État de l’art

Appariement de schémas JSON

Rédigé par Tanguy SMODIS

Projet d’ouverture à la recherche
Université Claude Bernard Lyon 1

Introduction

L’appariement de schémas de données consiste à régler les problèmes qui surviennent quand on cherche à intégrer ou fusionner des données provenant de différentes sources de données pour être capable de les exploiter de manière uniforme.

Ce sujet a été proposé par l’enseignant-chercheur et maître de conférence du LIRIS Fabien DUCHATEAU.

Ce processus crucial a déjà été étudié pour des bases de données en relationnel et en XML mais très peu en JSON. Cela est dû au fait que ce modèle est dit semi-structuré, il est donc plus difficile d’exploiter la façon dont les données sont stockées et les règles qui régissent ce stockage pour faire l’appariement. De plus, le schéma d’un document JSON n’est pas explicite et un ensemble de documents JSON, même stockés dans une même collection, ne partage pas forcément un schéma commun.

Il est cependant possible de faire de l’appariement s’appuyant sur les schémas si l’on s’appuie également sur d'autres méthodes visant à mesurer la similarités entre deux entrées d’une base de données.

Dans cet état de l'art, je commencerais par décrire les connaissances requises afin de comprendre la globalité du sujet puis je décrirai les difficultés observées par les chercheurs et les solutions développées pour y pallier. Je présenterai également différents outils créés pour mesurer l’efficacité des algorithmes d’appariement.

Cet état de l’art est annoté par de nombreux symboles(➿, ✜, ①, ❶, …) qui légendent les différents exemples présent dans les figures.

Prérequis

Comme indiqué dans l'introduction, l'appariement consiste à fusionner des bases de données de façon cohérente afin de pouvoir les exploiter par la suite.

Une base de données JSON peut prendre différentes formes pour représenter des données similaires. Ces données sont des objets qui peuvent être séparés en deux catégories, à savoir les littéraux et les collections. Les littéraux sont des valeurs atomiques souvent représentées par des nombres, des chaînes de caractères ou des booléens. Les collections sont des tableaux, des dictionnaires ou des records, chacun ont leur particularité mais ils contiennent tous un ensemble d'objets.

Les données sont organisées dans une structure imbriquée qui permet de comprendre facilement quel champ correspond à quelle entrée de la base.

L'exactitude de cette organisation ne dépend cependant que des façons de procéder des créateurs de la base ainsi que leurs habitudes.

Il est possible en JSON de référencer un autre document ou une autre collection. Par exemple dans le cas d'une facture il peut être utile d'avoir accès aux données du client, cela peut se faire de deux façons différentes :

- par inclusion (⇾ Fig1) : les données sont intégrées (généralement comme sous-document) dans le document qui souhaite référencer des données, ce qui nécessite de maintenir plusieurs documents à jour lors d'une modification sur les champs concernés mais permet de récupérer toutes les informations lors d'une interrogation

- par référence (✜ Fig1): le champ contient l'identifiant d'une entrée dans un autre document

L'appariement n'est que la première étape d'un long processus lorsque l'on fusionne des bases de données puisqu'il faut aussi fusionner les schémas, dédoublonner, etc.

Difficultés lors de l’appariement

Fig 1 : Base de données simple en JSON illustrant les particularités de ce langage

```
{
	"$ref" : "./antagonistes"
	"protagonistes": [①
		{
			"id": 0, ❶
			"login" : "Jojo",
			"mot_de_passe" : "H3rm!tPurpl3",
			"details" : {①
				"ville" : "New York",
				"telephone" : "06 74 15 23 68",
				"nom" : "Joestar",
				"prenom" : "Joseph"
			}
			"adversaire" : {"$ref" : "#/0"} ✤
		},
		{
			"id": 1,
			"login" : "JPP",
			"mot_de_passe" : "S!lv3rCh4r!ot",
			"details" : {
				"ville" : "Bordeaux",
				"telephone" : "07 69 27 41 55",
				"nom" : "Jean-Pierre",
				"prenom" : "Polnareff"
			}
			"adversaire" : {
				"login" : "Jojo",
				"mot_de_passe" : "Th3W0rld",
				"details" : {
					"ville" : "Le Caire",♦
					"telephone" : "06 66 99 33 44",
					"nom" : "Dio",
					"prenom" : "Brando"
				}
			}➪
		}
	]
}
{
	"antagonistes": [
		{
			"id": 0, ❶
			"login" : "Jojo",
			"mot_de_passe" : "Th3W0rld",
			"details" : {
				"ville" : "30° 2' 39.912" N 31° 14' 8.563" E",♦
				"telephone" : "06 66 99 33 44",
				"nom" : "Dio",
				"prenom" : "Brando"
			}
		}
	]
}
```

Appariement de Schéma

Semi-Structure du schéma (① Fig1):

Les données JSON sont stockées et réglées par des schémas semi-structurés. Cela s'explique par le fait qu'un objet JSON, qu'il soit un tableau, un record ou un dictionnaire peut contenir un autre objet JSON au lieu d'un simple élément atomique.

Fig 2 : Deux bases identiques en terme de données, avec ou sans schéma[1]

Detailed schema

```
{
   "Publication": "FoodNetwork",
   "Comments": [
      {
         "ID": 907,
         "Content": "Love this recipe!"
      },
      {
         "ID": 909,
         "Content": "I've had better"
}]}
```

No Schema

```
{
   "FoodNetwork": [
      [
         907,
         "Love this recipe!"
      ],
      [
         909,
         "I've had better"
      ]
]}
```

Manque de typage/de nom

Certaines données ne sont pas typées (int, string, bool,...), d'autres n'ont pas de nom décrivant le champ qu'elle renseigne.
Dans la figure 2, la base sans schéma ne possède aucun nom, les clés sont également des valeurs. Aucune de ces bases n'indiquent de types pour leurs champs.

Différences dans l'imbrication (nesting).

Puisque l'imbrication permet de savoir dans quelle entité ou partie de l'entité on se trouve, de petites différences dans la structure générale de l'imbrication peuvent nuire à la compréhension du document. Cela peut aussi perturber un algorithme d'appariement en déplaçant des informations sur les mauvaises entités.
Si, dans la figure 2, le tableau <[909,"I've had better"]> était décalé d'une indentation vers la gauche, cela ne changerait pas la structure mais nuirait à la compréhension du document.

[1] Kunal Waghray, *JSON Schema Matching: Empirical Observations*, 2020

Séparation entre schéma et donnée dépendante de l'utilisateur

Dans certains schéma, selon qui l'observe, certains champ peuvent être des clés ou des valeurs.
Dans la figure 2, “FoodNetwork” est une clé sans schéma et une valeur avec. L’ID est stocké dans un couple clé-valeur avec schéma et dans un tableau sans.

Appariement d’Entité

Collision de label (❶ Fig1) :

JSON n’utilise pas d’espaces de noms (namespace) contrairement à XML. Un namespace sert à différencier deux entités portant le même identifiant (label) si elles sont dans deux namespaces différents.
Dans le cas où deux entités partagent le même identifiant, une seule sera conservée et écrasera les données de l’autre entité.
Dans la figure 1, Joseph pourrait voir son compte être remplacé par celui de Dio

Hétérogénéité des données (♦ Fig1) :

La même information peut avoir différentes représentations bien que l’information représentée en elle-même soit identique indépendamment de sa représentation.
Un lieu peut être caractérisé soit par une adresse (chaîne de caractères, éventuellement découpée en plusieurs propriétés), soit par des coordonnées géographiques.
“2 rue des capucines, Amplepuis” et “45.97106336600258, 4.341183997291023” indiquent toutes les deux des adresses.

Différent termes pour même représentation / même terme pour différentes représentations :

Un même terme (acronyme, logo, symbole) peut avoir plusieurs significations de même qu’il est possible d’utiliser plusieurs termes différents pour faire référence à un seul et même objet.
“ISWC” au lieu de “International Semantic Web Conference” pour le nom d’une conférence.
“ISWC” au lieu de "International Standard Musical Work Code" pour un identifiant d'œuvre musicale.

Problèmes globaux

Ces problèmes inhérents au langage JSON ne sont pas les seuls à s'interposer dans le processus d'appariement. D'autres difficultés indépendantes du langage et du modèle de base de données peuvent survenir notamment liés aux facteurs humain et logiciel.

L'erreur est humaine et les bases de données n'y échappent pas, il arrive naturellement de saisir des données erronées ou ne correspondant pas au schéma tacite ou aux conventions de nommage, ces erreurs passent souvent inaperçues car non-relevés par la machine et nécessitent un regard humain pour être détectées.

Les appariements réalisés par un algorithme ne sont pas suffisamment fiables pour ne pas nécessiter la vérification post-process d'un humain pour valider les résultats. Le travail de validation est sans commune mesure avec le travail effectué par l'algorithme qui se charge de détecter la majeure partie des erreurs en amont, l'humain doit simplement vérifier et apporter une décision en cas de doute de la part de l'algorithme.

Solutions recherchées

Puisqu'il n'est pas suffisant de s'appuyer uniquement sur le schéma de la base de données, plusieurs méthodes ont fait l'objet d'études[2] dans le but de les combiner afin de mesurer la similarité entre deux entrées appartenant à des bases de données différentes. Ces mesures peuvent être linguistiques, structurales, basées sur les contraintes ou les règles, sur un réseau, sur le contenu, etc.
L'appariement s'effectue ensuite sur les résultats des diverses combinaisons de ces méthodes :

- Similarité Linguistique : basé sur l'exploitation des chaînes de caractères définissant les éléments comme les noms et les descriptions
- Similarité d'Instance : le schéma est vu comme similaire si ses instances le sont, basé sur des statistiques, des métadonnées ou des classificateur entraînés
- Similarité de Structure : basé sur les relations ou les chemins entre les éléments
- Similarité Graphique : basé sur les relations dans le schéma des graphes
- Similarité des contraintes : basé sur le type de données, leur variance, unicité, nullabilité et clés étrangères
- Similarité des Règles : basé sur des règles d'appariement exprimé par la logique de premier ordre
- Similarité d'Utilisation : basé sur l'analyse des requêtes effectué sur la base de données par ses utilisateurs pour déduire des informations sur comment ils utilisent cette base (égalité de jointure, champ contenant des liens comparés avec les SEO)
- Similarité de Contenu : basé sur la TF-IDF (term frequency-inverse document frequency) qui détermine l'importance d'un terme dans un document donc d'un champ dans une base. En regroupant les documents partageant un TF-IDF similaires sur les mêmes termes.
- Similarité des Liens : deux ontologies (modèle de données contenant des concepts et relations permettant de modéliser un ensemble de connaissances dans un domaine donné) sont traités comme similaires si les entités s'y référençant sont similaires
- Utilisation de Données Auxiliaires : acronymes, convention de nommages, dictionnaire officiels, etc.

Un outil d'appariement pourrait faire la moyenne entre le score d'une mesure linguistique sur le titre, le score d'une mesure linguistique sur l'auteur et le score de similarité d'une mesure sur le contenu de sa description afin de décider si deux état de l'art ont le même sujet.

---

[2] Philip A Bernstein, Jayant Madhavan, and Erhard Rahm. *Generic schema matching, ten years later. Proceedings of the VLDB Endowment*, 4(11):695–701, 2011

Ces méthodes sont accompagnées par des stratégies de traitements telles que le *workflow*, la parallélisation ou le *blocking* permettant de comparer des bases de données de grande envergure contenant énormément d'entités.

Le machine learning peut être utilisé pour effectuer l'appariement, dans le cas d'un apprentissage supervisé ou semi-supervisé il peut être nécessaire de conserver des traces lors de l'appariement afin de pouvoir l'entraîner. Il est également possible d'utiliser des méthodes plus simples à base de mécanismes de décision stricts (ex : un seuil déterminé arbitrairement à 0.95 pour détecter uniquement les résultats les plus similaires) les résultats de ce premier jet permettront d'entraîner le classificateur. L'apprentissage non supervisé est également envisageable, auquel cas les traces ne seront pas nécessaires.

Certains groupes de chercheurs ont tenté d'inférer un schéma au JSON[3] permettant à l'utilisateur de connaître les champs obligatoires ou facultatifs et les possibilités de ce champ (limites, type, relations, etc.). Cela permet de solutionner en partie les problèmes causés par l'absence de schéma en JSON notamment l'absence de typage, ce qui permet de simplifier l'application des méthodes de mesure puisqu'il n'est plus nécessaire de vérifier la similarité entre deux valeurs de types différents. L'inférence de schéma n'assiste en rien le processus d'appariement en ce qui concerne les problèmes au niveau entité.

[3] Mohamed-Amine Baazizi, Houssem Ben Lahmar, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. *Schema Inference for Massive JSON Datasets. In Extending Database Technology (EDBT)*, Venise, Italy, 2017.

Outils existants

JSONGlue[4]

Basé sur la linguistique, la sémantique et la distribution des données dans le but d'effectuer des appariement entre différents schémas, JSONGlue est un outil récent créé en 2020 par une équipe de chercheurs de l'université fédérale ABC situé à Santo André - SP - Brésil.

Il fonctionne en quatre étapes principales, un travail préliminaire suivi de trois modules :

1. Travail préliminaire : JSONGlue commence par normaliser les données et cherche s'il existe un document décrivant les conventions de nommage de la BDD. Il crée ensuite un graphe déconnecté composé de sous-graphes qui correspondent chacun à un schéma de données.

2. Module linguistique : Chaque sous-graphe est analysé progressivement en mesurant la similarité de chaque chaîne de caractère via l'algorithme Jaro-Winkler, si le résultat est jugé suffisamment bon, une arête est créée entre les deux éléments.

3. Module sémantique : il mesure la similarité entre chaque paire de nœuds en se basant sur la base de données WordNet monolingual, il relie aussi les deux nœuds similaires et y attache la valeur du résultat de l'algorithme sémantique.

4. Module distribution de données : Il mesure la distance entre chaque paire de nœuds, la déviation standard et la distance entre deux histogrammes décrivant la fréquence des caractères dans la paire de nœuds.

JSONGlue assiste la supervision humaine en générant à chaque étape des graphes permettant de visualiser le processus.

Cet outil a été testé sur une base de données du COVID-19 et les chercheurs ont remarqué que les résultats sont significativement meilleurs si tous les schémas suivent une convention de nommage et/ou s'ils utilisent des termes techniques d'un même domaine. L'absence de tels paramètres nuit sérieusement à la fiabilité du résultat mais d'autres problèmes tels que la nomenclature irrégulière, les fautes de frappes, les noms propres et la concaténation excessive peuvent aussi intervenir pour diminuer la fiabilité des résultats.

Actuellement JSONGlue supporte la parallélisation ce qui augmente ses performances lors de longues analyses et ses développeurs aimerait permettre l'intervention humaine lors d'un processus et amplifier les capacités d'analyse en ajoutant la capacité d'exploiter les représentations de connaissances comme les ontologies.

[4] Vitor Marini Blaselbauer and Joao Marcelo Borovina Josko. *Jsonglue: A hybrid matcher for json schema matching. In Companion Proceedings*, page 77, 2020.

EMBench[5]

Cet outil est un benchmark permettant de montrer les capacités et les limitations des outils de matching. Il permet à l'utilisateur de sélectionner les métriques qu'il souhaite évaluer lors de la comparaison entre différents algorithmes d'appariements. L'outil n'étant pas spécifique à JSON et pouvant s'adapter à toutes sortes de langages, il ne considère pas les problèmes de représentation (référence par inclusion ou par référence, séparation schéma/données) puisqu'il uniformise les bases de données pour en former une véritable base de test.

EMBench définit cinq catégories d'appariement :

1. Étudier les similarités au niveau atomique : très sensible aux fautes de frappes et conventions de nommage.

2. Voir les objets comme des ensemble de valeurs atomiques : concaténation de plusieurs données atomiques et comparaison du résultat.

3. Appariement collectif : exploiter les informations de deux sets de données en analysant les liens que partagent les objets et leur hiérarchie.

4. Études de blocs de données : les données sont organisées dans des blocs définis selon des paramètres et sont comparées aux autres données du même bloc souvent plus proches.

5. Exploiter les données du schéma : puisqu'un schéma décrit la structure des objets de la collection et qu'une entité fait partie de cette collection, elle se plie à toutes les règles mais ne remplit pas forcément tout l'espace qu'on lui réserve ce qui peut poser problème

[5] Ekaterini Ioannou, Nataliya Rassadko, and Yannis Velegrakis. *On Generating Benchmark Data for Entity Matching. Journal on Data Semantics*, 2(1):37–56, 2013.

Conclusion

L'appariement de base de donnés en JSON ne diffèrent pas énormément de l'appariement dans d'autres langages lorsque l'on observe les méthodes utilisées mais le schéma non structuré propre à JSON impose de rajouter une surcouche d'analyse pour contrebalancer ce manque.

Il existe encore à ce jour peu d'études ou d'avancées sur ce sujet si l'on compare aux autres langages de gestion de données (notamment Relationnel, XML ou RDF), on peut en déduire que cette surcouche est complexe et le besoin d'appareiller des bases en JSON n'est pas aussi important que celui d'autres langages.

En effet, certains domaines exigent un certain langage que ce soit par nécessité pour la représentation des données ou par la simple popularité de ce langage et la possibilité d'appareiller leurs données est une nécessité que ce soit pour les uniformiser dans le cadre d'une entreprise ou rassembler les travaux de plusieurs laboratoires internationaux.

L'utilisation plus répandue d'autres langages et leur complexité moindre explique la différence en nombre de travaux cela ne veut pas dire pour autant que l'appariement en JSON ne sera jamais aussi développé qu'en SQL aujourd'hui, le JSON est souvent utilisé dans de nombreux domaines (Web, Apprentissage des données, Programmation JavaScript) car très modelable par son absence de schéma. Il est très probable que les travaux sur l'appariement JSON rattraperont ceux des autres langages quand ceux-ci auront atteint un plafond.