Mini-projet - de la modélisation à la visualisation

1 Présentation du mini-projet

L'objectif de ce mini-projet est de vous amener à manipuler les technologies du Web sémantique au travers de la réalisation d'une petite application sémantique. Il s'agira de collecter des données, de les modéliser en RDF(S), de les interroger avec SPARQL et enfin d'offrir quelques fonctionnalités simples à des utilisateurs finaux au travers d'une petite application Web. Pour rendre le projet plus intéressant et plus personnel, vous êtes libres de choisir les données et les fonctionnalités de votre choix. La section suivante précise les conditions à respecter. Pour préciser un peu ce qui est attendu, voici quelques exemples à titre d'illustration :

- collecte de données historiques de météo et production de différentes statistiques selon des critères choisis par l'utilisateur ;
- collecte de sites touristiques, éventuellement, avec des avis, et affichage sur une carte selon différents critères, plus possibilité de marquer des sites comme visités ;
- collecte de résultats sportifs et fonctions de comparaison des performances entre plusieurs équipes ;
- application pour faciliter la recherche de stage à partir d'une collection de stages passés. Etant donné le temps limité dont vous disposerez, il est préférable de viser d'abord des fonctionnalités minimales, quitte `à prévoir des fonctionnalités supplémentaires si le temps vous le permet.

2 Tâches à réaliser

Voici la liste des tâches à réaliser, avec pour chacune les conditions à vérifier :

1. **Collecte des données**. Les données collectées doivent être déjà structurées (ex., tableaux, listes, catégories) car les autres (ex., textes, images) demandent des techniques spécifiques pour en extraire du contenu sémantique. La collection de données doit être assez grande pour être intéressante, mais de taille raisonnable pour ce mini-projet, typiquement des milliers ou dizaines de milliers d'entités. L'essentiel des données ne doit pas être déjà en RDF, mais une partie peut l'être, en particulier pour enrichir les données.
2. **Modélisation en RDF(S)**. Les données collectées doivent être modélisées en RDF. Cela implique deux sous-tâches : (1) la conception d'une ontologie RDFS ou OWL définissant le vocabulaire employée pour la modélisation, et (2) un processus, autant que possible automatisé, permettant la production du RDF à partir des données collectées. L'automatisation de la traduction en RDF vous permettra de facilement intégrer de nouvelles données ou bien de réviser votre modélisation. Une phase manuelle de nettoyage des données avant traduction peut cependant s'avérer nécessaire (voir outil OpenRefine avec extension rdf-extension). La correction du RDF produit devra être vérifiée, par exemple en le chargeant dans un store RDF (voir outil Fuseki).
3. **Point d'accès SPARQL**. Il s'agit de lancer un point d'accès SPARQL, si possible accessible en ligne, sur vos données RDF, afin de permettre à quiconque d'interroger et explorer vos données (par exemple avec l'outil Sparklis). L'outil Apache/Jena Fuseki permet très simplement de réaliser cette tâche.
4. **Elaboration de requêtes SPARQL**. Il s'agit ici d'élaborer les requêtes SPARQL qui permettront de

réaliser les fonctionnalités de votre application. Si ces fonctionnalités prennent en compte des critères utilisateur, il s'agira de requêtes paramétrées, c'est-à-dire avec des éléments à remplir avant l'émission de la requête.

5. **Accès à des données liées**. Il s'agit ici d'élaborer des requêtes SPARQL qui permettront de lier vos données à des données externes du LOD, localisées par exemple dans DBpedia, OSM ou encore wikidata. Cela à un intérêt pour compléter la présentation de vos données avec les éléments supplémentaires géographiques, culturels, données associées…
6. **Application Web**. Il s'agit de programmer une petite application Web servant d'interface entre l'utilisateur et le point d'accès SPARQL, si possible avec une interface graphique interactive. Toutes les technologies sont autorisées, mais je recommande l'utilisation de HTML5+CSS+JavaScript et des bibliothèques graphiques js : D3.js, plotly.js. Javascript permet de faire des requêtes AJAX au point d'accès SPARQL et donc de récupérer les réponses à une requête SPARQL (le code peut être trouvé sur Internet). HTML5 et CSS permettent l'affichage et l'interaction avec l'utilisateur. Il est seulement demandé que l'application soit fonctionnelle, sans exigence sur le "design". L'essentiel est que ces fonctionnalités exploitent les données sémantiques et leur interrogation avec SPARQL.

Les éléments d'évaluation du mini-projet seront :

— une courte présentation de 10 minutes lors de la dernière séance de TP, présentant les choix effectués et l'état courant ;
— un rapport de 5 pages présentant vos données et votre application;
— un git avec l'intégralité de votre projet (rapport, données, codes, instructions)
— Si possible la mise en ligne de votre application ou, à défaut, les sources et les instructions pour la faire tourner en local.

3 Quelques outils utiles

Convertisseurs RDF : il existe un certain nombre de convertisseurs depuis différents formats vers RDF. Ils sont répertoriés à la page http://www.w3.org/wiki/ConverterToRdf. Je vous conseille le convertisseur easyrdf.

OpenRefine : http://openrefine.org/. OpenRefine (formerly Google Refine) is a powerful tool for working with messy data : cleaning it ; transforming it from one format into another ; and extending it with web services and external data. Il peut être utile pour nettoyer des données tabulaires et offre même le moyen de produire du RDF en définissant un template. + extension rdf

Fuseki (v1) : https://jena.apache.org/download/ Fuseki is a SPARQL server. It provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP. Client web par défaut à : http://127.0.0.1:3030/

Sparklis : http://www.irisa.fr/LIS/ferre/sparklis/. Il permet à un utilisateur d'explorer un point d'accès SPARQL sans connaître le langage SPARQL, ni le schéma des données. Il peut aussi être utilisé pour fabriquer de façon interactive des prototypes de requêtes SPARQL, tout en contrôlant qu'elles retournent les résultats attendus.

Tutoriels sur requêtes SPARQL en JavaScript : qui montre comment envoyer la requête SPARQL, récupérer les résultats et les afficher dans une page HTML. Par exemple :

- D3sparql.js http://biohackathon.org/d3sparql/,
- js-query https://linkedwiki.com/query/Piechart_about_datasets_in_LinkedWiki#Code ,
- jquery-sparql https://github.com/SPARQL-Environment/jquery-sparql,
- jquery bioportal https://github.com/ncbo/sparql-code-examples/blob/master/javascript/index.html ,
- jquery ex https://jsfiddle.net/shivkumarganesh/JQ4qy/,
- ex jquery https://stackoverflow.com/questions/47613030/error-for-exec-a-sparql-query-into-js-script