Yana RAGOZINA
Thomas PAUL
SI5 IA-ID

Data Lifting

Ingénierie des connaissances

1/ Introduction

L'objectif de ce projet est de mettre en lien des données provenant de différentes sources. Les données proviennent d'une API et d'un fichier CSV converti en un fichier RDF (Resource Description Framework). Afin de mettre en lien ces deux sources de données, un micro-service SPARQL a été mis en place. Son architecture est la suivante : le client commence par faire une requête SPARQL au micro-service. Le micro-service va ensuite interroger un endpoint d'une API, récupérer sa réponse, coupler les données RDF avec la réponse en fonction de la requête faites par le client, et enfin, renvoyer la réponse au client.

2/ Cas d'usage

Pour réaliser ce projet, nous proposons à un utilisateur du micro-service de découvrir des recettes de cuisine à partir d'un mot clé qu'il peut choisir ("dessert" par exemple), et de voir les ingrédients disponibles par recette en vente dans les chaines des magasins Walmart. Pour un même ingrédient, plusieurs produits de différentes marques peuvent être proposés afin de s'adapter aux préférences du client. De cette manière, l'utilisateur pourra découvrir les produits disponibles dans la chaîne Walmart par rapport à la (aux) recette(s) de son choix.

3/ API et fichier CSV sélectionné

Le micro-service récupère la liste des recettes depuis l'API Edamam [1]. Elle propose un endpoint où nous pouvons passer en paramètre un mot clé (query) et l'API nous fournira en réponse une liste de recettes contenant ce mot clé, avec les ingrédients et les étapes de préparation. L'authentification est relativement simple, il suffit de se créer un compte sur le site internet de Edamam pour obtenir l'accès à un identifiant et une clé unique à fournir lors de chaque requête.

Le fichier CSV (*csvw\walmart_products.csv*) contient les données correspondantes à une liste de produits vendus par la chaîne de magasins Walmart [2]. Le fichier répertorie les produits de plusieurs magasins, chaque magasin ne possède pas tous les produits de ce fichier. Il était indispensable de faire un pré-traitement des données car le fichier CSV est très volumineux et il n'était pas nécessaire de garder toutes les colonnes du fichier. Nous avons gardé l'identifiant du produit, son nom, sa marque, son prix et son poids. De même, dans le fichier CSV initial il existe de nombreuses versions référençant souvent le même produit. Ainsi, nous avons également fait le choix de garder un produit sur deux. Dans le fichier RDF (*csvw\walmart_products.ttl)*, nous gardons uniquement les identifiants et les

noms des produits car pour le moment, nous n'avons pas besoin de plus d'attributs et nous souhaitons réduire au plus possible la taille de ce fichier.

Dans un travail futur, nous pourrons inclure davantage de données comme celle du prix des produits pour avoir plus d'informations sur chaque recette.

4/ Vocabulaires sélectionnés ou crées

Le choix du vocabulaire sélectionné se base sur la classe Recipe désignant une recette de cuisine, fournie par le vocabulaire de Schema.org (Thing > CreativeWork > HowTo > Recipe). Ainsi, avec chaque requête passée à l'API avec SPARQL micro-service, nous cherchons à récupérer une recette (a Recipe) avec les ingrédients nécessaires (propriété *recipeIngredient* de la classe Recipe) ainsi que le nom de la recette (propriété *name* de la classe Thing), identiquement à l'exemple ci-dessous.

```
@prefix schema: <http://schema.org/>

CONSTRUCT {
    ?recipe a schema:Recipe .
    ?recipe schema:recipeIngredient ?food .
    ?recipe schema:name ?recipeName ;

} WHERE {
    [] api:recipe ?recipe .
    ?recipe api:label ?recipeName .
    ?recipe api:ingredients [ api:food ?food ] .
}
```

De plus, nous utilisons le même vocabulaire de Schema.org (la classe Recipe et ses propriétés) pour la traduction du fichier CSV contenant les produits dans la chaîne de magasins Walmart. Ainsi, le fichier RDF final contient, pour chaque produit, son URI, la propriété désignant le nom du produit (le prédicat se base sur la propriété *name* de la classe Thing) et la valeur du nom du produit (valeur littérale), tel qu'illustré ci-dessous.

```
<http://example.org/walmart/products/0> <http://schema.org/name> "Roasted Red Pepper Hummus" .

<http://example.org/walmart/products/2> <http://schema.org/name> "Classic Hummus" .
```

Yana RAGOZINA
Thomas PAUL
SI5 IA-ID

5/ Conclusion

Au travers de la création du micro-service SPARQL, nous faisons appel à deux sources de données : un fichier RDF contenant tous les produits vendus dans les magasins Walmart et une API de recettes de cuisine qui contient la liste des ingrédients nécessaires à la réalisation de chaque recette. La fusion de ces deux sources de données par le micro-service permet à l'utilisateur de trouver chez Walmart les ingrédients dont il a besoin pour la réalisation de la recette de son choix.

6/ Références

[1] API de recettes de cuisines : https://www.edamam.com/

[2] Fichier CSV contenant les produits vendus par Walmart : https://www.kaggle.com/datasets/thedevastator/product-prices-and-sizes-from-walmart-grocery