Projet Web sémantique (4IF-WS)
Rapport de synthèse

Elise BACHET, Andy GONZALES, Louis LABORY, Jason LAVAL, William MICHAUD, Lou REINA-KUTZINGER

Abstract—Ce document présente les détails de l'architecture, les difficultés rencontrées (alignement d'ontologies, limites des LLM) ainsi que des réflexions sur le Web Sémantique. Il présente aussi des retours d'analyse de données : visualisation de graphes et connaissances / interprétation extraites etc.

Keywords—*Web sémantique, données, LLM, projet, logique et réflexions, backend, LOD, SPARQL, RDF*

Contents

1 Problématique et positionnement du projet **1**
1.1 Problématique initiale et limites de la recherche classique . . . 1
1.2 Objectifs du projet . . . 1
2 Architecture du Projet **2**
2.1 Couche de données . . . 2
Importation des données • Données récupérées
2.2 Couche Backend et logique métier . . . 2
2.3 Tests et Validation . . . 2
3 Difficultés rencontrées **3**
3.1 Les données.. . . . 3
3.2 Organisation du projet . . . 3
3.3 Trouver un sujet . . . 3
3.4 DBpedia access point . . . 3
3.5 Changement de modèle de langage en cours de projet . . . 3
4 Réflexions sur le Web Sémantique **3**
5 Interpretations extraites et visualisations **3**
5.1 Tri des cocktails par vibes . . . 3
5.2 Trouver des cocktails à partir d'ingrédients . . . 4
5.3 Cocktails Similaires . . . 4
5.4 Playlist et ingrédients . . . 4
5.5 Sous-graph des cocktails et ingrédients . . . 4
5.6 Optimiseur de Bar . . . 5
6 Conclusion synthétique **5**
A Annexes **5**
A.1 Liste complète des requêtes SPARQL . . . 5
A.2 Organisation du Travail . . . 5

1. Problématique et positionnement du projet

1.1. Problématique initiale et limites de la recherche classique

Dans le domaine de la mixologie, l'information est abondante, mais rarement exploitable de manière intelligente. Les moteurs de recherche et bases encyclopédiques permettent de trouver des recettes de cocktails, mais se limitent à une consultation isolée, sans réelle compréhension des relations entre cocktails, ingrédients ou styles.

Or, les besoins réels des utilisateurs vont bien au-delà de la simple lecture d'une fiche cocktail. Ils cherchent à répondre à des questions plus structurelles, telles que :

- *Quels cocktails puis-je réellement préparer avec les ingrédients dont je dispose ?*
- *Quels cocktails sont proches entre eux, et pour quelles raisons ?*
- *Quels ingrédients sont centraux ou structurants dans l'univers des cocktails ?*
- *Comment optimiser un bar pour maximiser la diversité de cocktails réalisables avec un nombre minimal d'ingrédients?*

Ces questions ne relèvent pas d'une recherche textuelle classique. Elles nécessitent une représentation explicite des entités et de leurs relations, ainsi qu'une capacité à analyser l'ensemble de ces relations de manière globale. Les approches traditionnelles atteignent rapidement leurs limites dès lors qu'il s'agit de raisonner, comparer ou optimiser à l'échelle d'un ensemble de cocktails.

1.2. Objectifs du projet

Le projet *MarmiTonic* a pour objectif de démontrer comment les technologies du Web sémantique peuvent être utilisées pour transformer des données ouvertes hétérogènes en connaissances exploitables.

Plus précisément, les objectifs du projet sont les suivants :

- Exploiter des données LOD (*Linked Open Data*), en particulier DBpedia, à l'aide de requêtes SPARQL pour extraire des connaissances structurées sur les cocktails et leurs ingrédients ;
- Transformer les résultats SPARQL bruts en un graphe de connaissances qui représente les relations cocktail–ingrédient ;
- Appliquer des méthodes de fouille de données (dans notre cas le Clustering) afin de révéler des proximités non triviales ;
- Proposer une application interactive permettant une exploration relationnelle de cet univers, orientée vers des cas d'usage concrets tels que la gestion de bar, la découverte et l'optimisation
- Offrir une base pour des fonctionnalités avancées, notamment l'assistance à la recommandation ou à la décision.

2. Architecture du Projet

Il est à noter que dans le cadre de ce rapport, nous ne détaillerons pas avec le même niveau de détail toutes les couches de notre application. La figure ci-dessous détaille la vue d'ensemble de l'application.

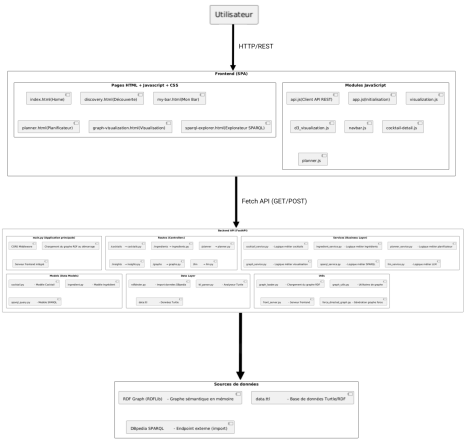

Figure 1. Schéma de l'architecture

Nous avons opté pour une architecture en 3 couches (front-end, back-end et data). L'objectif était de produire une architecture assez robuste et évolutive, tout en restant dans nos domaines de compétences respectifs afin de se concentrer sur le WS.

2.1. Couche de données

2.1.1. Importation des données

La première étape consiste à importer les données depuis DBpedia à partir d'une requête `CONSTRUCT` pour les stocker dans un fichier local : `data.ttl`.

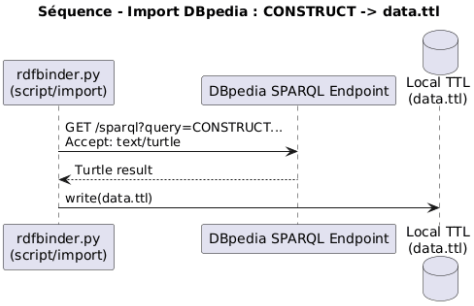

Figure 2. Diagramme de séquence : Importation des données

2.1.2. Données récupérées

Les données sont récupérées depuis DBpedia à l'aide d'une requête SPARQL de type `CONSTRUCT`. Nous parcourons l'information à partir de la ressource `dbr:List_of_IBA_official_cocktails`, ce qui nous permet d'identifier les cocktails officiellement reconnus par l'International Bartenders Association.

Pour chaque cocktail identifié, nous récupérons un ensemble de propriétés descriptives et contextuelles, telles que le nom (`rdfs:label`, `dbp:name`), une description textuelle (`dbo:description`), une illustration (`foaf:depiction`), ainsi que des informations issues de l'infobox Wikipédia (`dbp:ingredients`, `dbp:prep`, `dbp:garnish`, `dbp:served`). Ces données sont complétées par des catégories encyclopédiques (`dct:subject`) et par les liens sortants vers d'autres pages Wikipédia (`dbo:wikiPageWikiLink`), fournissant un contexte relationnel supplémentaire.

Ce sous-graphe reflète fidèlement la structure des informations telles qu'elles sont publiées dans DBpedia. Cependant les ingrédients et les instructions de préparation sont représentés sous forme de littéraux non structurés, ce qui empêche toute exploitation directe. De plus, les liens encyclopédiques et les catégories introduisent un certain bruit, en reliant les cocktails à des entités dont la pertinence analytique est parfois douteuse.

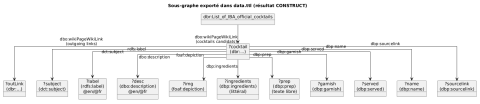

Figure 3. Sous-graphe reconstruit par requête CONSTRUCT

> **Note sur l'IBA**
>
> Nous avons volontairement limité l'étude aux cocktails officiels de l'International Bartenders Association (55 cocktails). Ce choix est avant tout pragmatique : sans ce cadre, DBpedia considère comme cocktail à peu près tout ce qui peut se boire, des *mini shields Fortnite* à la *lean* associée à la drogue. *AAA* certes, mais ici nous avons préféré recentrer l'analyse sur la mixologie au sens strict..

2.2. Couche Backend et logique métier

Nous allons illustrer comment à partir de notre fichier turtle, nous traitons les données RDF dans le backend. Pour ce faire la figure suivante illustrera la pipeline de traitement des données RDF.

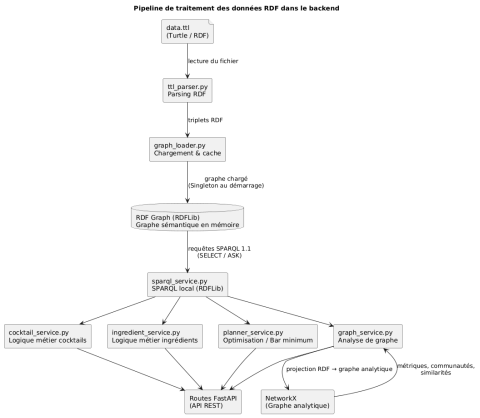

Figure 4. Pipeline des traitements RDF

> **Note sur le parsing**
>
> Les données des recettes n'étant pas toutes au même format, nous avons supprimer les informations de quantités pour ne garder que les ingrédients. Nous perdons certes en précision, mais cela permet d'avoir des données standardisées, et au même format pour les traitements. Par exemple : "30 ml gin" ou "1 splash of gin" devient alors "gin" et de même pour les autres ingredients.

2.3. Tests et Validation

Afin de garantir la fiabilité du backend et faciliter les évolutions du code, une suite de tests complète a été mise en place à l'aide du framework `pytest`.

La stratégie de test couvre plusieurs niveaux :

- **Tests Unitaires** : Vérification de la logique métier isolée, notamment pour les services de planification et d'optimisation des ingrédients.

- **Tests d'API** : Utilisation de `TestClient` de FastAPI pour simuler des requêtes HTTP et valider les réponses des endpoints.
- **Mocking** : Utilisation de `unittest.mock` pour isoler les composants et simuler les dépendances externes (comme les appels aux LLM), permettant des tests rapides et déterministes.
- **Tests d'Intégration** : Validation de la chaîne complète de traitement sur des scénarios nominaux.

Cette approche rigoureuse a été particulièrement bénéfique lors des phases de refactoring, permettant de détecter instantanément les régressions potentielles.

Nous avons atteint un test coverage total d'environ 69% sur le backend.

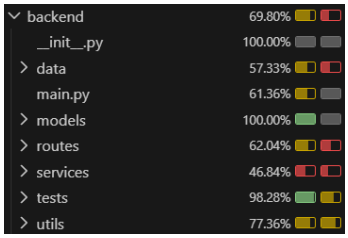

Figure 5. Test coverage

3. Difficultés rencontrées

3.1. Les données..

Bien que DBpedia fournisse un volume important d'informations sur les cocktails officiels de l'IBA, les données présentent plusieurs limites structurelles. Le modèle est avant tout encyclopédique et non orienté recette , ce qui se traduit par une impertinence initiale des champs (`dbp:ingredients`) et des instructions de préparation (`dbp:prep`) comme énoncé précédemment dans la note sur le parsing .

3.2. Organisation du projet

L'architecture logicielle et le model de données ont été crée avant d'être sur de quelle données allait être extraite. Ceci n'a pas posé de problèmes au début, mais par la suite, nous avions des modules qui utilisaient des données parsées et d'autres non, ce qui engendrait plein d'incohérences. Cela nous a pris 2 bonnes heures pour tout restandardisés et repartir sur de bonnes bases.

3.3. Trouver un sujet

Le cadre du projet étant volontairement libre, nous avons décidé de partir sur un sujet de la catégorie de type "*Autre*". Nous avons donc du vérifier au préalable que nous avions le volume de donnés nécessaire pour répondre au fonctionnalités que nous voulions développer. Cette étape de vérification supplémentaire et ce désir de "ne pas vouloir faire comme les autres" nous a fait perdre un peu de temps au début, mais nous sommes très satisfait d'avoir pu traiter ce sujet.

3.4. DBpedia access point

Lors de notre requete construct, nous n'obtenions pas les mêmes résultats. Nous prenions des timeouts de la part de DBpedia, ce qui stoppait notre collecte de données. Nous avons pris un certain temps avant de parvenir à une reproductibilité systématique. Il ne suffisait malheureusement pas de changer le paramètre timeout de notre requête...

3.5. Changement de modèle de langage en cours de projet

La solution utilisée au départ montrait plusieurs limites, notamment une génération de requêtes SPARQL peu fiable et même syntaxiquement incorrectes Pour garantir un comportement plus robuste et prédictible, nous avons choisi de migrer vers les modèles OpenAI. Ce changement a impliqué un coût en temps d'intégration, mais a permis d'obtenir des résultats plus exploitables dont nous sommes globalement satisfaits.

4. Réflexions sur le Web Sémantique

Sur le papier, le Web sémantique fait rêver. Des données bien structurées, des ontologies partagées et des machines capables de comprendre le sens des informations pour raisonner entre elles. Le cours nous vend presque une vision élégante : un Web où les graphes se comprennent mutuellement et où la sémantique émerge presque naturellement. Avec MarmiTonic, on a découvert que ce n'était pas si simple..

DBpedia coche pourtant toutes les cases théoriques : RDF, SPARQL, URI ontologie... Mais dès que l'on tente de l'utiliser sur un domaine concret comme la mixologie, les problèmes commencent. Dans notre cas les cocktails n'ont pas des recettes structurées, les ingrédients sont des chaînes de caractères et les relations sont souvent de simples liens encyclopédiques. On peut donc affirmer que les données sont très compréhensibles pour un humain mais un peu moins pour une machine qui aimerait avoir autre chose qu'un long champ texte séparé par des virgule.

Nous pensons que le point du projet est justement là. Sans intervention humaine, il y a très peu de sémantique exploitable, ou du moins dans notre domaine. Le sens n'est pas tombé du graph comme par magie, on a du filtrer agressivement, faire des hypothèses et reconstruire un sous-graphe cohérent. Le coeur du projet était ce travail de "traduction" entre des données ouvertes et des données exploitables (bien que l'écriture de requettes SPARQL aussi élégantes soient-elles en faisait aussi partie).

MarmiTonic montre également que les machines ne raisonnent pas vraiment entre elles dans ce contexte. Elles exécutent très bien des requêtes, mais uniquement sur des structures que des humains ont préalablement préparées avec soin. Les analyses intéressantes (similarité, centralité, optimisation de bar) ne viennent pas directement du Web sémantique, mais de couches analytiques ajoutées par-dessus. Le RDF sert ici de fondation solide, mais pas encore de cerveau autonome.

Cela ne signifie pas que le Web sémantique est inutile bien au contraire! Il permet d'explorer rapidement un domaine, de récupérer des données riches et d'éviter de tout construire à la main. Mais le fantasme d'un Web entièrement compréhensible et exploitable automatiquement reste, dans notre cas, assez éloigné de la réalité. MarmiTonic nous montre finalement une vérité toute simple : le Web sémantique fonctionne très bien… à condition qu'un humain soit là pour lui expliquer ce qu'il est censé vouloir dire, et aussi pour lui rappeler que tout ce qui peut se boire n'est pas forcément un cocktail!

5. Interpretations extraites et visualisations

5.1. Tri des cocktails par vibes

En tant qu'utilisateur, nous pouvons avoir accès à notre historique de cocktails (illustré en 1). De plus, nous fournissons à l'utilisateur des "vibes" de cocktails. Il trouvera des cocktails associé à la vibe dans le caroussel (illustré en 2).
Pour proposé cette fonctionnalite nous avons utilisé :

- `sentence-transformer` qui nous sert à transformer des phrases en vecteurs
- `FAISS`, une libraire qui retrouve les vecteurs les plus proches.
- On se retrouve donc avec un `RAG` qui nous permet de décrire les différents clusters.
- Ces clusters sont ensuite les inputs d'un LLM (MistralAI) qui nous donne en output des noms de clusters.

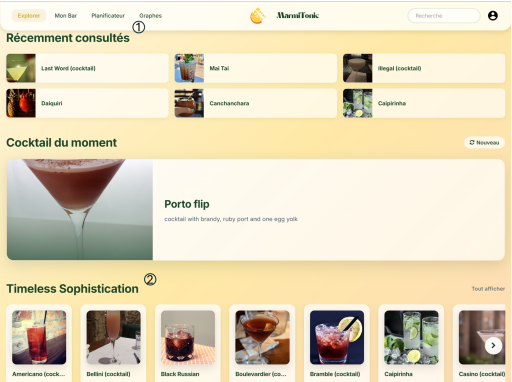

Figure 6. Snapshot de la page "Explorer"

5.2. Trouver des cocktails à partir d'ingrédients

Avec la fonctionnalité *"Mon Bar"* nous permettons à l'utilisateur de connaître les cocktails qu'il peut fabriquer à partir des ingrédients en sa possession, ainsi que ce qu'il est presque capable de réaliser. Cela lui évite d'aller chercher toutes les recettes et lui permet donc de gagner du temps, merci le WS !

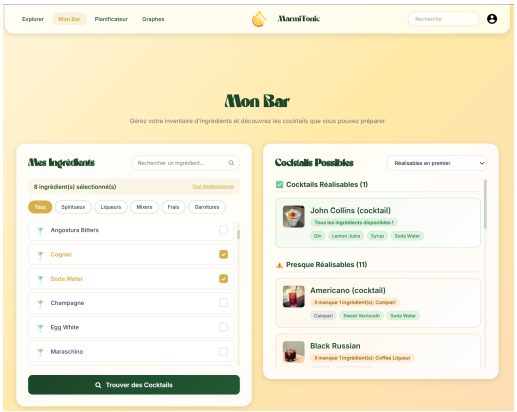

Figure 7. Snapshot de la page "Mon Bar"

5.3. Cocktails Similaires

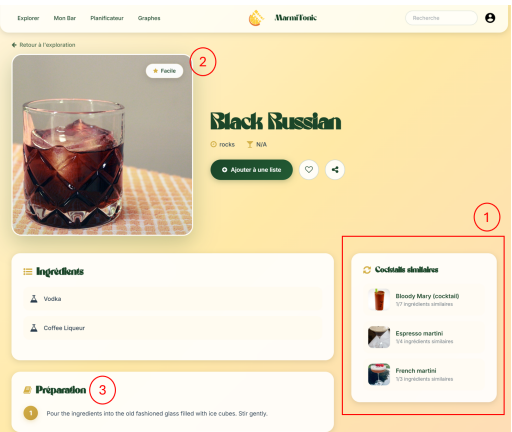

Figure 8. Snapshot de la page d'un Cocktail

Nous pouvons voir en 1 que l'utilisateur aura accès aux objets similaires à son cocktail par ingrédients.

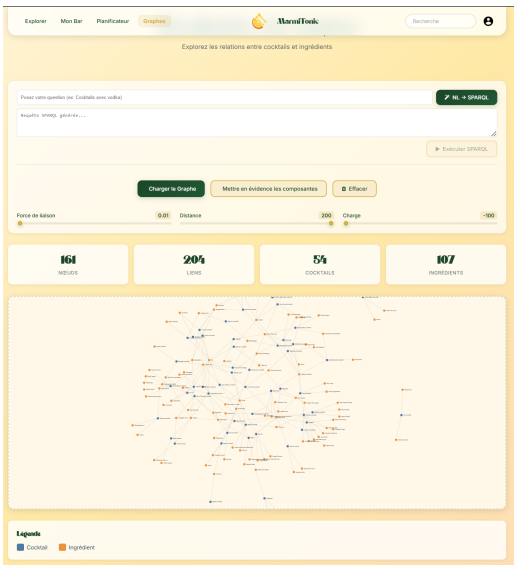

Figure 10. Snaphsot de la page Graph

Note et observations

Les cocktails d'une même vibe sont systèmatiquement recommendés. On en déduit que nos clusters sont donc cohérents.

En 2 il aura accès à la difficulté de la réalisation du cocktail (Facile, Moyen, Difficile) en fonction du nombre d'ingrédients dans le cocktail.

En 3 il aura accès à la recette de son cokctail.

Note sur l'attribution de la difficulté

Nous aurions également pu faire cette attribution via un LLM, cependant par manque de temps nous avons choisi une attribution plus simple.

5.4. Playlist et ingrédients

L'utilisateur peut créer des playlists de cocktails et obtenir directement tous les ingrédients nécessaires pour tous les faire.

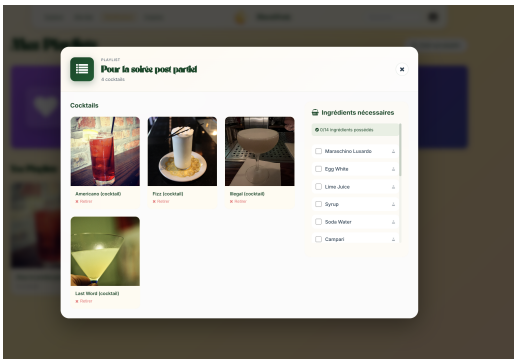

Figure 9. Snaphsot de la page Playlist

5.5. Sous-graph des cocktails et ingrédients

Depuis cette page, nous pouvons à partir du language naturel(NL), transformer en requête SPARQL. Ensuite nous obtenons un graph avec les cocktails et les ingrédients.

Note et observations

Depuis le graph général (tous les cocktails et tous les ingrédients), on peut déjà répérer des cocktails et des ingrédiets "hub".

5.6. Optimiseur de Bar

Comme on pouvait s'y attendre, les ingrédients hub aparaissent bel et bien dans notre bar Optimal. Ainsi pour un nombre d'ingrédients donné, nous pouvons sorir le maximum de cocktails.

Esprit critque

Comme nous avons perdu en précision en enlevant les quantités, il est donc nécessaire de faire des recherches supplémentaires affin de pouvoir obtenir directement une liste de course.

Figure 11. Snapshot de la page Optimiseur

6. Conclusion synthétique

Nous n'avons pas cherché à pousser le projet inutilement loin, mais il tient fonctionne parfaitement et se révèle même utile : on y trouve déjà de vrais usages, ce qui est une victoire. Certaines briques techniques relèvent objectivement de l'overkill (notamment le RAG), mais ce choix était assumé et motivé par une volonté d'apprentissage et d'exploration. Le sujet, volontairement décalé, nous a permis de travailler avec engagement et curiosité, sans perdre de vue les objectifs techniques.

Malgré une qualité de données initialement médiocre, nous avons réussi à en extraire de l'information exploitable et cohérente. La gestion et le traitement des données constituaient le cœur du projet, et c'est précisément sur cet aspect que nous estimons avoir fourni le travail le plus solide.

En résumé : des données douteuses, des choix techniques parfois excessifs, mais un système qui fonctionne… et qui, contre toute attente, dit même des choses intéressentes.

A. Annexes

A.1. Liste complète des requêtes SPARQL

Toutes les requêtes sont présentées dans un fichier SPARQL-QUERIES.md.

A.2. Organisation du Travail

Nous possédons notre propre ooutil de gestion de projet que nous utilisons depuis le 1er projet 4IF. Nous travaillons avec la méthode AGILE avec une sous variante KANBAN, qui nous permet de nosu rendre compte de l'avancée des tâches, et de où en sont les membres de l'hexanôme.

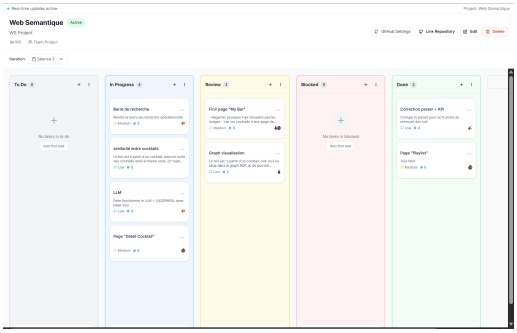

Figure 12. Kanban custom tool : Répartition des tâches