the 17th ISWC (OM 2018),Vol. 2288. 76–116
2. Baazizi, Colazzo, Ghelli, et al. 2019. Schemas and types for json data: From theory to practice. In Proceedings of the 2019 International Conference on Management of Data. ACM, 2060–2063.
3. Hong-Hai Do and Erhard Rahm. 2002. COMA: a system for flexible combination of schema matching approaches. In Proceedings of the 28thinternational conference on Very Large Data Bases. VLDB Endowment,610–621.
4. AnHai Doan, Alon Halevy, and Zachary Ives. 2012.Principles of data integration. Elsevier.
6. Lihong He, Chao Han, Arjun Mukherjee, Zoran Obradovic, and Eduard Dragut. 2019. On the dynamics of user engagement in news comment media. Wiley Interdisciplinary Reviews: Data Mining Knowledge Dis-covery(11 2019).
9. Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and AlmaGómez-Rodríguez. 2015. Ontology matching: A literature review. Ex-pert Systems with Applications42, 2 (2015), 949–971.
10. M Tamer Özsu and Patrick Valduriez. 2011.Principles of distributed database systems. Springer Science & Business Media.
12. Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. the VLDB Journal10, 4 (2001), 334–350.

Acknowledgements

I would like to thank Dr. Eduard Dragut for his mentoring on this work and Dr. AnHai Doan for his valuable feedback
Research grant support was provided by NSF CNS-1757533 and BigData-1838145