JSONDiscoverer: Visualizing the schema lurking behind JSON documents

Javier Luis Cánovas Izquierdo[a], Jordi Cabot[b]

[a] jcanovasi@uoc.edu. *UOC. Barcelona, Spain*
[b] jordi.cabot@icrea.cat. *ICREA - UOC. Barcelona, Spain*

Abstract

The so-called API economy is pushing more and more companies to provide open Web APIs to access their data, typically using the JavaScript Object Notation (JSON) as interchange data format. While JSON has been designed to be easy to read and parse, their structure is implicit. This poses a serious problem when consuming and integrating Web APIs from different sources since it forces us to manually analyze each individual API in detail. This paper presents JSONDiscoverer, a tool that alleviates this problem by discovering (and visualizing) the implicit schema of JSON documents as well as possible composition links among JSON-based Web APIs.
Tool website: http://som-research.uoc.edu/tools/jsonDiscoverer

Keywords: JSON, schema discovery, concept matching, Web API

1. Introduction

The number of Web APIs is growing every day[1] opening the door to an unlimited number of new services built on top of such APIs. Most of those Web APIs use JavaScript Object Notation (JSON) as a data interchange format. JSON mimics the JavaScript syntax, thus becoming human readable and easily parseable. However, it is schemaless, i.e., JSON documents do not include an explicit definition of the structure of the JSON objects contained in them.

[1] The website Programmable web (http://www.programmableweb.com) alone indexes over 13,000 APIs.

This has several advantages but it is a serious problem when developing Web services that need to consume and exchange information among a set of APIs since developers need to figure out the structure of the JSON data provided by each one and the possible relationships between them. JSONDiscoverer pretends to liberate developers from performing these tasks by inferring and visualizing the implicit schema of JSON data as well as the possible composition links among JSON-based Web APIs. The tool has been made available as a web application. Since its release, JSONDiscoverer has been used to parse on average 375 JSON documents each month.

2. Problem and Background

The first thing needed to reuse/combine Web APIs is a good understanding of the data model behind them: what the data is about, what attributes each data object has, how they are related and so on.

JSON being schemaless combined with the fact that the few languages to specify Web APIs are still under development (e.g., RAML[2] or Swagger[3], which allow specifying API services and their parameters but not the full API schema) or are not widely used (e.g., JSON Schema [1]) implies that developers must manually test Web APIs and try to deduce the data model lurking behind their services. Earlier research efforts (e.g., [2] and [3]) could be applied to analyze JSON documents, however, they are specially tailored to NoSQL databases and do not provide assistance to integrate Web APIs.

Figure 1 (grey boxes) illustrates the typical scenario when trying to integrate several JSON-based Web APIs. First, developers test each service provided and reverse engineer the implicit structure of the JSON data returned when calling them. Then, these individual service schemas need to be combined to build the full Web API schema (i.e., the global data model the API is giving access to). If two or more Web APIs need to be integrated, a last step is required aimed at identifying possible connections by searching for similar JSON elements that could be representing the same concept. Besides, the signature of each individual Web API service must be also considered to guarantee the accessibility of the target resources. Needless to say this is a time-consuming and error-prone task.

[2] `http://raml.org`
[3] `http://swagger.wordnik.com`

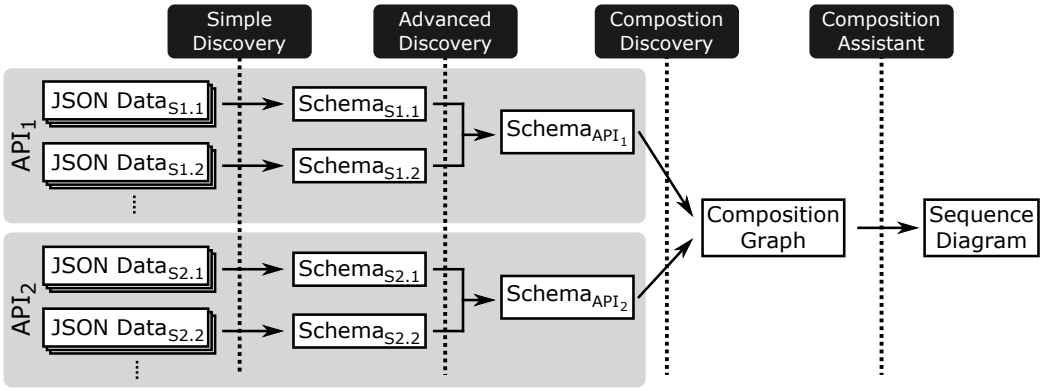

Figure 1: Discovery of the implicit structure from a set of JSON-based Web APIs. The main functionalities of our tool are represented with black-filled rounded boxes while input/output data is depicted as white-filled boxes.

3. Software Functionalities

JSONDiscoverer alleviates this situation by executing an automatic discovery process on the JSON data that uncovers the schema behind JSON-based Web APIs and suggests possible composition paths.

The tool provides three main functionalities, which can be used separately or chained (see black-filled boxes in Figure 1):

1. **Simple discovery**. It discovers the schema of a given set of JSON documents returned by a single service. The more the better, since some properties of the (implicit) model can only be deduced when having several examples.
2. **Advanced discovery**. It takes the output of the simple discoverer for each service of a given Web API to infer its global schema.
3. **Composer**. It takes a set of inferred Web API schemas and looks for composition links (i.e., common concepts or attributes). As a result, it generates a composition graph. The *Composition Assistant* will use this graph to help you find composition paths among the Web APIs.

JSONDiscoverer draws schema information as UML class diagrams, including concepts (i.e., classes) and their properties (i.e., attributes and associations linking the different concepts). Potential Web API compositions are represented by means of UML sequence diagrams showing the possible sequence of Web API calls.

More information about the discovery rules applied in the simple and advanced discoverers can be found in [4], while a description about the composition rules and sequence diagram generation is done in [5].

4. Implementation

JSONDiscoverer has been developed as a servlet-based web application including: (1) a backend developed in Java and providing the functionalities listed above; and (2) a front-end website implemented as an AngularJS web application. The website includes overlays to help newcomers to use the tool and a section explaining the inner workings for advanced users. Beyond this web frontend, JSONDiscoverer can also be executed from regular Java programs (see details on the tool website).

The parsing and management of JSON documents is performed by using the GSON library[4]. Model management relies on the Eclipse Modeling Framework (EMF)[5] while we use EMF2GV[6] for their rendering.

5. Example

Figure 2 shows the Simple Discoverer page with an example. Once the user provides the JSON document to analyze in the textbox (or uses the default example for testing purposes) the button *Discover Schema* launches the discovery process, which sends the JSON document to the backend. As a result, the backend returns the domain model (i.e., schema) and JSON data (as an instance of the generated model), both as EMF models and pictures. The EMF models can be downloaded by the user and directly imported into other modeling tools for further analysis and manipulation.

6. Conclusion

In this paper we have presented JSONDiscoverer, a tool aimed at promoting the integration and composition of JSON-based Web APIs. As further work we plan to enhance the concept-matching heuristics in the (advanced) discovery and composition process, and to provide code-generation facilities to realize the selected API's integrations.

[4] `https://github.com/google/gson`

[5] `https://eclipse.org/modeling/emf`

[6] `https://marketplace.eclipse.org/content/emf-graphviz-emf2gv`

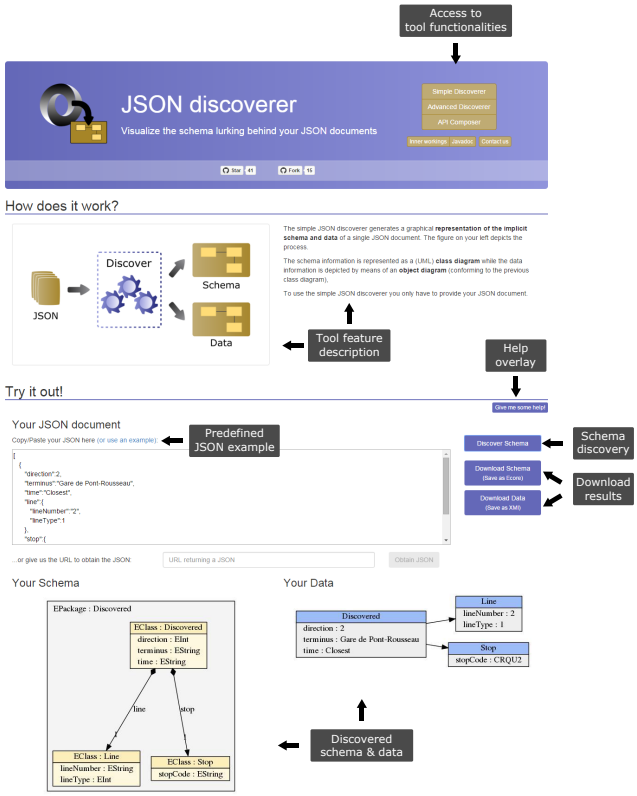

Figure 2: Webpage for the simple discovery.

References

[1] IETF, JSON Schema Specification. `http://json-schema.org/`.

[2] M. Klettke, U. Störl, S. Scherzinger, Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores, in: BTW conf., 2015, pp. 425–444.

[3] D. Sevilla, S. Feliciano, J. G. Molina, Inferring Versioned Schemas from NoSQL Databases and Its Applications, in: ER conf., 2015, pp. 467–480.

[4] J. L. Cánovas Izquierdo, J. Cabot, Discovering Implicit Schemas in JSON Data, in: ICWE conf., Vol. 7977, LNCS, 2013, pp. 68–83.

[5] J. L. Cánovas Izquierdo, J. Cabot, Composing JSON-based Web APIs, in: ICWE conf., Vol. 8541, LNCS, 2014, pp. 390–399.

Required Metadata

Current executable software version

Note that the tool has been developed as a servlet-based web application and it is provided as a WAR to be deployed in a servlet container.

Nr.	(executable) Software metadata description	Please fill in this column
S1	Current software version	v2.1.2
S2	Permanent link to executables of this version	`https://github.com/SOM-Research/jsonDiscoverer/tree/v2.1.2`
S3	Legal Software License	EPL
S4	Computing platform/Operating System	Java-compatible platform.
S5	Installation requirements & dependencies	Java Servlet container
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	`http://som-research.uoc.edu/tools/jsonDiscoverer/#/doc`
S7	Support email for questions	`jcanovasi@uoc.edu`

Table 1: Software metadata (optional)

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v2.1.2
C2	Permanent link to code/repository used of this code version	`https://github.com/SOM-Research/jsonDiscoverer/tree/v2.1.2`
C3	Legal Code License	EPL
C4	Code versioning system used	Git
C5	Software code languages, tools, and services used	Java, JavaScript, HTML, CSS, Eclipse, Tomcat
C6	Compilation requirements, operating environments	Client-side: HTML, CSS, JavaScript Server-side: Java, Servlets, Tomcat
C7	If available Link to developer documentation/manual	`http://som-research.uoc.edu/tools/jsonDiscoverer/#/doc`
C8	Support email for questions	`jcanovasi@uoc.edu`

Table 2: Code metadata (mandatory)