

Foodle

Rapport de Projet Web Sémantique

Auteurs:

H4142

Lou Bezzina, Daniel Blasko, Nolwenn Deschand, Laetitia Dodo,
Emilien Marchet, Damien-Joseph Rispal, David-Marcus Toma

Sommaire

I/ Introduction	2
II/ Architecture	2
1) Technologies choisies	2
2) Structure du projet	2
III/ Fonctionnalités	3
1) Spécification des fonctionnalités	3
2) Implémentation technique	7
IV/ Difficultés rencontrées	8
V/ Conclusion	9
V/ Annexes	10

I/ Introduction

L'objectif de ce projet est de créer un moteur de recherche spécialisé, utilisant DBpedia et les principes du Web sémantique.

Le Web sémantique est une extension du Web qui permet la recherche par concept. Cela est rendu possible par le langage RDF, avec lequel on peut décrire des connaissances en annotant des pages Web. Les données ainsi annotées sont stockées dans une base de connaissances (ici DBpedia) qui lie les concepts entre eux. De cette manière, le Web sémantique permet aux machines de filtrer et utiliser l'information sans intervention humaine.

Nous avons choisi de nous orienter vers le domaine de la nourriture. Notre moteur de recherche propose à l'utilisateur de rechercher un plat, les plats utilisant un certain ingrédient, les plats d'un certain type ou provenant d'un certain pays. À tout moment de la recherche, il est possible de cliquer sur le nom d'un plat, d'un pays ou d'un ingrédient pour voir la page présentant le plat, le pays ou les plats contenant cet ingrédient.

II/ Architecture

1) Technologies choisies

Afin de réaliser notre moteur de recherche, nous avons décidé de construire une interface web en HTML/CSS, par familiarité avec ces technologies. Le style des différentes pages s'appuie pour certains éléments sur des classes Bootstrap ce qui a permis d'arriver rapidement à un résultat visuel élégant.

Les données sur lesquelles reposent les pages de notre moteur de recherche sont récupérées sous un format JSON depuis une requête HTTP qui interroge la base de données DBpedia avec une requête SPARQL, directement depuis le client Web.

Le lien entre les données et les pages web est fait par différents scripts JavaScript qui ont pour rôle de générer la requête SPARQL en fonction de la recherche faite sur la page, de l'envoyer par HTTP et de recevoir la réponse, pour l'intégrer dans le code HTML après traitements.

Notre environnement de travail durant ce projet a été composé de :

- ClickUp pour tout ce qui concerne gestion de projet, répartition des tâches et notes
- IntelliJ / VS Code pour le développement
- GitHub pour la gestion des versions

2) Structure du projet

Afin de structurer notre projet au mieux, nous avons choisi de coder chaque fonctionnalité par une page HTML spécifique. Ainsi, en plus de la page d'accueil `index.html`, nous avons une page pour les plats, les pays et pour le résultat d'une recherche. Ces pages sont stockées dans le dossier `src`.

À chaque page on peut associer un fichier CSS qui gère le style de la page en question, ces fichiers se trouvent dans le dossier `style`. Ce dossier contient également des ressources images et des polices de caractère.

Nous avons aussi un dossier `components` qui contient des composants HTML réutilisables, comme `header-bar.html` que nous retrouvons en haut de chaque page du moteur de recherche.

Enfin, le dossier `js` contient les scripts JavaScript nécessaires à chaque page ainsi que les scripts utilisés sur tout le moteur de recherche.

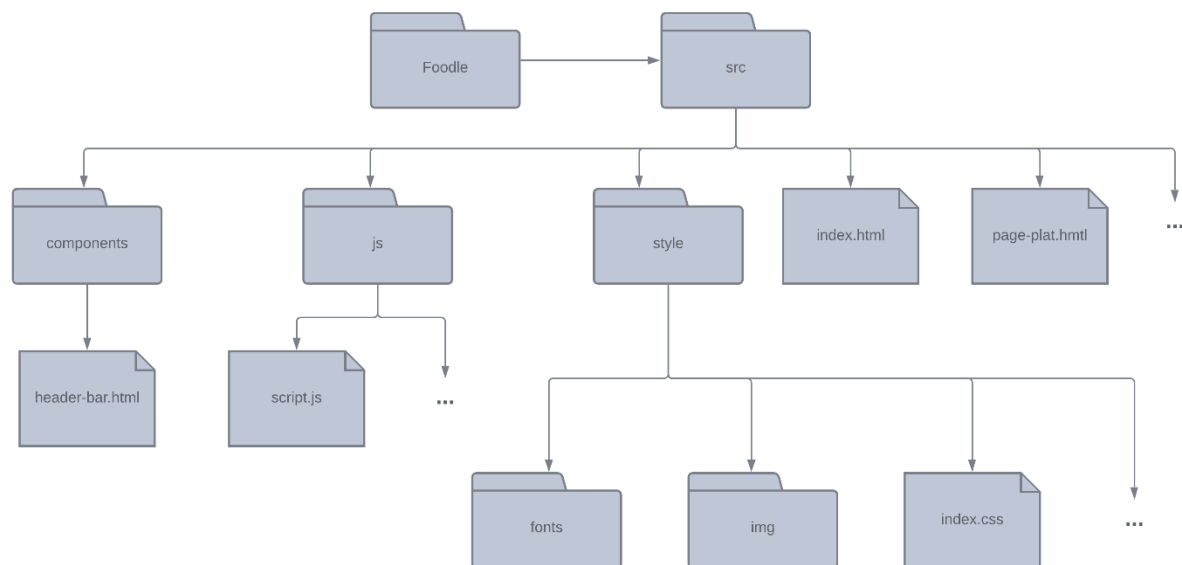


Figure 1: Arborescence des fichiers

III/ Fonctionnalités

1) Spécification des fonctionnalités

A. Page d'accueil

Search by: (1) Name (2) Ingredient (3) Type (4) Country

Dish name... Filter by country... (5) ▼

Search 🔍 (6)

Figure 2 : Page d'accueil - barre de recherche

La page d'accueil est la page principale de recherche. Afin d'améliorer la précision des résultats et la variété des requêtes, la recherche est possible selon différents critères : le nom du plat ⁽¹⁾, les ingrédients qui le composent ⁽²⁾, ou son type ⁽³⁾.

Ces critères sont croissables avec un filtre supplémentaire, décrivant l'origine du plat ⁽⁵⁾. Ce filtre est sélectionnable dans une liste déroulante des pays et zones géographiques référencés par DBpedia, qui peut aussi faire office d'autocomplétion.

Il est également possible de directement rechercher un pays pour trouver tous ses plats caractéristiques ⁽⁴⁾.

Pour lancer la recherche, on peut cliquer sur le bouton *Search* ⁽⁶⁾, tout comme appuyer sur la touche entrée ↵ du clavier.

Les recherches peuvent être effectuées dans différentes langues, que ça soit pour les ingrédients, le type ou le nom du plat. L'objectif de cette fonctionnalité est de ne pas exclure les plats typiques et régionaux qui parfois n'ont pas de label anglais (par exemple la galette des rois). Cependant, cette fonctionnalité est dépendante des liens implémentés par DBpedia : si le nom d'un ingrédient n'a pas d'équivalence en slovaque, notre moteur de recherche ne pourra pas le trouver.



Figure 3: Page d'accueil - highlighted food

Pour agrémenter la page d'accueil, à chaque nouveau chargement, un plat est mis en avant : le *Highlighted Food*. Un plat aléatoire est sélectionné à chaque fois, et son image ainsi qu'une description y sont associées. De quoi ouvrir les portes d'un monde culinaire inconnu ! Pour plus d'informations sur un plat du jour, il est possible de cliquer sur son nom pour accéder à la page détail du plat.

B. Header

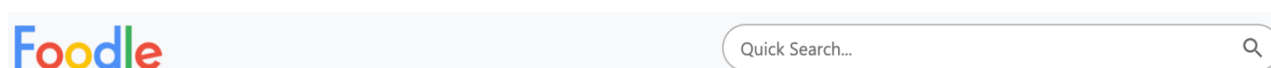


Figure 4: Barre de navigation

Sur toutes les autres pages de l'application, il est possible de lancer une recherche rapide. Cette recherche correspond à une recherche sur le nom du plat uniquement. Pour revenir à la page d'accueil, il suffit de cliquer sur le logo *Foodle*.

C. Page résultats de recherche

Results for "Caramel"

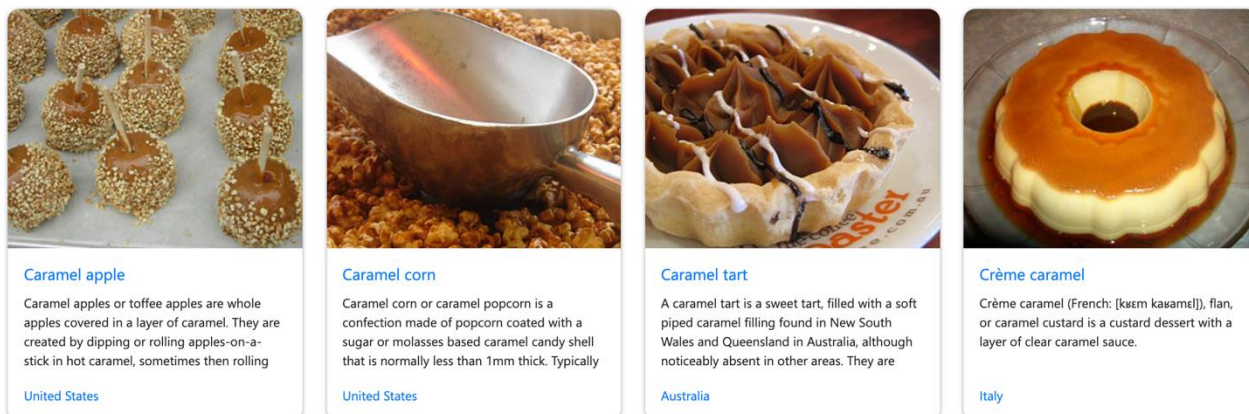


Figure 5: Résultats de recherche

Lorsqu'une recherche est effectuée, les résultats sont affichés sous forme d'une galerie de cartes permettant d'avoir un aperçu du plat. Il est possible de cliquer sur chacune de ces cartes pour accéder à la page du plat. Les cartes contiennent une image, la provenance, et le début de la description.

Une roue de chargement indique l'état de la recherche. S'il n'existe pas de plat correspondant aux critères choisis, aucune carte ne s'affiche.

D. Page détail plat

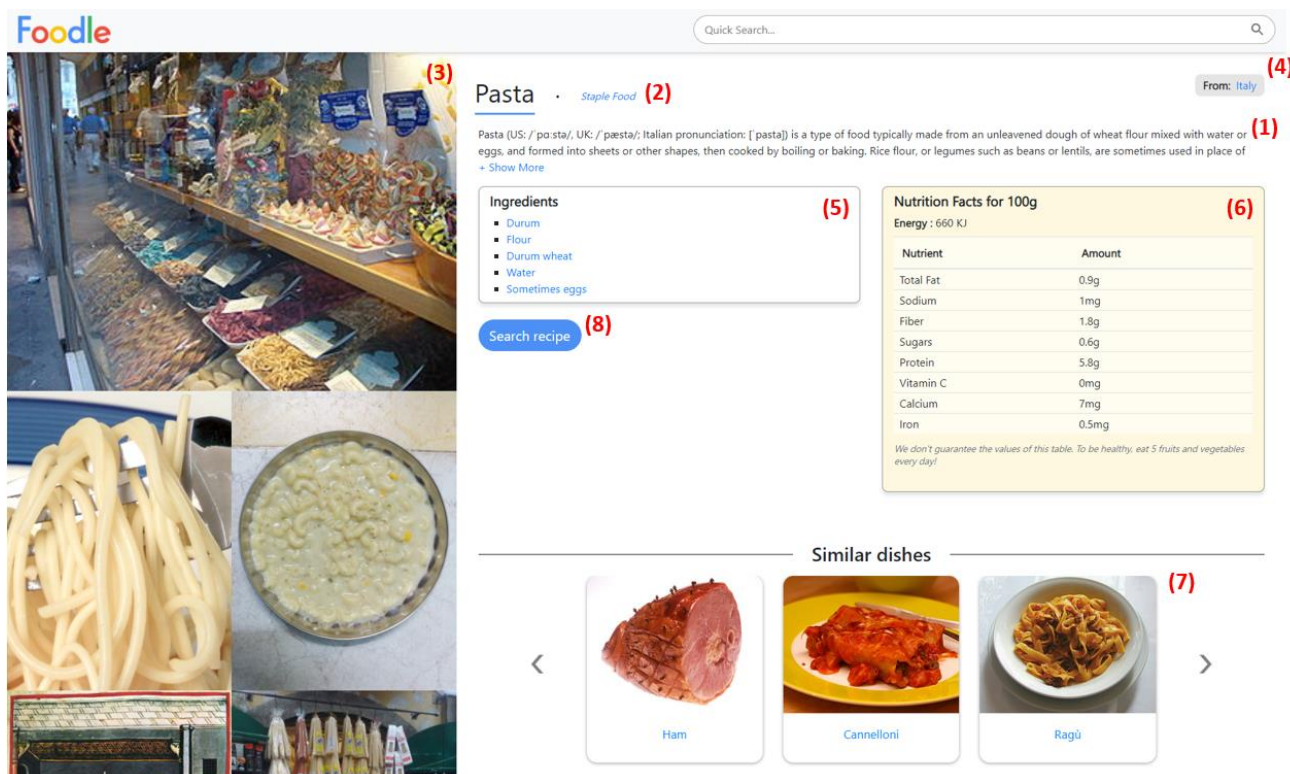


Figure 6: Page détail d'un plat

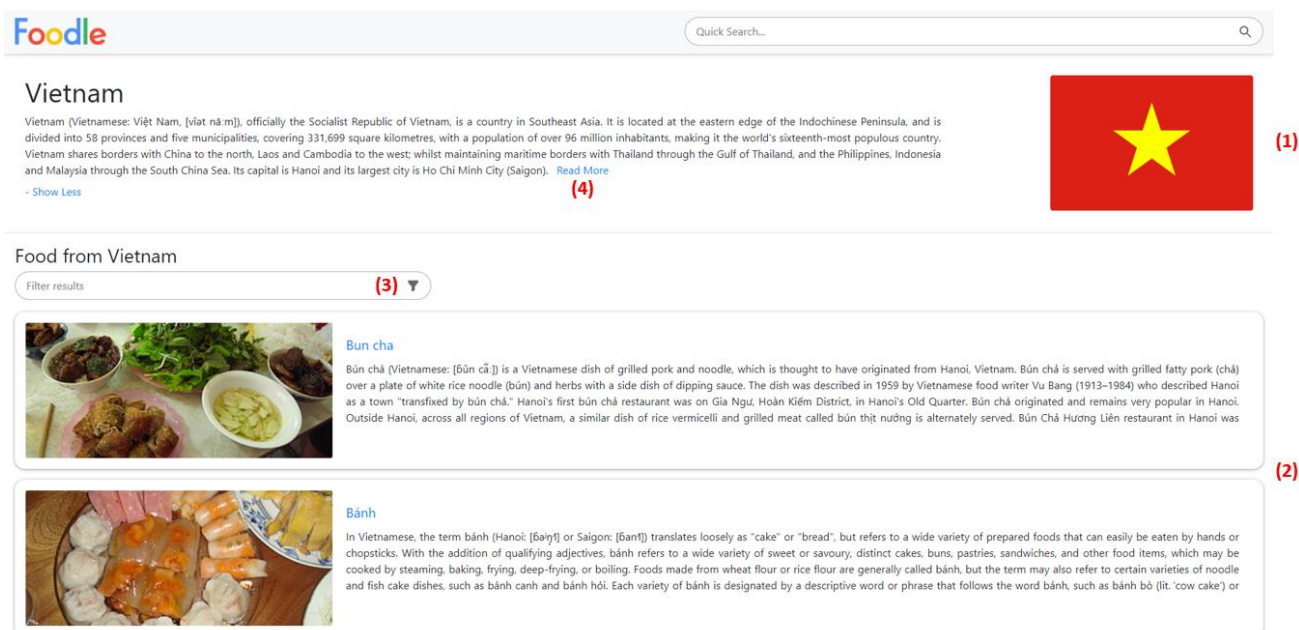
Sur notre application, chaque plat a sa page dédiée, permettant d'afficher les détails de ce dernier. Si les informations sont disponibles, on affiche la description ⁽¹⁾, le type ⁽²⁾, une galerie d'images ⁽³⁾, la provenance (pays / région) ⁽⁴⁾, la liste des ingrédients ⁽⁵⁾, les valeurs nutritionnelles ⁽⁵⁾ et les plats similaires ou associés ⁽⁶⁾.

Cliquer sur le nom d'un ingrédient, sur la provenance du plat ou sur son type permet de lancer une recherche de plats comportant cet ingrédient, provenant du même pays ou du même type respectivement.

On peut également cliquer sur les plats associés pour obtenir plus de détails sur ces derniers. Si seulement trois plats similaires sont affichés par soucis de place, il est possible de naviguer pour en afficher plus grâce aux boutons sur les côtés, et les plats affichés défilent dynamiquement sur la page.

Le bouton *Search Recipe* ⁽⁸⁾ permet de rechercher directement une recette pour ce plat sur le site www.allrecipes.com.

D. Page détail pays



The screenshot shows the 'Foodle' website interface. At the top, there's a search bar and a navigation menu. The main content area is titled 'Vietnam' and features a description of the country, a flag, and a list of food items. The first food item is 'Bun cha', which is described as a Vietnamese dish of grilled pork and noodle. The second food item is 'Banh', which is described as a variety of prepared foods that can easily be eaten by hands or chopsticks. The page also includes a 'Filter results' button and a 'Read More' link.

Figure 7: Page détail d'un pays

Lorsqu'on effectue une recherche par zone géographique ou qu'on clique sur une provenance depuis un des pages précédentes, on arrive sur la page détail du pays.

Cette page comprend des infos sur le pays (drapeau, description) ⁽¹⁾, ainsi que les plats provenant du pays ⁽²⁾. Ces plats peuvent être filtrés par nom grâce à la barre de recherche ⁽³⁾.

Il est également possible d'être redirigé vers la page Wikipédia du pays en cliquant sur *Read more* ⁽⁴⁾ dans la description.

2) Implémentation technique

Comme expliqué dans la [section II/ 1](#), le fonctionnement typique de l'application est le suivant :

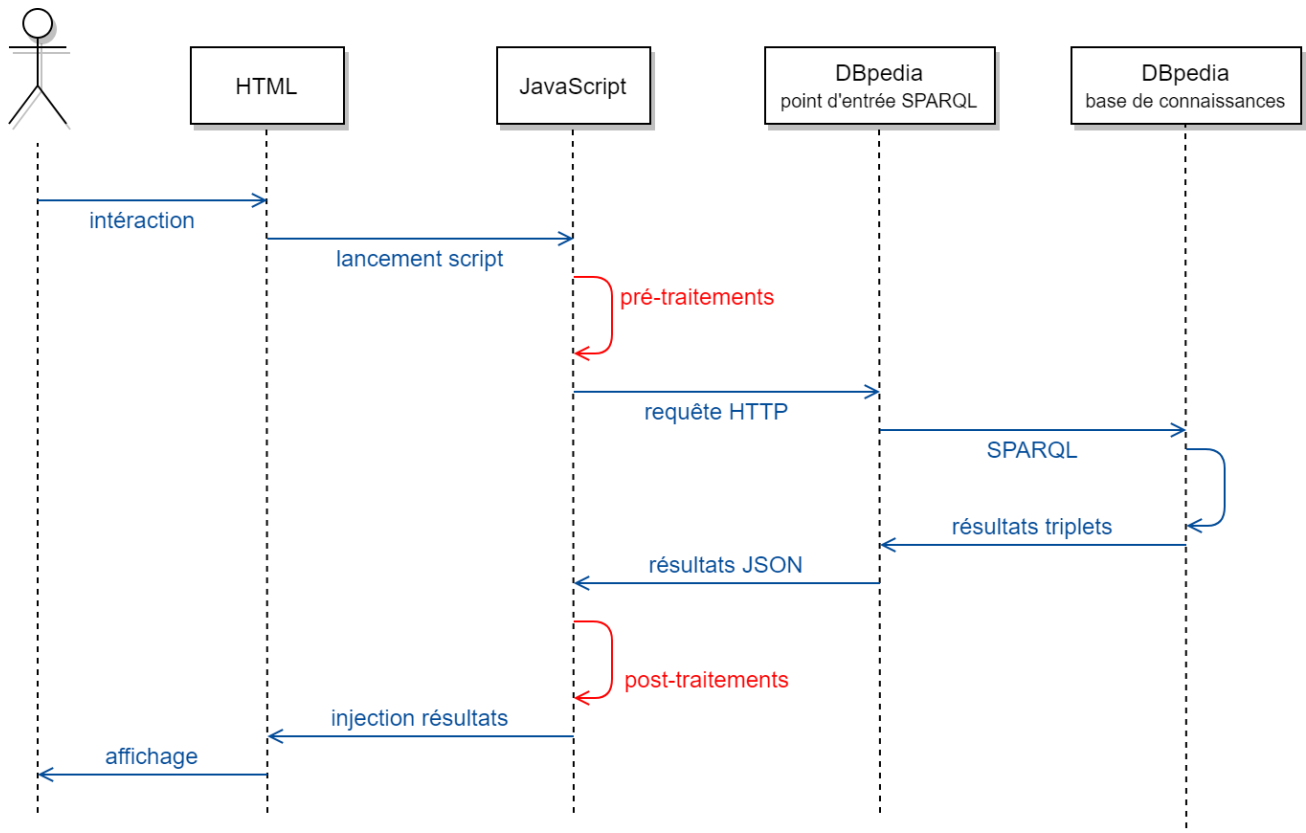


Figure 8: Flow des données dans l'application

A. Traitements pré-requêtes

Les **pré-traitements** faits par JavaScript (avant d'envoyer une requête vers DBpedia) consistent principalement à générer des requêtes SPARQL selon le contexte de l'application (communiqué dans les paramètres de l'URL des pages).

Requêtes de recherche de plats :

Pour les recherches, nous avons mis en place un système de génération de requêtes dynamique, car les résultats renvoyés ont souvent le même format pour les différentes recherches (image, nom plat, description, pays), et répondent à des filtres communs (conditions sur les langues).

La fonction `queryBuilder()` permet de construire un string de requête générique SPARQL pour effectuer une recherche de plats, en injectant des triplets et des filtres donnés en paramètre.

De cette manière, il est facile de réutiliser la même structure de requête pour les recherches selon le nom du plat, ses ingrédients ou son type. Le filtre par pays d'origine devient une fonctionnalité triviale aussi car il suffit de rajouter un filtre de plus dans la liste des filtres donnés en paramètre à `queryBuilder()`.

Requêtes d'obtention de détails sur le plat :

Pour le détail de plats, les requêtes ont toujours la même structure : pour chaque type d'information sur le plat (type, origine, ingrédients, etc), on cherche les sujets associés à des prédicats tels que ceux-ci soient présents dans une liste donnée dans un filtre.

Par exemple, pour ingrédients, on cherche les sujets associés aux prédicats (*dbo:ingredient*, *dbo:ingredientName*, *dbp:mainIngredient*, *dbp:minorIngredient*).

Autres requêtes :

Les autres requêtes sont écrites au cas par cas, hardcodées dans les scripts JS.

B. Traitements post-requêtes

Les **post-traitements** faits par JavaScript (après avoir reçu la réponse de DBpedia) consistent principalement à assurer la cohérence et le bon formatage des données renvoyées avant de les afficher. Ces problèmes de cohérence sont détaillés dans la [section IV/](#).

IV/ Difficultés rencontrées

Nous avons fait face à quelques problèmes dans ce projet, principalement à cause de DBpedia et de son système de classification et formatage, mais aussi par rapport aux fonctionnalités de recherche.

A. Pluralité et incomplétude des prédicats

La classification DBpedia est parfois incohérente, et une même information peut être stockée sous différents prédicats selon la ressource à laquelle on s'intéresse. Pour pallier ce problème, il a fallu chercher tous les prédicats concernés, puis les sélectionner selon leur priorité d'intérêt (si prédicat1 existe, l'utiliser, sinon si prédicat2 existe, l'utiliser, sinon combiner prédicat3 et prédicat4...).

B. Multiplicité inattendue des sujets des prédicats

De plus, un même prédicat peut avoir plusieurs valeurs différentes dans une même page, et ces valeurs peuvent ne pas avoir de lien logique (par exemple : l'attribut *dbo:abstract* de la page France sur DBpedia a trois valeurs, dont la première est une description de la ville brésilienne Franca).

Ceci implique aussi qu'un plat peut avoir plusieurs pays d'origine, et dans ce cas il faut bien tous les inclure dans la page du plat.

C. Formatage des ingrédients

Nous avons également eu quelques soucis avec le formatage des informations sur DBpedia, notamment pour les listes d'ingrédients.

Les ingrédients sont décrits avec des prédicats différents (pas toujours tous présents), tantôt des ressources, tantôt des littéraux listant les ingrédients avec un string.

Les informations se complètent d'un prédicat à l'autre, donc il faut tout prendre en compte et détecter les doublons – en uniformisant leur formatage et en les mettant dans un Set.

De plus, les séparateurs utilisés dans les littéraux ne sont pas cohérents : parfois des virgules, parfois des points-virgules, parfois avec des éléments vides.

Les ingrédients eux-mêmes peuvent être mal formatés, avec des espaces manquants entre les mots, ou des mots parasites comme « or » ou « sometimes » qui sont restés dans la liste.

`'Yeastdough ;chocolate, ,jam, orchese'`

Figure 9: Exemple de mauvais formatage des ingrédients

D. Images cassées et doublons

Nous nous sommes rendus compte que certaines images *foaf:depiction* avaient une URI irrécupérable, mais que la même URI avec le paramètre supplémentaire *?width=300* (ou toute autre valeur) fonctionnait très bien, et que c'est d'ailleurs cette URI qui était utilisée dans *dbo:thumbnail*. Pour récupérer les images d'un plat il a donc fallu utiliser les images issues des deux prédicats pour anticiper cette erreur.

Ceci a donc généré des doublons pour d'autres plats qu'il a fallu corriger en comparant les URIs pour voir si l'une (*depiction*, sans le paramètre *width*) était contenue dans l'autre (*thumbnail*, avec le paramètre *width*).

Pour les autres images irrécupérables, nous avons fait en sorte d'envoyer une requête HTTP pour tester leur validité, et de les remplacer par une image « not found » si on obtenait une erreur 404.

E. Regex de recherche

Les résultats d'une recherche pour « *tea* » peuvent inclure des mots plus globaux comme « *gâteau* » ou « *steak* ». De même, lorsqu'on cherche les plats de la région « *Aden* », on peut retrouver des plats de la région « *Baden-Wurtemberg* ». Nous avons essayé de corriger ces problèmes avec des patterns regex plus poussés, mais ces cas restent assez rares.

V/ Conclusion

Au cours de ce projet, nous avons réalisé à quel point le Web sémantique peut être une ressource utile. En effet l'injection de sens dans les pages web pour les machines permet l'automatisation du traitement des données et plus largement l'enrichissement des bases de données web en réseaux de données structurés et réutilisables. Ce deuxième point est garanti par l'interopérabilité des données et de leur ontologie que contrôle les normes du W3C (URI, RDF, OWL). Cet outil semble idéal pour permettre la mutualisation des données à l'échelle mondiale et améliorer la pertinence des résultats de recherche web.

Cependant, les données du web sémantique souffrent d'un manque cruel d'uniformisation et d'harmonisation, par le fait que tout le monde peut contribuer à la base de connaissances. Cela résulte en des informations dispersées sur une multitude de prédicats, ou au formatage non normé.

Ainsi, avant qu'une machine puisse exploiter automatiquement les diverses connaissances du web sémantique, il faut qu'un humain fasse une analyse des triplestores pour un domaine expert donné (pour comprendre sa structure), et d'instruire la machine en fonction des incohérences observées.

Pour remédier à ces problèmes, nous pouvons envisager une norme plus robuste pour obliger les contributeurs à respecter des règles de formatage, ou de vérifier si la création d'un nouveau prédicat est vraiment nécessaire (dans le cas où un autre prédicat au nom similaire existe déjà dans le graph).

Pour s'assurer du respect de ces normes, nous pouvons imaginer par exemple une IA qui vérifie les nouveaux triplets injectés dans le graph et les rejette si des problèmes sont détectés, en donnant des suggestions d'améliorations (« utiliser ce prédicat déjà existant plutôt que celui-là »).

VI/ Annexes

Voici quelques requêtes SPARQL que nous avons utilisé pour interroger DBpedia :

A. Récupérer les pays qui ont un plat

```
SELECT DISTINCT ?label WHERE {
  ?Country a dbo:Country ; rdfs:label ?label.
  ?Food a dbo:Food ; dbo:country ?Country.
  FILTER(
    (lang(?label) = "" || langMatches(lang(?label), "en")) &&
    !contains(?label,"cuisine")
  )
}
ORDER BY ASC (?label)
```

B. Récupérer les plats similaires d'un plat donné

```
SELECT ?nom, SAMPLE(?Thumbnail) AS ?thumbnail WHERE {
  ?food a dbo:Food ; rdfs:label '{1}'@en ; ?predicat ?sujet. ?sujet a dbo:Food.
  ?sujet rdfs:label ?nom ; dbo:thumbnail ?Thumbnail.
  FILTER(!isLiteral(?sujet) || lang(?sujet) = '' || langMatches(lang(?sujet), 'en')).
  FILTER(lang(?nom) = '' || langMatches(lang(?nom), 'en')).
  FILTER(?predicat IN (owl:sameAs, dbo:hasVariant,
    dbp:variations,dbp:wikiPageWikiLink, dbp:similarDish)).
}
GROUP BY ?nom
```

Remplacer {1} par le label du plat donné

C. Fonction qui génère une requête de recherche générique

```
function queryBuilder(triplets, filters, limit) {

    let query = "";

    // Header
    query += "SELECT DISTINCT (SAMPLE(?Food) AS ?food) ?label (SAMPLE(?CountryName)
    AS ?countryName) (SAMPLE(?Thumbnail) AS ?thumbnail) (SAMPLE(?Abstract) AS ?abstract)
    WHERE { \n" +
    "?Food a dbo:Food ; rdfs:label ?label ; dbo:country ?country ;
    dbo:thumbnail ?Thumbnail ; dbo:abstract ?Abstract .\n" +
    "?country rdfs:label ?CountryName .";

    // Extra triplets
    for (let i = 0; i < triplets.length; i++) {
        query += triplets[i] + "\n";
    }

    // Filters
    query += "FILTER(langMatches(lang(?Abstract), 'en') || lang(?Abstract) = '') \n" +
    "FILTER(langMatches(lang(?CountryName), 'en') || lang(?CountryName) = '').";

    // Extra filters
    for (let i = 0; i < filters.length; i++) {
        query += filters[i] + "\n";
    }

    // More filters
    query += "FILTER(langMatches(lang(?label), 'en') || lang(?label) = '')";

    // Footer
    query += "} \n" +
    "GROUP BY ?label \n" +
    "ORDER BY ASC(?label) \n" +
    "LIMIT " + limit;

    return query;
}
```