Synthèse RDF, RDFS, SPARQL : Web de données

Yann Martin d'Escrienne, Yohann Tognetti

I. Qu'avons nous appris dans ce cours ?

Dans ce cours, nous avons appris à manipuler les différents langages qui nous ont été présentés en Web de données. Nous avons appris à créer des modèles RDF, de les décrire avec RDF Schema et enfin récupérer les différentes informations de notre modèle avec le langage de requête SPARQL.

Nous avons également eu une introduction à *CORESE*, un programme pour manipuler facilement les requêtes SPARQL sur un graphe de connaissance RDF (accompagné ou non d'une structuration par un fichier RDFS).

Nous avons également utilisé SPARQL sur DBpedia, une énorme base de données sous forme de graphes de connaissances formatées en RDF contenant des données extraites de wikipédia.

Nous avons également manipulé différentes syntaxes pour RDF et RDFS notamment *turtle* qui étend *N-triples* avec lequel nous avons créé nos modèles et nos descriptions de modèle mais aussi RDF/XML. Nous avons également appris à compléter nos modèles par des ontologies existantes comme FOAF.

II. Quels sont les principes de RDF, RDFS et SPARQL ?

RDF :

Comme son nom l'indique Resource Description Framework, RDF est un langage permettant de décrire des ressources provenant généralement du web.
L'utilisation de ce langage accorde la création de structure sur ces ressources et des liens entre elles sous forme de métadonnées. Cela peut s'avérer très pratique dans un ensemble de données qui n'est justement pas ou très peu structuré. Il autorise également par le biais de SPARQL de retrouver facilement et rapidement un ensemble d'informations sur la structure.

Il s'agit donc d'un langage abstrait utilisant un système de triplet. Un triplet contient donc un sujet, un prédicat et un objet. Un *sujet* est une URI correspondant à une ressource, un *objet* est une valeur littérale, un nombre ou bien même une autre URI et un *prédicat* est le lien qui unit le sujet et l'objet. Le prédicat est également sous la forme d'une URI.

Sa représentation peut être affichée sous forme d'un graphe orienté où les nœuds sont représentés par les sujets et les objets, et les arcs par les prédicats. L'URI correspondant sujet peut être existant ou non.

RDFS :

RDFS correspond à RDF Schema est un langage qui permet de créer des classes et des propriétés pour un modèle RDF. Il apporte une structuration supplémentaire et aide également à faire de l'ontologie sur le modèle.
RDFS possède une notion d'héritage que ce soit pour les classes ou bien les propriétés/prédicats qui permettent notamment l'inférence de type sur les ressources. Il apporte également un typage pour le sujet et l'objet des prédicats permettant là aussi de l'inférence de type.

SPARQL :

SPARQL (SPARQL Protocol and RDF Query Language*)* est un langage de requête pour RDF. Il aide à accéder à des ressources d'un modèle de données par l'aide de requêtes plus ou moins complexes selon les sujets, prédicats ou valeur des objets. Il permet également de créer des sous-graphes réunissant les ressources par différentes nouvelles propriétés ou valeurs à l'aide de fonctions comme COUNT, SUM ou AVERAGE.

III. Quels sont les éléments clefs du langage ?

RDF :

Comme dit précédemment, les éléments principaux du langage RDF sont les triplets composés du sujet du prédicat et de l'objet.

Lorsque les URI des membres des triplets se base sur un début d'URI commun, il est alors conseillé d'utiliser les "*prefix".* Les prefix sont des sortes de macro ou raccourcis pour une base d'URI et permettent donc de simplifier grandement la création et la compréhension d'un modèle RDF. Il est possible d'utiliser "*base"* qui se comporte alors comme un *prefix* par défaut pour le modèle.

Une fois une ressource créée, il est ensuite possible de lui ajouter plusieurs propriétés qui pointent ainsi vers différentes valeurs ou autres ressources. On retrouve alors notre système de triplet du graphe de connaissance. Il est tout à fait possible de créer des ressources ne comportant pas de propriétés et servant juste d'objet sous forme d'URI pour certains sujets de triplets.

Toutes les URI des ressources pointées par une ressource décrite par le modèle ne doivent pas forcément se trouver dans le fichier. Elles peuvent faire référence à un autre graphe de connaissance. Il faut alors utiliser d'autres syntaxes comme le *TriG* ou *N-Quads* pour créer des liens entre ces graphes. Mais cela peut aussi être une URI existant sur le web.

RDFS :

Pour RDFS, les deux mots-clefs principaux sont les *Class* et les *Property.* Comme dit plus haut, ils permettent de déclarer des classes et des propriétés. Les mots-clefs "*subClassOf*" et "*subPropertyOf*" permettent de créer l'héritage entre les classes et les propriétés.

Il est possible d'ajouter un libellé et un commentaire pour chaque classe et propriétés avec les mots-clefs "*label"* et "*comment"*. Cela afin d'améliorer la compréhension de celles-ci car leur nom n'est pas toujours explicite dans un modèle. Pour différencier les libellés et

commentaires selon la langue, il est permis d'utiliser “*@Langue”* en mettant le code de la langue désirée.

Enfin, une property peut posséder un “*domain”* et un “*range” tous* deux utilisés pour l’inférence de type et la structuration du modèle. Ils permettent de dire sur quelle classe porte un prédicat et vers quelle classe il se réfère.

IV. Quelles sont les bonnes pratiques de modélisation ?

Tout d’abord une des bonnes pratiques dans la création d’un modèle est d’utiliser “*base”* et de nombreux “*prefix”* cela afin d’améliorer la compréhension et la réalisation du modèle. Il faut généralement utiliser “*base”* pour les URI des ressources.

Dans un modèle RDF, il faut utiliser autant que possible des URI plutôt que des valeurs littérales ou nombres dans les objets des triplets. Cela ajoute d’autant plus de détails que la ressource pointée par l’URI est décrite. Si possible ces URI doivent provenir d’un autre graphe de connaissances exprimé en RDF.

Lorsque l’on utilise des nombres, littéraux… pour les objets des triplets, toujours typer les données à l’aide de *XSD*.

Il faut toujours essayer de compléter son vocabulaire par des ontologies existant en web de données, par exemple *FOAF* (Friend Of A Friend). Le vocabulaire et structure doit s'effectuer de préférence dans le fichier RDFS et non directement dans le modèle.

Il est également bienvenu de toujours compléter ses classes et propriétés d’une structuration RFDS par les “*label”* et les “*comment”* afin de nommer et décrire les éléments d’une manière claire.

Pour compléter l’utilisation des “label” et “comment”, il faut toujours donner plusieurs versions de ceux-ci dans différentes langues dont au moins l’anglais. Cela autorise alors la description des données dans le langage choisi.

Il faut privilégier au maximum l’inférence de type offert par RFDS. Il faut donc toujours ajouter des *domain* et des *range* aux *property* ainsi qu’utiliser l’héritage autant que possible pour créer des arbres de parenté sur nos classes et propriétés.