SemanticBot: Intégration Semi-Automatique de Données au Web des Données

Benjamin Moreau, Nicolas Terpolilli, Patricia Serrano-Alvarado

SemanticBot: Intégration Semi-Automatique de Données au Web des Données

Benjamin Moreau*,** Nicolas Terpolilli**,
Patricia Serrano-Alvarado*

*GDD-LS2N – Nantes University, France
{Name.LastName@}univ-nantes.fr,
https://www.ls2n.fr
**OpenDataSoft
{Name.LastName}@opendatasoft.com
https://www.opendatasoft.com

Résumé. L'intégration de données structurées au Web des données est possible avec des *Mappings RDF*. Cependant, pour le développement de ces mappings il est nécessaire d'être familier avec RDF, en plus de connaître parfaitement le jeu de données. Le faible nombre de personnes satisfaisant ces deux conditions est un obstacle à la démocratisation du Web des données. Nous présentons ici un outil capable de générer, semi-automatiquement, des mappings RDF pour des jeux de données structurés. Le défi consiste à automatiser la partie du processus d'intégration qui nécessite que l'utilisateur soit familiarisé avec RDF.

1 Introduction et Motivation

Le Web des données est un ensemble de bonnes pratiques pour publier des données au format RDF [1]. Les données publiées dans le Web des données sont décrites à l'aide d'ontologies. Une ontologie est un ensemble de concepts (i.e., classes) et de relations (i.e., propriétés) représentant un domaine de connaissance. RDFS [2] et OWL [3] sont des langages permettant de décrire ces ontologies en RDF. Utiliser des ontologies déjà existantes est une bonne pratique qui permet d'accroitre l'interopérabilité entre les jeux de données du web des données.

Un *Mapping RDF* définit la transformation d'un jeu de données structuré (relationel, JSON, etc.) vers un jeu de données au format RDF. Cependant, développer un mapping RDF n'est pas trivial. Le Tableau 1 représente un extrait d'un jeu de données orienté colonnes. La Figure 1 est un mapping RDF permettant de transformer ce jeu de données en RDF. Pour écrire un tel mapping, il est nécessaire d'avoir la réponse à certaines questions, par exemple: (i) A quels concepts appartiennent les instances des colonnes *Name* et *Birth city* ? Ici, *Name* contient des personnes et *Birth city* des lieux. (ii) Quels sont les relations entre ces concepts ? Dans cet exemple, les lieux sont les villes de naissance des personnes. (iii) Quels sont les ontologies existantes pour décrire ce jeu de données. Dans ce cas, nous pourrions utiliser DBpedia, Schema.org, etc.

1. https://www.w3.org/RDF/
2. https://www.w3.org/TR/rdf-schema/
3. https://www.w3.org/TR/owl2-overview/

string	date	string	string	float	float
Name	**Birth**	**Birth City**	**Birth Province**	**Lat**	**Long**
Augustus	0062-09-23	Rome			
Caligula	0012-08-31	Antitum			
Claudius	0009-08-01	Lugdunum	Gallia Lugdunensis	47.932559	0.191854
...	...	...	...	...	...

TAB. 1 – *Un extrait de jeu de données structuré représentant des empereurs romains.*

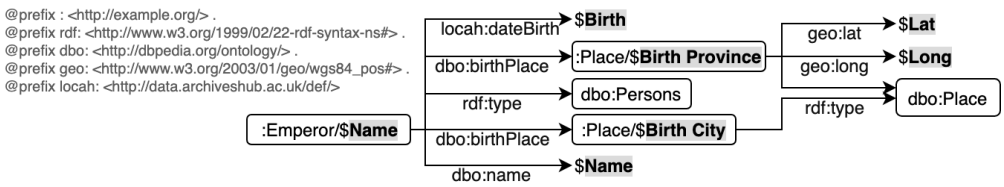

FIG. 1 – *Un mapping RDF du jeu de données des empereurs romains. Les informations en gras commençant par $ référencent les colonnes du dataset.*

Pour répondre à ces questions, il est à la fois nécessaire de bien connaitre le jeu de données et son domaine de connaissance, mais aussi, d'être familier avec RDF (RDFS, OWL et les langages de mapping RDF). Malheureusement, un nombre important de producteurs de données ne sont pas familiarisés avec le Web des données et ne sont pas encore prêts à s'y investir. Notre problème de recherche est le suivant: *Comment simplifier l'intégration de données structurées dans le Web des données ?* Le défi auquel nous sommes confrontés est d'automatiser la partie du processus d'intégration qui nécessite que l'utilisateur soit familiarisé avec RDF.

RML(1) et SPARQL-Generate(6) sont deux langages de mapping RDF. Même si il existe des syntaxes simplifiées tel que YARRRML(3), écrire un mapping RDF nécessite d'être familier avec RDF. Récemment, des outils tels que KARMA(2), RMLeditor(4) ou Juma(5) ont été proposés pour assister les utilisateurs pendant la création de mappings RDF. Cependant, ces outils ne sont pas encore adaptés aux utilisateurs n'étant pas familiers avec RDF.

Nous proposons *SemanticBot*, un outil capable de générer un mapping RDF pour un jeu de données structuré. Il fonctionne de manière semi-automatique en ne posant que des questions à l'utilisateur à propos de son jeu de données. Une version anglaise de cette démonstration a été publiée à ISWC 2019 [4].

2 L'outil SemanticBot

Afin de générer un mapping RDF pour un jeu de données structuré, *SemanticBot* se base sur les graphes de connaissances DBpedia et YAGO, les ontologies existantes dans LOV [5] et les langages OWL et RDFS.

4. https://hal.archives-ouvertes.fr/hal-02194315v1
5. https://lov.linkeddata.es/dataset/lov/

En résumé, pour chaque colonne du jeu de données, *SemanticBot* extrait un ensemble d'instances et cherche les entités correspondantes dans DBpedia et YAGO. L'objectif est de trouver les classes pouvant décrire les instances de chaque colonne. Ensuite, LOV est utilisé pour trouver les propriétés les plus cohérentes avec le nom et le types de chaque colonne. Ainsi, chaque instance de la colonne correspondra à l'objet de la propriété. Pour confirmer et enrichir ces correspondances, *SemanticBot* pose des questions simples à l'utilisateur. Les réponses aux questions servent à générer un premier mapping RDF qui est, par la suite, saturé selon des règles OWL et RDFS.

Nous détaillons ici les différentes étapes de notre outil en utilisant comme exemple le jeu de données des empereurs romains du Tableau 1.

Identifier les classes. Dans la première étape, l'instance *Augustus* de la colonne *Name* correspond à l'entité `http://dbpedia.org/resource/Augustus` de classe `dbo:Person`. *Augustus* est donc identifié comme entité de classe `dbo:Person`. Quand cette étape est terminée, nous obtenons deux correspondances suggérant que les colonnes *Name* et *Birth City* contiennent respectivement des entités de classes `dbo:Person` et `dbo:Place`. Ces suggestions sont ensuite proposées à l'utilisateur sous la forme de questions simples: "La colonne Name dans votre dataset contient-elle des Personnes ?". Pour cacher les URIs, les questions sont formulées en utilisant l'attribut `rdfs:label` des classes.

Identifier les propriétés et leurs objets. À l'étape suivante, la recherche des propriétés dans LOV, à l'aide du nom et type des colonnes, nous donne 5 résultats. Les colonnes *Name*, *Birth*, *Birth City*, *Lat* et *Long* sont respectivement identifiées comme contenant les objets des propriétés `dbo:name`, `locah:dateBirth`, `dbo:birthPlace`, `geo:lat` et `geo:long`. Comme précédemment, ces suggestions sont proposées à l'utilisateur: "La colonne Lat représente t-elle la Latitude d'un objet localisé ?". La question est formulée avec le `rdfs:label` et le `rdfs:domain` des propriétés.

Identifier les sujets des propriétés. Pour compléter, l'outil demande à l'utilisateur, pour chaque propriété validée, de sélectionner la colonne contenant les sujets de la propriété. Si l'utilisateur confirme la propriété `geo:lat` sur la colonne *Lat*, l'outil demande: "latitude est un attribut d'un objet localisé. Sélectionnez la colonne contenant ces objets localisés.". Dans notre exemple, si l'utilisateur répond correctement à la question, la colonne *Birth Province* sera associée au sujet de la propriété `geo:lat`.

Notre outil utilise des heuristiques pour limiter le nombre de questions posées à l'utilisateur: (i) Il ne suggère qu'au plus une classe et une propriété par colonne. (ii) Seule la classe correspondant au plus grand nombre d'instances dans une colonne est suggérée. (iii) La propriété suggérée pour une colonne est celle qui a le meilleur score de popularité sur LOV. (iv) Enfin, une propriété n'est pas proposée à l'utilisateur si son score LOV est inférieur à une valeur fixée.

L'ensemble des suggestions confirmées par l'utilisateur (classes et propriétés) sont utilisées pour générer un premier mapping RDF, lequel est saturé en lui appliquant les règles d'inférence RDFS et OWL [6].

6. Nous considérerons uniquement les règles RDFS `https://www.w3.org/TR/rdf11-mt/#rdfs-entailment` 2, 3, 5, 7, 9 et 11 et les règles OWL `https://www.w3.org/TR/owl-ref` basées sur `owl:equivalentClass` et `owl:equivalentProperty`.

Le mapping RDF de notre exemple est disponible en YARRRML à l'adresse `https://git.io/fjKY6` et le résultat de la transformation selon ce mapping est disponible ici: `https://git.io/fjKYo`.

3 Démonstration

SemanticBot encourage de nouveaux utilisateurs à faire leur premier pas dans le Web sémantique. Il peut être testé ici `https://semanticbot.opendatasoft.com/` avec des jeux de données d'Opendatasoft [7]. Des explications supplémentaires sur le fonctionnement de l'outil ainsi que son code source sont disponibles sur GitHub [8].

Références

[1] Dimou, A., M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, et R. Van de Walle (2014). RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Workshop on Linked Data on the Web (LDOW) collocated with WWW*.

[2] Gupta, S., P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, et M. Muslea (2012). Karma: A System for Mapping Structured Sources Into the Semantic Web. In *Extended Semantic Web Conference (ESWC), Poster&Demo*.

[3] Heyvaert, P., B. De Meester, A. Dimou, et R. Verborgh (2018). Declarative Rules for Linked Data Generation at Your Fingertips ! In *Extended Semantic Web Conference (ESWC), Poster&Demo*.

[4] Heyvaert, P., A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, et R. Van de Walle (2016). RMLEditor: a Graph-Based Mapping Editor for Linked Data Mappings. In *Extended Semantic Web Conference (ESWC)*.

[5] Junior, A. C., C. Debruyne, et D. O'Sullivan (2018). An Editor that Uses a Block Metaphor for Representing Semantic Mappings in Linked Data. In *Extended Semantic Web Conference (ESWC), Poster&Demo*.

[6] Lefrançois, M., A. Zimmermann, et N. Bakerally (2017). A SPARQL Extension For Generating RDF From Heterogeneous Formats. In *Extended Semantic Web Conference (ESWC)*.

Summary

The integration of structured data to the web of data is possible with *RDF Mappings*. However, for the development of these mappings it is necessary to be familiar with RDF, in addition to knowing perfectly the dataset. The small number of people satisfying these two conditions is an obstacle to the democratization of the Web of data. We present here a tool able to generate, semi-automatically, RDF mappings for structured datasets. The challenge is to automate the part of the integration process that requires the user to be familiar with RDF.

7. `https://data.opendatasoft.com`
8. `https://github.com/opendatasoft/semantic-bot`