Projet Tourisme Sémantique

MISSAOUI Ahmed
BENATHMANE Ayoub

Contexte et motivation

Chacun d'entre nous préfère voyager, voyager veut dire découvrir de nouveaux endroits pays où ville, et pour cela nous aurons besoin d'un guide touristique contenant des informations sur un lieu donné. Ces informations peuvent être à la fois une présentation du pays du point de vue culturel, religieux, archéologique et historique, mais aussi des plans, itinéraires et cartes de ce pays.

Avec le développement technologique et surtout informatique, nous avons constaté qu'il y a de nouvelles formes de ces guides qui ont vu le jour, principalement des applications web (ex : TripAdvisor) où desktop (ex : Google Earth).

Nous notons le manque d'application dans le domaine de tourisme utilisant le web sémantique. Ce dernier permet la définition de la sémantique des données, d'inférer des nouvelles connaissances et aussi assure la liaison avec les bases de données structurées tel DBpedia. Pour bénéficier de ces avantages, nous avons choisi de développer une application sémantique dans le domaine de tourisme.

Sujet du projet et objectifs

Dans le cadre du mini projet de ce module, nous proposons la réalisation d'une application dont le sujet est intitulé : "**Guide de Voyage touristique**", qui s'appuie sur les standards du web : HTML, CSS, et les standards du web sémantique : RDF, SPARQL, SKOS et OWL.

L'application a pour objectif de permettre aux utilisateurs de pouvoir chercher pour une ville x, une brève description et la liste des monuments historiques, les vol en provenance ou à destination d'une ville et aussi lui suggérer des hôtels et des restaurants.

Architecture logicielle

Notre application se décompose en deux parties :

- une partie Backend exposant les données sémantique RDF avec une SPARQL Endpoint et stockées dans une Triple Store (Corese)
- une partie Front End: exposition des données RDF extraites à partir des requêtes effectuées sur le SPARQL EndPoint à travers des pages web.

Implémentation

Dans l'Implémentation de notre projet nous avons opter pour l'utilisation de données récupérées à partir de DBpedia pour la représentation des villes et des monuments. Ce choix est motivé par le fait de mettre en évidence l'aspect Linked Data une des possibilités du Web sémantique. De plus, nous avons récupéré un ensemble d'informations concernant les hôtels et les restaurants en requêtant TripAdvisor. Cette récupération a été réalisé en utilisant un crawler développé dans le cadre de ce projet. Ce crawler génère des fichers CSV. La troisième source de données étant des fichiers CSV que nous avons défini pour représenter les vols et les agences de location. Les fichiers CSV et les données récupérées à partir de DBpedia, constituent nos données d'entrée pour l'outil développé dans le cadre de ce projet. Cet outil génère notre modèle RDF sous le format n3 en tenant compte de notre ontologie.

Ontologie

Pour la construction de notre ontologie nous avons choisi d'utiliser OWL. Ce dernier permet de construire des ontologies sémantiquement plus expressives qu'avec RDFS. Car il propose de nouvelles types de propriétés (transitive,..) mais aussi des restrictions sur des propriétés lors de définition des classes.

Pour la documentation de notre ontologie nous avons choisi d'utiliser SKOS bénéficiant des propriétés tel que "skos:defintion"… Recommandé par M. CORBY, nous avons séparé la documentation de l'ontologie afin de ne pas avoir OWL et SKOS dans le même fichier et la liaison est assuré par des URIs.

Pour plus de sémantique nous avons défini des règles d'inférence en utilisant le langage Rule. Ces règles ont pour objectif de recommander des hôtels , des monuments , et des agence en se basant sur le type de vol. Par exemple si le vol est de type loisir vers une ville X alors on le recommande un ensemble de monuments de cette ville.

1. **Les classes OWL :**
   - **Ville** : *Représente l'objet ville, elle a une restriction sur la propriété "contientMonument" qui réfère sur la classe "Batiment" et elle est une sous classe de la classe "Emplacement"*
   - **Emplacement** : *représente l'objet emplacement, qui représente un lieu (ville ou pays) et elle est disjointe à la classe "Batiment"*
   - **Batiment** : *Représente l'objet bâtiment, elle a une restriction sur la propriété "dans" qui réfère sur la classe "Emplacement", et elle est disjointe à la classe "Emplacement"*
   - **Hotel** : *Représente l'objet hôtel, elle est une sous classe de la classe "Batiment"*
   - **Vol** : *Représente l'objet vol*
   - **VolAffaire** : *Représente l'objet vol de type affaire, elle est une sous classe de la classe "Vol"*
   - **VolLoisir** : *Représente l'objet vol de type loisir, elle est une sous classe de la classe "Vol"*
   - **AgenceLocation** : *Représente une Agence Location*
   - **AgenceLocationVoiture** : *Représente une Agence Location des voitures, elle est une sous classe de la classe "AgenceLocation"*
   - **Restaurant** : *Représente l'objet Restaurant, elle est disjointe à la classe "Monument" et une sous classe de la classe "Batiment"*
   - **Monument** : *Représente l'objet Monument, elle est une sous classe de la classe "Batiment"*
   - **Pays** : *Représente un pays*
   - **URL** : *Représente un lien pour designer une image ou un site web*

2. **Les propriétés OWL :**
   - **dans :** *propriété de type transitive, permet de designer si un "Bâtiment" (hotel, monument, restaurant ou agence) situe dans une "Ville", et si cette ville est dans un "Pays" alors le raisonneur OWL infère automatiquement qu'il se situe dans ce "Pays"*
   - **siteWeb** : *propriété de type URL*
   - **imageUrl** : *propriété de type URL*
   - **contientMonument** : *propriété de type ressource, indique qu'un "Emplacement" contient un "Monument", elle est inverse à la propriété "estContenueDans"*

- **estContenueDans** : *propriété de type ressource, indique qu'un "Monument" se situe dans un "Emplacement", elle est inverse à la propriété "contientMonument "*
- **destination** : *propriété de type ressource, indique qu'un "Vol" à pour destination une "Ville"*
- **depart** *: propriété de type ressource, indique qu'un "Vol" à pour départ une "Ville"*
- **recommenderMonument** : *propriété de type ressource, indique que pour un "Vol" à destination d'une ville, on recommande une ensemble de "Monument"*
- **recommenderHotel** : *propriété de type ressource, indique que pour un "Vol" à destination d'une ville, on recommande une ensemble de "Hotel"*
- **recommenderAgence** : *propriété de type ressource, indique que pour un "Vol" à destination d'une ville, on recommande une ensemble de "Agence" de location*
- **nom, description, departement, codePostale, localisation, adresse, telephone, note** : *Propriétés de type DatatypeProperty.*

Scénario d'utilisation

1. L'utilisateur se connecte à l'application, et fait une recherche
2. Le page envoi la demande au serveur SPARQL
3. Le serveur interroge DBpedia pour récupérer des informations sur les villes et les monuments dont il ne les connaît pas en temps réel sinon il interroge nos données RDF déjà existantes.
4. Envoi du résultat de la Requête Sparql au serveur web
5. La page représente le résultat de la recherche
6. A partir d'une recherche sur DBpedia, nous pouvons aussi enrichir notre graphe RDF local pour optimiser les recherches, et aussi utiliser l'application en mode hors connexion.