Abstract

In this paper, we present the JSON Stats Analyzer, a free-to-use open-source web-based JavaScript tool and module that provides JSON document analysis. We explain how the JSON Stats Analyzer works, its usage alongside the demonstration of eleven JSON documents from Tier 1, Tier 2 and Tier 3 from our proposed taxonomy that categorizes JSON documents according to their size, content, redundancy and nesting characteristics. For each JSON document, we provide its definition, characteristics and the document structure, alongside a visual representation of the JSON document structure and its summary statistics.

1 Motivation

To the best of our knowledge, there are gaps in the current literature that prevents a sound understanding of JSON [3] documents at a higher level that provides us with an insight of the instances themselves. As a consequence, research experiments that involve JSON documents do not follow a methodical approach for solving the input selection problem. To solve this problem, [8] (Table 2) introduces a formal tiered taxonomy consisting of 36 categories that form a common basis to class JSON documents based on their size, type of content, characteristics of their structure and redundancy criteria.

Given the universality of the JSON [3] data interchange format as the lingua-franca of the web [7], a growing amount of research is conducted taking JSON documents as input. For any experiment following a thorough scientific approach, the quality and representativity of the input data is a significant factor that determines the quality of the experiment results.

Software systems make use of JSON to model diverse and domain-specific data structures. Each of these data structures have characteristics that distinguish them from other data structures. For example, a data structure that models a person is fundamentally different from a data structure that models sensor data. The difference lies in the characteristics which use diverse data types such as text, numeric and boolean. Therefore, two instances of the same data structure may inherit the same or similar characteristics despite having different values.

Intuitively and on observation, we know that these characteristics exist, but in the existing literature we lack a common terminology to describe them in an unambiguous way. In an attempt to solve this ambiguity problem, our taxonomy [8] (Table 2) presents a formal vocabulary to describe, reason, discuss and reason about JSON documents in a high-level manner.
2 Taxonomy Definition

Our proposed taxonomy [8] aims to classify JSON documents into a limited and useful set of categories that is easy to reason about rather than exhaustively considering every possible aspect of a data structure. This taxonomy categorizes JSON documents according to their size, content, redundancy and nesting characteristics.

2.1 Naming Conventions

We present the naming conventions of the taxonomy in Table 1.

Table 1: The letter *T* stands for *Tier 1*, the letter *S* stands for *Tier 2* and the letter *L* stands for *Tier 3*.

| Tier          | Content Type | Redundancy | Structure   | Acronym |
|---------------|--------------|------------|-------------|---------|
| **Size Minified < 100 bytes** |              |            |             |         |
| Tier 1        | Numeric      | Redundant  | Flat        | TNRF    |
| Tier 1        | Numeric      | Redundant  | Nested      | TNRN    |
| Tier 1        | Numeric      | Non-Redundant | Flat   | TNNF    |
| Tier 1        | Numeric      | Non-Redundant | Nested | TNNM    |
| Tier 1        | Textual      | Redundant  | Flat        | TTRF    |
| Tier 1        | Textual      | Redundant  | Nested      | TTRN    |
| Tier 1        | Textual      | Non-Redundant | Flat  | TTNF    |
| Tier 1        | Textual      | Non-Redundant | Nested | TNNN    |
| Tier 1        | Boolean      | Redundant  | Flat        | TBRF    |
| Tier 1        | Boolean      | Redundant  | Nested      | TBRN    |
| Tier 1        | Boolean      | Non-Redundant | Flat  | TBNF    |
| Tier 1        | Boolean      | Non-Redundant | Nested | TBNN    |
| **Size Minified ≥ 100 < 1000 bytes** |              |            |             |         |
| Tier 2        | Numeric      | Redundant  | Flat        | SNRF    |
| Tier 2        | Numeric      | Redundant  | Nested      | SNRN    |
| Tier 2        | Numeric      | Non-Redundant | Flat  | SNNF    |
| Tier 2        | Numeric      | Non-Redundant | Nested | SNNN    |
| Tier 2        | Textual      | Redundant  | Flat        | STRF    |
| Tier 2        | Textual      | Redundant  | Nested      | STRN    |
| Tier 2        | Textual      | Non-Redundant | Flat  | STNF    |
| Tier 2        | Textual      | Non-Redundant | Nested | STNN    |
| Tier 2        | Boolean      | Redundant  | Flat        | SBRF    |
| Tier 2        | Boolean      | Redundant  | Nested      | SBRN    |
| Tier 2        | Boolean      | Non-Redundant | Flat  | SBNF    |
| Tier 2        | Boolean      | Non-Redundant | Nested | SBNN    |
| **Size Minified ≥ 1000 bytes** |              |            |             |         |
| Tier 3        | Numeric      | Redundant  | Flat        | LNRF    |
| Tier 3        | Numeric      | Redundant  | Nested      | LNRN    |
| Tier 3        | Numeric      | Non-Redundant | Flat  | LNNF    |
| Tier 3        | Numeric      | Non-Redundant | Nested | LNNN    |
| Tier 3        | Textual      | Redundant  | Flat        | LTRF    |
| Tier 3        | Textual      | Redundant  | Nested      | LTRN    |
| Tier 3        | Textual      | Non-Redundant | Flat  | LTNF    |
| Tier 3        | Textual      | Non-Redundant | Nested | LTNN    |
| Tier 3        | Boolean      | Redundant  | Flat        | LBRF    |
| Tier 3        | Boolean      | Redundant  | Nested      | LBRN    |
| Tier 3        | Boolean      | Non-Redundant | Flat  | LBNF    |
| Tier 3        | Boolean      | Non-Redundant | Nested | LBNN    |
Figure 1: [8] introduces a formal taxonomy to class JSON documents that consists of 36 categories.

2.2 Size

- **Tier 1 Minified < 100 bytes.** A JSON document is in this category if its UTF-8 [2] minified form occupies less than 100 bytes.
- **Tier 2 Minified ≥ 100 < 1000 bytes.** A JSON document is in this category if its UTF-8 [2] minified form occupies 100 bytes or more, but less than 1000 bytes.
- **Tier 3 Minified ≥ 1000 bytes.** A JSON document is in this category if its UTF-8 [2] minified form occupies 1000 bytes or more.

2.3 Content Type

- **Textual.** A JSON document is in this category if it has at least one string value and its number of string values multiplied by the cumulative byte-size occupied by its string values is greater than or equal to the boolean and numeric counterparts.
- **Numeric.** A JSON document is in this category if it has at least one number value and its number of number values multiplied by the cumulative byte-size occupied by its number values is greater than or equal to the textual and boolean counterparts.
- **Boolean.** A JSON document is in this category if it has at least one boolean or null value and its number of boolean and null values multiplied by the cumulative byte-size oc-
cupied by its boolean and null values is greater than or equal to the textual and numeric counterparts.

- **Structural.** A JSON document is in this category if it does not include any string, boolean, null or number values.

A JSON document can be categorized as textual, numeric and boolean at the same time.

### 2.4 Redundancy

- **Non-redundant.** A JSON document is in this category if less than 25
- **Redundant.** A JSON document is in this category if at least 25 percent of its scalar and composite values are redundant.

### 2.5 Nesting

- **Flat.** A JSON document is in this category if the height of the document multiplied by the non-root level with the largest byte-size when taking textual, numeric and boolean values into account is less than 10. If two levels have the byte size, the highest level is taken into account.
- **Nested.** A JSON document is in this category if it is considered structural and its height is greater than or equal to 5, or if the height of the document multiplied by the non-root level with the largest byte-size when taking textual, numeric and boolean values into account is greater than or equal to 10. If two levels have the byte size, the highest level is taken into account.

### 3 JSON Stats Analyzer

The JSON Stats Analyzer is a free-to-use open-source web application that implements the taxonomy introduced in [8] and provides summary statistics. Using JSON Stats Analyzer, a user can determine the summary statistics corresponding to the taxonomy for any given JSON document. To complement the web application, the project also distributes a Node.js command-line interface program and a JavaScript library for programmatic use.

The JSON Stats Analyzer is developed using the JavaScript programming language, the CodeMirror version 5.65.0 open-source embeddable web editor (MIT License), the open-source Chart.js version 3.7.0 charting JavaScript library (MIT License), and the UI Kit version 3.10.0 CSS open-source web component framework (MIT License). The web application is deployed to the GitHub Pages free static-hosting service.

To make use of the JSON Stats Analyzer web application, the user opens https://sourcemeta.github.io/json-taxonomy/ on a web browser, writes or copy-pastes a JSON document of choice on the text editor at the left-hand side of the screen, and clicks the Analyze JSON button at the top right corner of the screen. The analysis results are presented at the right-hand side of the screen. If the syntax of the JSON document is invalid, an error is displayed at the right-hand side of the screen.

The analysis results are presented in the following order:

- The classification of the input JSON document according to the taxonomy introduced in [8].
- A set of pie charts representing the distribution of content type when taking into consideration the number of values of the input JSON document and the byte-size of the input JSON document in UTF-8 [2] minified form.
- A high-level summary of the input JSON document in terms of its byte-size in UTF-8 [2] minified form, its number of values, its height and its number of duplicated values.

---

2https://nodejs.org
3https://codemirror.net
4https://www.chartjs.org
5https://getuikit.com
6https://pages.github.com
4 Usage

4.1 JavaScript

We publish an npm\(^7\) package which can be installed as shown in Figure 2.

```
$ npm install --save @sourcemeta/json-taxonomy
```

Figure 2: The JavaScript module that implements the JSON Taxonomy is installed through the npm package manager.

The module exposes a single function that takes any JSON value and returns the sequence of taxonomy qualifiers as an array of strings. Its usage is exemplified in Figure 3.

```
const taxonomy = require('@sourcemeta/json-taxonomy')
const value = {
  foo: 2
}
console.log(taxonomy(value))
// ['tier 1', 'numeric', 'non-redundant', 'flat']
```

Figure 3: The JavaScript module exposes a single function that computes the JSON Taxonomy for any JSON value.

4.2 CLI

The npm package includes a simple command-line interface program that can be globally installed as shown in Figure 4.

```
$ npm install --global @sourcemeta/json-taxonomy
```

Figure 4: The companion command-line interface program that implements the JSON Taxonomy is also installed through the npm package manager.

The CLI program takes the path to a JSON document as an argument and outputs the taxonomy to standard output as shown in Figure 5.

5 Demonstration

In this section, we demonstrate various JSON documents for Tier 1, Tier 2 and Tier 3 as per the taxonomy defined in [8]. For each demonstration, we provide the definition, characteristics and document structure. These documents are as follows:

\(^7\)https://www.npmjs.com
Figure 5: An example of running the JSON Taxonomy command-line program with a fictitious JSON document file as input.

- **Grunt.js Clean Task.** Tier 1 Minified $< 100$ bytes, textual, redundant, flat. See Figure 6.
- **CircleCI Matrix.** Tier 1 Minified $< 100$ bytes, numeric, non-redundant, nested. See Figure 7.
- **TSLint Linter (Basic).** Tier 1 Minified $< 100$ bytes, boolean, non-redundant, nested. See Figure 8.
- **Entry Point Regulation Manifest.** Tier 2 Minified $\geq 100 < 1000$ bytes, textual, redundant, nested. See Figure 9.
- **TravisCI Notifications Configuration.** Tier 2 Minified $\geq 100 < 1000$ bytes, textual, redundant, flat. See Figure 10.
- **GeoJSON Example Document.** Tier 2 Minified $\geq 100 < 1000$ bytes, numeric, redundant, nested. See Figure 11.
- **GitHub FUNDING Sponsorship (Empty).** Tier 2 Minified $\geq 100 < 1000$ bytes, boolean, redundant, flat. See Figure 12.
- **NPM Package.json Example Manifest.** Tier 3 Minified $\geq 1000$ bytes, textual, non-redundant, flat. See Figure 13.
- **JSON Resume Example.** Tier 3 Minified $\geq 1000$ bytes, textual, non-redundant, nested. See Figure 14.
- **ESLint Configuration Document.** Tier 3 Minified $\geq 1000$ bytes, numeric, redundant, flat. See Figure 15.
- **Nightwatch.js Test Framework Configuration.** Tier 3 Minified $\geq 1000$ bytes, boolean, redundant, flat. See Figure 16.
5.1 Grunt.js Clean Task

**Definition.** Grunt.js is an open-source task runner for the JavaScript programming language used by a wide range of companies in the software development industry such as Twitter, Adobe, and Mozilla. In Figure 6, we demonstrate a **Tier 1 minified < 100 bytes textual redundant flat** (Tier 1 TRF from Table 1) JSON document that consists of an example configuration for a built-in plugin to clear files and folders called grunt-contrib-clean.

**Characteristics.** With a size of 92 bytes in UTF-8 minified form, 10 values, a height of 3 and 3 duplicated values, Grunt.js Contrib Clean is a JSON document whose content is determined by structure both in terms of number of values and cumulative byte-size. It has 4 non-structural values: 2 textual values and 2 boolean values. However, the textual values are 4.3% larger in byte-size than the boolean values.

**Document Structure.** This document contains equal number of boolean and textual values, but the cumulative byte-size of the textual values is larger than the cumulative byte-size of the boolean values. Therefore, this document is textual according to the taxonomy. The boolean and string values in the document have a 50% duplication. Therefore, this document is redundant according to the taxonomy. The height of this document is 3 and its largest level is 2. Therefore, this document is flat according to the taxonomy.

---

Figure 6: Grunt.js Clean Task

---

8https://gruntjs.com
9https://gruntjs.com/who-uses-grunt
10https://github.com/gruntjs/grunt-contrib-clean
5.2 CircleCI Matrix

**Definition.** CircleCI \(^{11}\) is a commercial cloud-provider of continuous integration and deployment pipelines used by a wide range of companies in the software development industry such as Facebook, Spotify, and Heroku \(^{12}\). In Figure 7, we demonstrate a **Tier 1 minified \(< 100 \text{ bytes numeric non-redundant nested}** (Tier 1 NNN from Table 1) JSON document that represents a pipeline configuration file for CircleCI that declares the desired CircleCI version and defines a workflow that contains a single blank matrix-based job.

**Characteristics.** With a size of 94 bytes in UTF-8 \([2]\) minified form, 13 values, a height \([8]\) of 9 and no duplicated values, CircleCI Matrix is a JSON document whose content is determined by structural values both in terms of number and cumulative byte-size. It only has non-structural values that are numeric and add up to less than 7% of the total byte-size.

**Document Structure.** Leaving structural values aside, this document only contains numeric values. Therefore, this document is **numeric** according to the taxonomy \([8]\). None of the values in this document are duplicated. Therefore, this document is **non-redundant** according to the taxonomy \([8]\). The height of this document is 9 and its largest level is 9. Therefore, this document is **nested** according to the taxonomy \([8]\).

![Figure 7: CircleCI Matrix](image-url)

\(^{11}\)https://circleci.com

\(^{12}\)https://circleci.com/customers/
5.3 TSLint Linter (Basic)

Definition. TSLint is now an obsolete open-source linter for the TypeScript programming language. TSLint was created by the Big Data analytics company Palantir and was merged with the ESLint open-source JavaScript linter in 2019. In Figure 8, we demonstrate a Tier 1 minified <100 bytes boolean non-redundant nested (Tier 1 BNN from Table 1) JSON document that consists of a basic TSLint configuration that enforces grouped alphabetized imports.

Characteristics. With a size of 66 bytes in UTF-8 minified form, 5 values, a height of 4 and no duplicated values, TSLint Linter (Basic) is a JSON document whose content is determined by structural values both in terms of number and cumulative byte-size. It has 1 non-structural value: a boolean that add up to less than 7% of the total byte-size.

Document Structure. Leaving structural values aside, this document only contains boolean values. Therefore, this document is boolean according to the taxonomy. None of the values in this document are duplicated. Therefore, this document is non-redundant according to the taxonomy. The height of this document is 4 and its largest level is 4. Therefore, this document is nested according to the taxonomy.

Figure 8: TSLint Linter (Basic)

---

13https://palantir.github.io/tslint
14https://www.typescriptlang.org
15https://www.palantir.com
16https://github.com/palantir/tslint/issues/4534
5.4 Entry Point Regulation Manifest

**Definition.** Entry Point Regulation (EPR) [5] is a W3C proposal led by Google that defines a manifesto that protects websites against cross-site scripting attacks by allowing the developer to mark the areas of the application that can be externally referenced. EPR manifests are used in the web industry. In Figure 9, we demonstrate a **Tier 2 minified ≥ 100 < 1000 bytes textual redundant nested** (Tier 2 TRN from Table 1) JSON document that defines an example EPR policy for a fictitious website.

**Characteristics.** With a size of 519 bytes in UTF-8 [2] minified form, 32 values, a height [8] of 4 and 10 duplicated values, Entry Point Regulation Manifest is a JSON document whose content is determined by textual values in terms of number but structural values in terms of cumulative byte-size. It has 20 non-structural values: 14 textual values, 1 numeric value and 5 boolean values. Together, the 20 non-structural values represent 50% of the cumulative byte-size.

**Document Structure.** This document includes textual, numeric and boolean values. Leaving structural values aside, this document is dominated by textual values in terms of number and cumulative byte-size. Therefore, this document is **textual** according to the taxonomy [8]. In this document, 3 out of 5 boolean values, 4 out of 14 textual values and 3 out of 12 structural values are duplicates. Therefore, 31.25% of its values are duplicates, making this document **redundant** according to the taxonomy [8]. The height of this document is 4 and its largest level is 3. Therefore, this document is **nested** according to the taxonomy [8].

![Figure 9: Entry Point Regulation Manifest](image-url)
5.5 TravisCI Notifications Configuration

**Definition.** TravisCI is a commercial cloud-provider of continuous integration and deployment pipelines used by a wide range of companies in the software development industry such as ZenDesk, BitTorrent, and Engine Yard. In Figure 10, we demonstrate a **Tier 2 minified \( \geq 100 < 1000 \text{ bytes} \)** **textual redundant flat** (Tier 2 TRF from Table 1) JSON document that consists of an example pipeline configuration for TravisCI that declares a set of credentials to post build notifications to various external services.

**Characteristics.** With a size of 672 bytes in UTF-8 minified form, 16 values, a height of 3 and 12 duplicated values, TravisCI Notifications Configuration is a JSON document whose content is determined by structural values in terms of number and textual values in terms of cumulative byte-size. All of its 7 non-structural values are strings that add to 75% of the total byte-size.

**Document Structure.** Leaving structural values aside, this document only contains textual values. Therefore, this document is **textual** according to the taxonomy. In this document, 6 out of 7 textual values and 6 out of 9 structural values are duplicates. Therefore, 75% of its values are duplicates, making this document **redundant** according to the taxonomy. The height of this document is 3 and its largest level is 3. Therefore, this document is **flat** according to the taxonomy.

![TravisCI Notifications Configuration](https://travis-ci.com)
5.6 GeoJSON Example Document

Definition. GeoJSON [1] is a standard to encode geospatial information using JSON. GeoJSON is used in industries that have geographical and geospatial use cases such as engineering, logistics and telecommunications. In Figure 11, we demonstrate a Tier 2 minified \( \geq 100 < 1000 \) bytes numeric redundant nested (Tier 2 NRN from Table 1) JSON document that defines an example polygon using the GeoJSON format.

Characteristics. With a size of 189 bytes in UTF-8 [2] minified form, 53 values, a height [8] of 5 and 21 duplicated values, GeoJSON Example Document is a JSON document whose content is determined by numeric values in terms of number but structural values in terms of cumulative byte-size. It has 31 non-structural values: 1 textual value and 30 numeric values. Together, the 31 non-structural values represent 49.7% of the cumulative byte-size.

Document Structure. This document includes numeric and textual values. Leaving structural values aside, this document is dominated by numeric values in terms of number and cumulative byte-size. Therefore, this document is numeric according to the taxonomy [8]. In this document, 18 out of 30 numeric values and 3 out of 22 structural values are duplicates. Therefore, 39.62% of its values are duplicates, making this document redundant according to the taxonomy [8]. The height of this document is 5 and its largest level is 5. Therefore, this document is nested according to the taxonomy [8].

![GeoJSON Example Document](image)

Figure 11: GeoJSON Example Document
5.7 GitHub FUNDING Sponsorship (Empty)

**Definition.** The GitHub software hosting provider defines a FUNDING file format to declare the funding platforms that an open-source project supports. The FUNDING file format is used by the open-source software industry. In Figure 12, we demonstrate a **Tier 2 minified ≥ 100 < 1000 bytes boolean redundant flat** (Tier 2 BRF from Table 1) JSON document that consists of a definition that does not declare any supported funding platforms.

**Characteristics.** With a size of 182 bytes in UTF-8 [2] minified form, 11 values, a height [8] of 1 and 8 duplicated values, GitHub FUNDING Sponsorship (Empty) is a JSON document whose content is determined by boolean values in terms of number but structural values in terms of cumulative byte-size. It has 10 non-structural values: 1 textual value and 9 boolean values. Together, the 10 non-structural values represent 29.1% of the cumulative byte-size.

**Document Structure.** This document is dominated by boolean values in terms of number. Leaving structural values aside, this document is dominated by boolean values in terms of cumulative byte-size. Therefore, this document is **boolean** according to the taxonomy [8]. In this document, 8 out of 9 boolean values are duplicates. Therefore, 72.73% of its values are duplicates, making this document **redundant** according to the taxonomy [8]. The height of this document is 1 and its largest level is 1. Therefore, this document is **flat** according to the taxonomy [8].

---

Figure 12: GitHub FUNDING Sponsorship (Empty)

---

18https://github.com
19https://docs.github.com/en/github/administering-a-repository/managing-repository-settings/displaying-a-sponsor-button-in-your-repository
5.8 NPM Package.json Example Manifest

**Definition.** Node.js Package Manager (NPM)\(^{20}\) is an open-source package manager for Node.js\(^{21}\), a JavaScript\(^4\) runtime targeted at the web development industry. A package that is published to NPM is declared using a JSON file called package.json\(^22\). In Figure 13, we demonstrate a **Tier 3 minified ≥ 1000 bytes textual non-redundant flat** (Tier 3 TNF from Table 1) JSON document that consists of a package.json manifest that declares a particular version of the Grunt.js\(^{23}\) task runner.

**Characteristics.** With a size of 2258 bytes in UTF-8\(^2\) minified form, 72 values, a height\(^8\) of 3 and 3 duplicated values, NPM Package.json Example Manifest is a JSON document whose content is determined by textual values in terms of both number and cumulative byte-size. It has 61 non-structural values. Together, the non-structural values represent 84.72% of the cumulative byte-size.

**Document Structure.** Leaving structural values aside, this document only contains textual values. Therefore, this document is *textual* according to the taxonomy\(^8\). In this document, 3 out of 61 textual values are duplicates. Therefore, only 4.17% of its values are duplicates, making this document *non-redundant* according to the taxonomy\(^8\). The height of this document is 3 and its largest level is 1. Therefore, this document is *flat* according to the taxonomy\(^8\).

![Figure 13: NPM Package.json Example Manifest](https://example.com/package.json)

\(^{20}\)https://www.npmjs.com
\(^{21}\)https://nodejs.org
\(^{22}\)https://docs.npmjs.com/cli/v6/configuring-npm/package-json
\(^{23}\)https://gruntjs.com
5.9 JSON Resume Example

**Definition.** JSON Resume is a community-driven proposal for a JSON-based file format that declares and renders themable resumes used in the recruitment industry. In Figure 14, we demonstrate a Tier 3 minified \( \geq 1000 \) bytes textual non-redundant nested (Tier 3 TNN from Table 1) JSON document that consists of a detailed example resume for a fictitious software programmer.

**Characteristics.** With a size of 3047 bytes in UTF-8 [2] minified form, 99 values, a height [8] of 4 and 2 duplicated values, JSON Resume Example is a JSON document whose content is determined by textual values in terms of both number and cumulative byte-size. It has 68 non-structural values. Together, the non-structural values represent 68.69% of the cumulative byte-size.

**Document Structure.** Leaving structural values aside, this document only contains textual values. Therefore, this document is textual according to the taxonomy [8]. In this document, 2 out of 68 textual values are duplicates. Therefore, only 2.02% of its values are duplicates, making this document non-redundant according to the taxonomy [8]. The height of this document is 4 and its largest level is 3. Therefore, this document is nested according to the taxonomy [8].

![Figure 14: JSON Resume Example](https://jsonresume.org)

---

[24]https://jsonresume.org
5.10 ESLint Configuration Document

Definition. ESLint 25 is a popular open-source extensible linter for the JavaScript [4] programming language used by a wide range of companies in the software development industry such as Google, Salesforce, and Airbnb. In Figure 15, we demonstrate a Tier 3 minified ≥ 1000 bytes numeric redundant flat (Tier 3 NRF from Table 1) JSON document that defines a browser and Node.js linter configuration that defines general-purposes and React.js-specific 26 linting rules.

Characteristics. With a size of 1140 bytes in UTF-8 [2] minified form, 54 values, a height [8] of 4 and 39 duplicated values. ESLint Configuration Document is a JSON document whose content is determined by numeric values in terms of number but structural values in terms of cumulative byte-size. It has 47 non-structural values: 3 textual values, 39 numeric values and 5 boolean values. Together, the 47 non-structural values represent only 9.91% of the cumulative byte-size.

Document Structure. This document includes textual, numeric and boolean values. While this document is dominated by numeric values in terms of number, numeric values do not make for the majority of cumulative byte-size by content type. However, the high difference in the number of numeric values and other types of values increases the numeric weight of the document and makes the document numeric according to the taxonomy [8]. In this document, 36 out of 39 numeric values and 3 out of 5 boolean values are duplicates. Therefore, 72.23% of its values are duplicates, making this document redundant according to the taxonomy [8]. The height of this document is 4 and its largest level is 2. Therefore, this document is flat according to the taxonomy [8].

![Figure 15: ESLint Configuration Document](https://eslint.org)

25 https://eslint.org
26 https://reactjs.org
5.11 Nightwatch.js Test Framework Configuration

**Definition.** Nightwatch.js is an open-source browser automation solution used in the software testing industry. In Figure 16, we demonstrate a Tier 3 minified ≥ 1000 bytes boolean redundant flat JSON document that consists of a Nightwatch.js configuration file that defines a set of general-purpose WebDriver and Selenium options.

**Characteristics.** With a size of 1506 bytes in UTF-8 minified form, 66 values, a height of 3 and 42 duplicated values, Nightwatch.js Test Framework Configuration is a JSON document whose content is determined by boolean values in terms of number but structural values in terms of cumulative byte-size. It has 56 non-structural values: 10 textual values, 14 numeric values and 32 boolean values. Together, the 56 non-structural values represent only 16.2% of the cumulative byte-size.

**Document Structure.** This document includes textual, numeric and boolean values. Leaving structural values aside, this document is dominated by boolean values in terms of number and cumulative byte-size. Therefore, this document is boolean according to the taxonomy. In this document, 5 out of 10 textual values, 4 out of 14 numeric values, 29 out of 32 boolean values and 4 out of 10 structural values are duplicates. Therefore, 63.64% of its values are duplicates, making this document redundant according to the taxonomy. The height of this document is 3 and its largest level is 1. Therefore, this document is flat according to the taxonomy.

![Nightwatch.js Test Framework Configuration](https://nightwatchjs.org)

---

27 https://nightwatchjs.org
28 https://www.selenium.dev
6 Conclusions and Further Work

In this paper, we present JSON Stats Analyzer, a free-to-use open-source web-based JavaScript tool that implements our proposed JSON taxonomy [8] that categorizes JSON documents according to their size, content, redundancy and nesting characteristics.

There are several avenues to extend our work: a visualization dashboard that compares multiple JSON documents across the tiers based on their content type, redundancy characteristics and document structure providing us with a meta analysis of how the documents with a variety of characteristics fare across the tiers; building a wider dataset of real-world JSON documents that demonstrates the characteristics pertinent to each tier. We welcome contributions to build this dataset from the industry professionals and academic community alike.

References

[1] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub. 2016. The GeoJSON Format. RFC. IETF. https://doi.org/10.17487/RFC7946

[2] The Unicode Consortium. 2019. The Unicode Standard, Version 12.1.0. Standard. The Unicode Consortium, Mountain View, CA.

[3] ECMA. 2017. ECMA-404: The JSON Data Interchange Syntax. ECMA, Geneva, CH. https://www.ecma-international.org/publications/standards/Ecma-404.htm

[4] ECMA. 2021. ECMA-262: ECMA-Script 2021 language specification. ECMA, Geneva, CH. https://www.ecma-international.org/publications/standards/Ecma-262.htm

[5] David Ross and Mike West. 2016. Entry Point Regulation. W3C Working Group Note. W3C. https://w3c.github.io/webappsec-epr/

[6] Simon Stewart and David Burns. 2020. WebDriver. W3C Working Draft. W3C.

[7] Santiago Vargas, Utkarsh Goel, Moritz Steiner, and Aruna Balasubramanian. 2019. Characterizing JSON Traffic Patterns on a CDN. In Proceedings of the Internet Measurement Conference (Amsterdam, Netherlands) (IMC ’19). Association for Computing Machinery, New York, NY, USA, 195–201. https://doi.org/10.1145/3355369.3355594

[8] Juan Cruz Viotti and Mital Kinderkhedia. 2022. A Benchmark of JSON-compatible Binary Serialization Specifications. arXiv:2201.03051 [cs.SE]