Soutenance de PSTL

Benchmarking de solutions optimistes pour génération de données test à partir de JSON Schema

—

Auteurs:
Zaky Abdellaoui &
Abdelkader Boumessaoud

Encadrants:
Mohamed-Amine Baazizi
Lyes Attouche

Qu’est ce que JSON Schema ?

JSON-Schema : Exemple

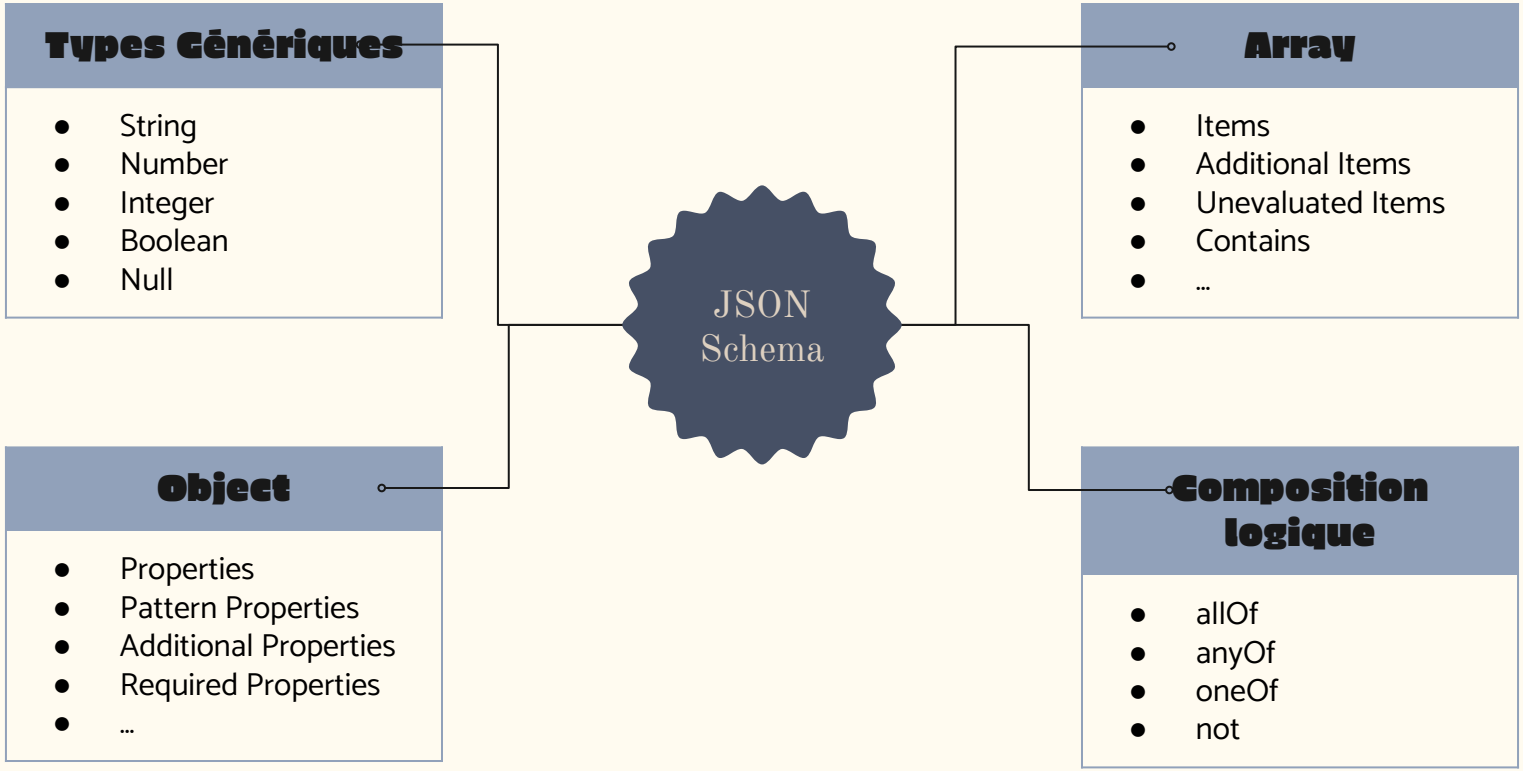
```json
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "number"
    },
    "email": {
      "type": "string",
      "format": "email"
    }
  }
}
```

Schéma

```json
{
  "name": "John Doe",
  "age": 25,
  "email": "john.doe@example.com"
}
```

Instance valide

JSON-Schema : Description



JSON-Schema : Exemple de composition logique

```json
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    }
  },
  "not": {
    "required": ["age"]
  }
}
```

Schéma

```json
{
  "name": "John Doe",
  "age": 25
}
```

Instance non valide

Génération de données JSON Schema

Génération d'instances : Bibliothèques

- *json-schema-faker*

- *json-everything*

- *json-data-generator*

Génération d'instances : Bibliothèques

- *json-schema-faker*

- *json-everything*

- *json-data-generator* ⟶ Documenté

Génération d'instances : Bibliothèques

- *json-schema-faker*

- *json-everything* ⟶ 59.30 %

- *json-data-generator* ⟶ Documenté

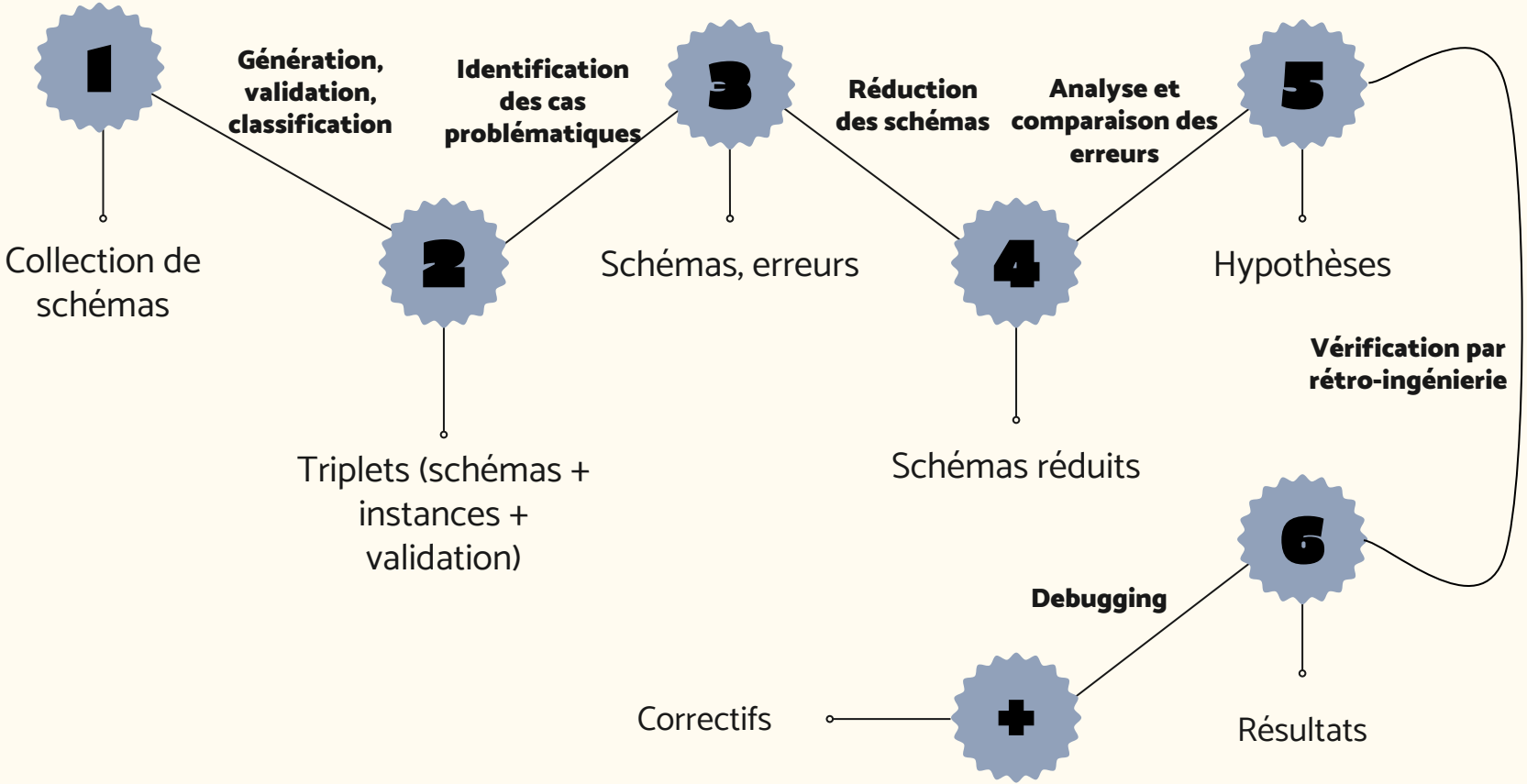
Génération d'instances : Bibliothèques

- *json-schema-faker* ⟶ 89.75 %

- *json-everything* ⟶ 59.30 %

- *json-data-generator* ⟶ Documenté

Notre approche



Résultats : Limitations logiques

Limitations “fortes”, d’ordre logique

- Négation
- Disjonction exclusive
- Conjonction d’une proposition avec une négation

Limitations “faibles”, résolubles

- Nombre de propriétés
- Génération de propriétés non spécifiées

Exemple : conjonction proposition/négation

```
{
    "type": "object",
    "properties": {
        "prop": {
            "not": {
                "type": "number"
            }
        }
    },
    "required": [
        "prop"
    ]
}
```

Génère des instances valides

```
{
    "type": "object",
    "properties": {
        "prop": {
            "allOf": [
                {
                    "type": "string"
                },
                {
                    "not": {
                        "type": "number"
                    }
                }
            ]
        }
    },
    "required": [
        "prop"
    ]
}
```

Génère des instances erronées

Synthèse des résultats et observations

- Code peu commenté et peu lisible
- Approche empirique et non logique
- Implémentations incomplètes
- Impossibilité d’atteindre des implémentations correctes à 100%

Conclusion & Perspectives

- Contrainte principale : manque de temps pour approfondir le projet.
- Investissement important dans la maîtrise et la compréhension du langage et ses outils vu que nouvelles pour l'équipe.
- Souhait d'approfondir l'analyse des erreurs en expérimentant
- Objectif d'effectuer une étude comparative approfondie sur d’autres solutions de génération

Merci de votre attention

Génération d'instances : Benchmark

Générateur	Dataset .	Total (schémas)	Instance (Valid/Invalid)	Taux de validité / Dataset (%)	Taux de validité general (%)
json-schema-faker	Snowplow	409	402 / 7	98.28 %	89.75 %
	WashingtonPost	125	102 / 23	81.60 %	
	Kubernetes	1082	984 / 98	90.94 %	
	GitHub	5957	5254 / 703	88.19 %	
json-everything	Snowplow	409	307 / 102	75.06 %	59.30 %
	WashingtonPost	125	48 / 77	38.40 %	
	Kubernetes	1082	699 / 383	64.60 %	
	GitHub	5957	3524 / 2433	59.15 %	

Taux de validité des instances générées

Approche adoptée : Résumé de la classification

Datasets	Erreurs	Taux d'occurrence
Snowplow	type maxProperties minProperties	42.86 % 28.57 % 28.57 %
WashingtonPost	additionalProperties type required enum dependencies	52.17 % 17.39 % 13.04 % 13.04 % 4.35 %
Kubernetes	type required unionType	82.65 % 16.33 % 1.02 %
GitHub	required type maxItems . . .	34.57 % 17.78 % 7.97 % 39.86 %

Taux d'occurrence des erreurs de validation

Validation de données JSON : Exemple

```
{
  "type": "object",
  "properties": {
    "first_name": { "type": "string" },
    "last_name": { "type": "string" },
    "birthday": { "type": "string", "format": "date" },
    "address": {
      "type": "object",
      "properties": {
        "street_address": { "type": "string" },
        "city": { "type": "string" },
        "state": { "type": "string" },
        "country": { "type" : "string" }
      }
    }
  }
}
```

Schéma

```
{
  "first_name": "George",
  "last_name": "Washington",
  "birthday": "1732-02-22",
  "address": {
    "street_address": "3200 Mount Vernon Memorial Highway",
    "city": "Mount Vernon",
    "state": "Virginia",
    "country": "United States"
  }
}
```

```
{
  "name": "George Washington",
  "birthday": "February 22, 1732",
  "address": "Mount Vernon, Virginia, United States"
}
```

Instances