Identification systématique de modules dans une collection de requêtes SPARQL

Alix Regnier

UNIVERSITÉ DE RENNES
MASTER 1 BIO-INFORMATIQUE ET GÉNOMIQUE
2022–2023

Alix REGNIER

Identification systématique de modules dans une collection de requêtes SPARQL

Encadrant : **Olivier DAMERON**

IRISA, équipe Dyliss
263 Av. Général Leclerc, 35000 Rennes

Université de Rennes

UFR Sciences de la vie & de l'environnement

ENGAGEMENT DE NON PLAGIAT

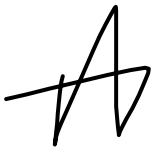
Je, soussigné (e) REGNIER Alix

Etudiant (e) en Master 1 de Bio-informatique et génomique

Déclare être pleinement informé (e) que le plagiat de documents ou d'une partie de documents publiés sous toute forme de support (y compris l'internet), constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour la rédaction de ce document.

Signature



INSERM U1242 OSS Equipe PROSAC

Centre Eugène Marquis Avenue de la Bataille Flandres Dunkerque
35042 Rennes

Annabelle MONNIER
annabelle.monnier@univ-rennes1.fr

TÉL. 33 (0)2 23 23 61 14

Remerciements

Je tiens particulièrement à remercier les membres et stagiaires des équipes du projet Symbiose [1] qui m'ont permis d'effectuer mon stage dans un environnement de travail très agréable tant bien humainement que scientifiquement.

1. Symbiose est l'union des membres des équipes Dyliss, GenScale et de la plateforme GenOuest

Sommaire

Abréviations

ANTLR	ANother Tool for Language Recognition
CAH	Classification Ascendante Hiérarchique
CALIPHO	Computer Analysis and Laboratory Investigation of Proteins of Human Origin
FAIR	Findable Accessible Interoperable Reusable
GO	Gene Ontology
HUPO	HUman Proteome Organization
MCIS	Maximum Common Induced Subgraph *(usually called MCS)*
MCS	Maximum Common Subgraph
QC	Quality Control
RDF	Resource Description Framework
SIB	Swiss Institute of Bioinformatics
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

1 Introduction

Le dernier recensement mentionnait plus de 1600 bases de données de référence en sciences de la vie [1]. Mais leurs implémentions posent de réels soucis techniques lorsqu'il advient de devoir les combiner [2]. De plus, une étude prospective montre que les sciences de la vie constituent le domaine des Big Data le plus exigeant en termes d'acquisition, de stockage, de distribution et d'analyse de données et que cette tendance va vraisemblablement s'amplifier [3]. C'est pourquoi il est nécessaire de trouver une manière plus adéquate afin de répondre à ces problèmes.

Le Web Sémantique fournit une solution technique qui permet de résoudre le problèmes d'intégration et d'interrogation, notamment avec les graphes RDF. C'est d'ailleurs ce qui explique le fait que de plus en plus de bases de références adoptent RDF pour décrire les données et le langage SPARQL comme interface pour les interroger [4, 5, 6]. En revanche, l'expertise technique nécessaire à son utilisation est un frein à son adoption [7], notamment par les personnes qui ne connaissent pas SPARQL et dont ce n'est pas leur métier. Les bases de données fournissent souvent des diagrammes simplifiés et des exemples de requêtes mais cela est loin d'être suffisant pour permettre son utilisation sans connaissances préalables. Aucun de ces éléments n'apporte de solution satisfaisante à l'obstacle qui existe entre ce que veulent faire les utilisateurs (le « quoi ») et le code SPARQL correspondant (le « comment »). C'est pourquoi il est nécessaire de créer de nouvelles méthodes afin d'automatiser et de faciliter la création de requête SPARQL. En outre, neXtProt est une base de données d'annotation de protéines humaines interrogeable avec SPARQL. D'une part ils fournissent un diagramme et une documentation complète de leur schéma de données, d'autre part, ils proposent une collection de 777 exemples de requêtes annotées qui peuvent être utilisées.

2 Objectif

Nous cherchons à établir un ensemble de fragments de requête réutilisables comme « protéine dans un état chimique » ou « protéine localisée dans un compartiment cellulaire » qui d'un côté, pourraient être assemblés pour créer facilement des requêtes qui répondent à des questions biologiques comme « quelles sont les protéines phosphorylées localisées dans le cytoplasme ? » et d'un autre côté pourraient être réutilisés entre les requêtes. L'intérêt serait ainsi de pouvoir simplifier l'écriture de requêtes en les composant à partir des fragments. Plutôt que de créer ces fragments *a priori* et manuellement, nous supposons qu'il est possible de les identifier à partir d'une collection de requêtes existantes. Par la suite, nous appellerons « modules » les fragments

de requêtes ayant un sens biologique et pouvant être assemblés pour créer de nouvelles requêtes SPARQL. **Le but de ce stage est d'établir une méthode permettant d'identifier des modules dans une collection de requêtes SPARQL.** Les 777 requêtes annotées de la base neXtProt [1] nous serviront de jeu de validation et d'évaluation. Nous faisons l'hypothèse que parmi les requêtes, il y a une similarité entre les modules qu'elles partagent et leurs annotations.

3 Contexte

3.1 Web Sémantique

Le Web Sémantique [8] [2] fournit un cadre technique commun qui permet de partager et de réutiliser des données au-delà des limites des applications, des entreprises et des communautés. Il s'agit d'un effort de collaboration mené par le W3C avec la participation d'un grand nombre de chercheurs et de partenaires industriels. Il est basé sur le cadre de description des ressources (RDF) [3].

RDF, URI & Triplet En RDF, toute entité est appelée ressource et est identifiée de manière unique par un URI (Uniform Resource Identifier). Les données sont décrites sous forme de triplets qui permettent de déclarer une ressource comme ayant une relation avec d'autres ressources ou ayant des propriétés intrinsèques, comme illustré sur la figure 1. Enfin, l'ensemble de ces triplets forme ce que l'on appelle un graphe RDF, illustré sur la figure 2.

1. `https://github.com/calipho-sib/nextprot-queries`
2. `https://www.w3.org/standards/semanticweb/`
3. RDF *Resource Description Framework* : `https://www.w3.org/2001/sw/`

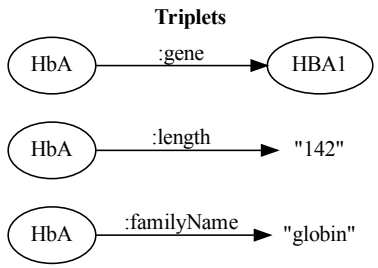


Figure 1. Triplets RDF décrivant la protéine HbA. Les ressources sont représentées par des nœuds ovales ; leur identifiant sous forme d'URI n'est pas figuré. Les relations sont également identifiées par des URI, dont seule la partie locale est représentée. Les nombres et les chaînes de caractères sont délimités par des guillemets.

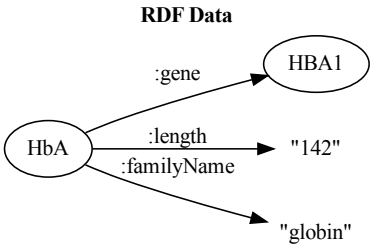


Figure 2. Graphe RDF constitué des triplets de la figure 1. Lorsque plusieurs triplets font référence à la même ressource (comme par exemple HbA), leurs nœuds respectifs sont fusionnés. Le graphe RDF obtenu a une structure de graphe dirigé étiqueté.

SPARQL SPARQL est le langage de requête développé par le W3C qui permet d'interagir avec les données RDF [9]. Les variables commencent par un point d'interrogation et permettent d'extraire la valeur du sommet dans les résultats. Les requêtes SPARQL sont des motifs de graphes (c.f figures 7 et 8), constitués de triplets mais où certaines entités ou certaines relations sont des variables (leur nom commence par un point d'interrogation). Rechercher les résultats d'une requête sur un graphe de données RDF consiste à déterminer tous les sous-graphes du graphe de données qui correspondent aux contraintes de la requête. Les valeurs que prennent les variables dans chaque sous-graphe constituent alors une solution.

3.2 Le protéome fonctionnel et la base de données neXtProt

Il y a encore beaucoup de protéines humaines dont la fonction n'est pas encore connue ou annotée [10] [4]. Le projet HUPO (*HUman Proteome Organization*) a pour but de trouver toutes les protéines manquantes, protéines n'ont toujours pas été découvertes ou dont la fonction reste encore inconnue à ce jour. En collaboration avec HUPO, neXtProt amorce la mise en place de la prédiction fonctionnelle des protéines [11] [5].

neXtProt neXtProt est une base de données développée au sein du SIB (*Swiss Institute of Bioinformatics*) par le groupe CALIPHO (*Computer Analysis and Laboratory Investigation of Pro-*

4. `https://www.nextprot.org/about/human-proteome`
5. `https://www.hupo.org/about-hupo`

teins of Human Origin) [12]. Elle contient uniquement des données publiques sur les protéines humaines, sujet auquel le projet se consacre. La base contenait à l'origine des protéines provenant des bases de données UniProt et Swiss-Prot. Le but de neXtProt est de pouvoir proposer des protéines humaines minutieusement annotées et de haute qualité. Le schéma de données de neXtProt illustré à la figure 3 permet de représenter le rôle de la protéine, le gène dont elle provient, la manière dont elle a été annotée, les références ontologiques, les bases de données connexes, de quelle manière elle a été mise en évidence, le pathway, les interactions protéine-protéine, etc [6].

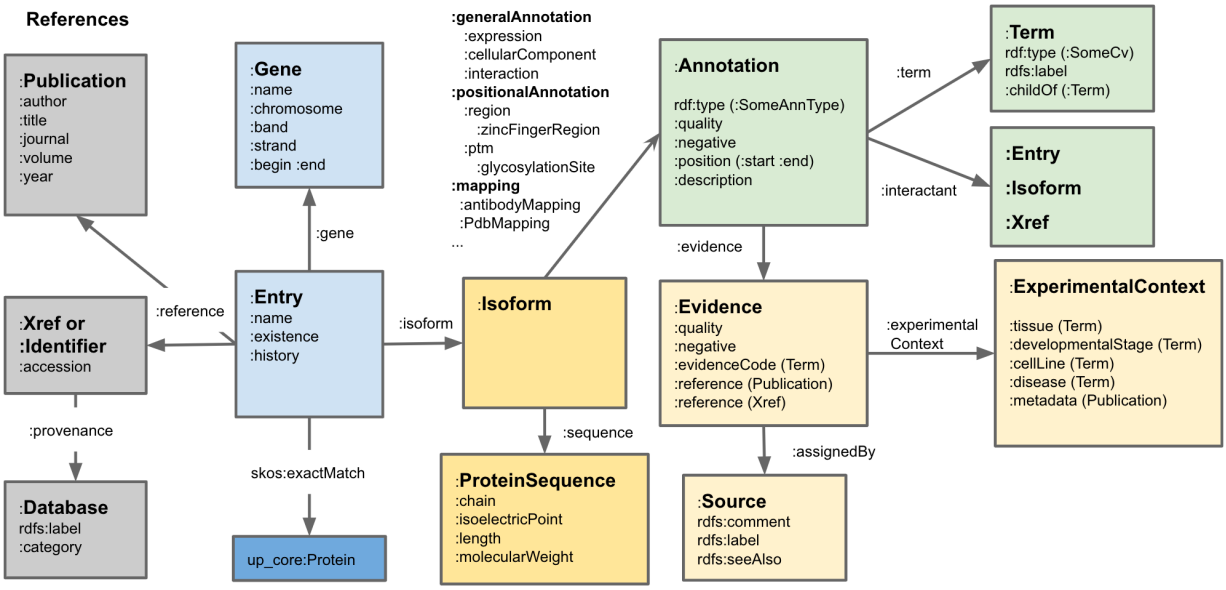


Figure 3. Schéma simplifié de la base de données neXtProt (source : `https://www.nextprot.org/rdf/`)

4 Matériels et méthodes

4.1 Développement logiciel et gestion des données

De par la nécessité de faire du développement, de devoir gérer du code et dans un engagement FAIR, le jeu de données, les scripts et les programmes qui ont permis de trouver les résultats sont mis à disposition sur des dépôts Git [7, 8].

6. `https://www.nextprot.org/about/nextprot`
7. `https://github.com/AlixRegnier/nextprot-sparql`
8. `https://github.com/AlixRegnier/sparql-visualizer`

4.2 Analyse du jeu de données

Afin d'estimer si on pouvait s'attendre à trouver des modules pertinents dans les requêtes de neXtProt et d'évaluer la pertinence des modules que je vais identifier, nous envisageons d'annoter automatiquement les modules grâce aux mots-clefs des requêtes où ils sont présents. Alors, j'ai réalisé une étude des requêtes et de leurs mots-clefs. La manière dont sont annotées les requêtes est visible sur la figure 7 (ligne 6). La distribution du nombre de requêtes par mot-clef indique si un mot-clef est utilisé dans plusieurs requêtes, ce qui suggère l'existence d'un module dans ces requêtes. La co-occurrence des mots-clefs pourrait servir à inférer des modules qui sont présents lorsque plusieurs et même mots-clefs sont présents, ce qui pourrait nous servir pour la construction de futures requêtes. Ensuite, j'ai effectué une analyse quantitative des mots-clefs afin de savoir si certains mots-clefs sont plus spécifiques voire plus rares que d'autres.

4.3 SPARQL Visualizer

J'avais besoin de traiter automatiquement des requêtes SPARQL de manière à reconnaître les entités, les relations et les différents éléments de SPARQL (e.g. les blocs, les filtres, etc.). C'est pourquoi j'ai développé un programme qui permet d'effectuer cette tâche appelé *SPARQL Visualizer*.

4.3.1 Analyse syntaxique de requêtes SPARQL avec ANTLR4

SPARQL Visualizer utilise un analyseur syntaxique aussi appelé « parser », généré par l'outil ANTLR4 [13] [9]. Son parser nous permet notamment de parcourir un arbre syntaxique abstrait et j'ai étendu le parser afin qu'il puisse répondre à nos besoins. J'ai généré le parser en utilisant une grammaire de requête SPARQL respectant celle établie par le W3C [10] que j'ai modifiée afin d'accepter les mots-clefs SPARQL écrits en minuscules. La grammaire est disponible au format G4 (format de grammaire ANTLR4) sur le Git du projet [11].

4.3.2 Conversion de requêtes SPARQL en graphe dirigés

SPARQL Visualizer permet de convertir nos requêtes en graphe et cela me permet de travailler sur les données en utilisant la théorie des graphes. Cela nous permet également de pouvoir visualiser nos requêtes et cela nous est essentiel dans le but de comprendre la structure de nos requêtes.

9. `https://www.antlr.org/about.html`
10. `https://www.w3.org/TR/sparql11-query/#sparqlGrammar`
11. `https://github.com/AlixRegnier/sparql-visualizer/blob/main/Sparql.g4`

Afin de pouvoir travailler sur nos graphes, je me suis servi de la librairie Python **NetworkX** [12] qui permet la création simple de graphe tout en proposant beaucoup d'algorithmes utiles tel que VF2 qui nous permet de détecter efficacement si des graphes sont isomorphiques [14, 15]. Pour ce qui est de l'affichage, nous convertissons nos graphes en notation **dot** [13] afin qu'ils puissent être rendus en image par **GraphViz** [14].

4.4 Extraction des modules

4.4.1 Sous-graphe commun maximum

Utiliser la structure des graphes et notamment leurs similarités pourrait être une piste pour extraire les modules. Ce qui voudrait dire que l'on pourrait réduire l'identification de modules à la détection d'isomorphismes entre deux sous-graphes de deux graphes de requêtes. En faisant l'assertion que les modules sont des sous-graphes induits et faiblement connectés, je veux comparer les graphes des requêtes entre-eux afin de trouver les plus grands sous-graphes en commun. Il s'agit d'un problème classique et NP-difficile. Cependant, dans la littérature, les algorithmes existants ne proposaient pas de réponses satisfaisantes quant à ce que nous voulions car soit ils se basaient sur des heuristiques, soit ils n'étaient pas adaptés pour des graphes orientés ou soitils ne permettaient pas d'assurer que le résultat soit un sous-graphe faiblement connecté [16, 17]. J'ai alors écrit un algorithme qui permet de répondre à nos besoins et qui calcule tous les sous-graphes communs maximaux induits (*Maximum Common Induced Subgraph*). De part la nature de nos graphes, on suppose que chaque MCIS est au moins faiblement connecté.

4.5 Analyse des modules

4.5.1 Composition de modules

Nous appelons « sous-modules », tout module qui est inclus dans au moins un autre module. Afin de mesurer l'inclusion des modules au sein d'autres modules, nous comparons tout nos modules les uns avec les autres et on regarde si un sous-graphe d'un module $m1$ est un isomorphisme d'un module $m2$ (avec $m1 \neq m2$). Car, nous faisons l'assertion que s'il existe un isomorphisme entre un sous-graphe de $m1$ et le graphe de $m2$, alors le graphe de $m2$ est un sous-graphe du graphe

12. `https://networkx.org/`
13. `https://graphviz.org/doc/info/lang.html`
14. `https://graphviz.org/about/`

de $m1$ donc le module $m2$ est une sous-module de $m1$. Pour tout module, nous pouvons alors savoir quels sont les modules qui le *composent*, donc nous pouvons dessiner un graphe de composition.

Un module qui ne comporte aucun sous-module est une « feuille » (car sur un graphe de composition, il n'aurait aucun enfant). Nous nous attendons à ce que les feuilles soient essentiellement des modules de petite taille ou des modules assez spécifiques.

4.5.2 Association de modules

Une autre problématique est comment *associer* des modules, c'est-à-dire, sur quels sommets les graphes de deux modules se superposent en mettant en correspondance des sommets d'un module avec des sommets d'un autre module. Nous pouvons calculer comment les modules se superposent dans les requêtes en utilisant leur mapping distinct aux même requêtes. En effet, si deux modules sont présents dans une même requête et qu'ils ont des sommets du graphe de la requête en commun, alors on peut inférer comment les modules peuvent s'associer.

4.5.3 Annotation des modules par des mots-clefs

La manière la plus simple d'annoter un module est d'utiliser l'intersection des mots-clefs des requêtes dans lesquelles on peut trouver le module. En effet, si un ensemble de mots-clefs est commun à chaque requêtes où est présent le module, on peut alors faire la conjecture que ce module semble être associé à cet ensemble de mots-clefs.

5 Résultats

5.1 Développement logiciel et gestion des données

Le code qui a été écrit pour effectuer l'analyse, pour visualiser les graphes et extraire les modules est disponibles sur plusieurs dépôt Git [15, 16]. Les dépôts constituent un total d'environ 1700 lignes de codes.

15. `https://github.com/AlixRegnier/nextprot-sparql`
16. `https://github.com/AlixRegnier/sparql-visualizer`

5.2 Analyse du jeu de données

5.2.1 Relations entre requêtes et mots-clefs

Les données présentées ci-dessous ont été analysées sur un jeu de données contenant 777 requêtes SPARQL annotées de la banque de données neXtProt. Les requêtes sont également annotées par des mots-clefs, nous en comptons 74 différents.

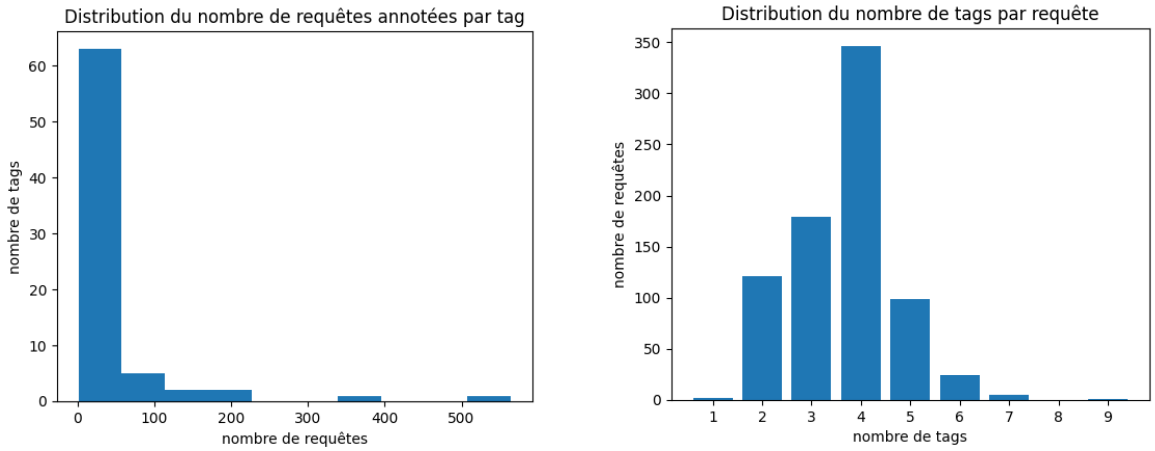


Figure 4. Barplots et histogrammes des distributions entre requêtes et mots-clefs. On voit que la plupart des mots-clefs annotent entre 1 et 50 requêtes, et que la plupart des requêtes sont annotées par 2 à 5 mots-clefs

L'histogramme de gauche de la figure 4 montre que les mots-clefs sont assez spécifiques. La grande majorité des mots-clefs annotent en général une cinquantaine de requêtes sur les 777 analysées. Par ailleurs, nous remarquons que les quelques mots-clefs annotant au moins 200 requêtes sont en général des mots-clefs informatifs tels que *Tutorial* ou *QC*. D'après le barplot de droite de la figure 4, nous pouvons voir que la majorité des requêtes sont annotées par 2 à 5 mots-clefs.

5.2.2 Co-occurrence des mots-clefs dans une requête

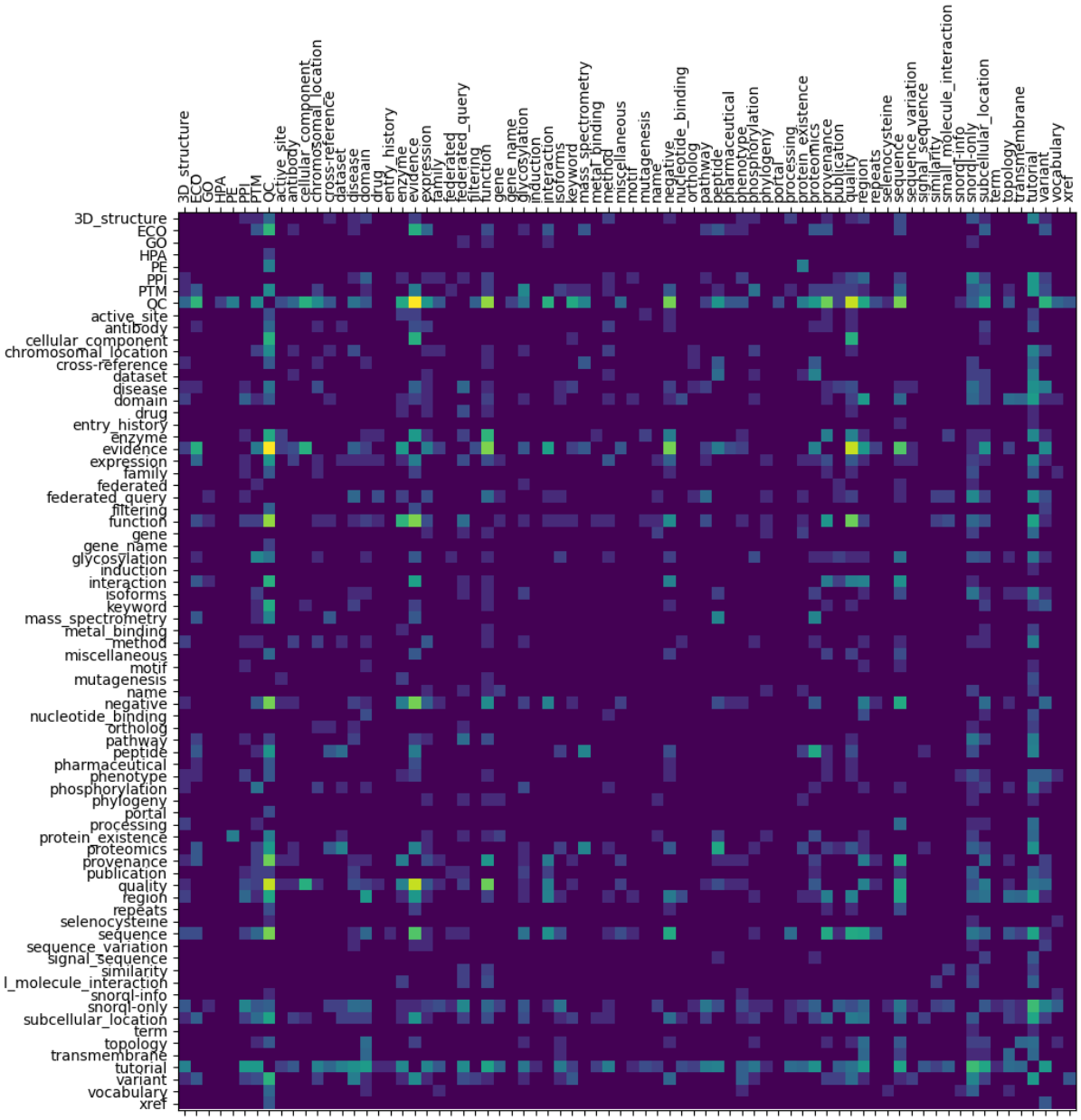


Figure 5. Heatmap de la co-occurrence des mots-clefs au sein des requêtes

La heatmap (figure 5) contient le nombre de fois où deux mots-clefs annotent la même requête. Elle permet d'avoir une idée rapide des mots-clefs qui peuvent être liés, c'est-à-dire très souvent en commun. Mais également si 2 mots-clefs ont des apparitions similaires avec d'autres mots-clefs. Afin de réduire l'impact des grandes valeurs sur les gradients de la heatmap, les valeurs de

la matrice ont subi la fonction $log(m_{i,j}+1)$, j'ajoute 1 pour éviter une erreur de domaine lorsque $m_{i,j}=0$. La diagonale a été mise à 0 pour que les valeurs n'interfèrent pas sur la coloration. Les résultats mettent en évidence qu'il y aura certainement des façons privilégiées de combiner des modules.

5.2.3 Mots-clefs contre requêtes

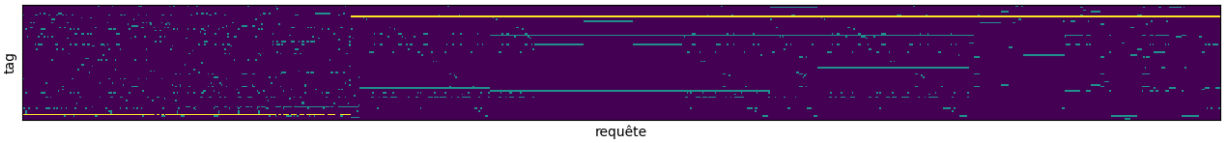


Figure 6. Heatmap de la présence des mots-clefs au sein des requêtes

Sur la heatmap (figure 6), chaque point représente le fait qu'une requête (colonne) est annotée par un mot-clef (ligne). Cette heatmap est composée de 0 (en violet) et de 1 (en cyan). Dans le but d'illustrer le fait que presque toutes les requêtes sont exclusivement annotées par les mots-clefs *QC* ou *Tutorial*, j'ai affiché ces lignes en jaune.

5.3 Analyse des requêtes SPARQL neXtProt

```
1  #id:NXQ_00002
2  #title:Proteins that are located in both the nucleus and in the cytoplasm
3  #comment:Select protein entries with an isoform.
4  #comment:The isoform must be localized in two subcellular components,
5  #comment:cytoplasm and nucleus and any child term thereof, without a negative evidence.
6  #tags:subcellular location,tutorial
7
8  select distinct ?entry where {
9    values ?cytoloc {cv:GO_0005737 cv:SL-0086} # GO and SL values for cytoplasm
10   values ?nucloc {cv:GO_0005634 cv:SL-0191} # GO and SL values for nucleus
11   ?entry :isoform ?iso.
12   ?iso :cellularComponent ?loc1, ?loc2 .
13   ?loc1 :term /:childOf ?cytoloc .
14   ?loc2 :term /:childOf ?nucloc .
15   filter not exists {?loc1  :negativeEvidence ?negev} # No negative localization evidence
16   filter not exists {?loc2  :negativeEvidence ?negev} # No negative localization evidence
17 }
```

Figure 7. Exemple d'une requête neXtProt permettant d'interroger la base de données sur les protéines qui sont localisées à la fois dans le noyau et dans le cytoplasme. Dans les commentaires, on trouve une description textuelle (champ *title*, ligne 2 et un ensemble de mots-clefs (champ *tag* ligne 6))

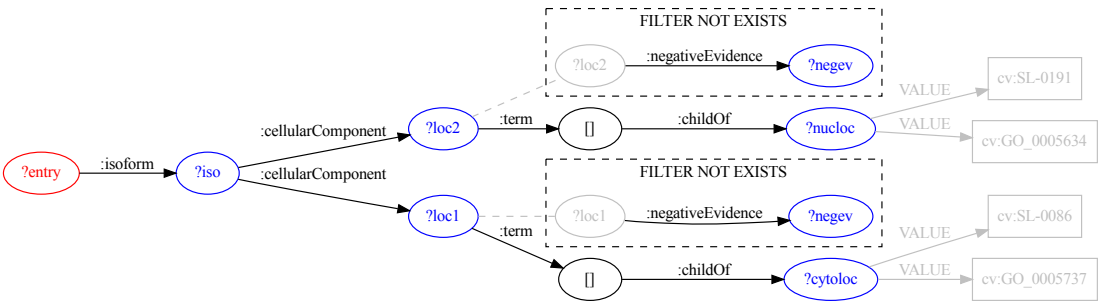


Figure 8. Graphe de la requête de la figure 7. On souhaite qu'au moins un isoforme (variable `?iso`) de la protéine (variable `?entry`) permette de suivre la relation `cellularComponent` sur les variables `?loc1` et `?loc2`, sur lesquelles on applique des contraintes supplémentaires pour qu'elles correspondent respectivement au cytoplasme et au noyau.

Le graphe de la figure 8 est une représentation fidèle à la requête de la figure 7, nous avons fait

le choix de dupliquer les nœuds qui sont inclus dans plusieurs niveaux du graphe (par exemple le nœud « ***?loc1*** » est à la fois dans un cluster *FILTER NOT EXISTS* et à la fois dans le graphe principal. Nous avons également fait le choix d'afficher les blank nodes (nœuds intermédiaires qui n'ont pas besoin d'un identifiant) avec le label « **[]** ».

5.4 Extraction de modules

5.4.1 Sous-graphe commun maximum induit

L'algorithme renvoie de très bons résultats et je filtre les résultats de manière à ne garder que les modules qui ont au moins 3 nœuds (qui contiennent donc au moins 2 arcs, minimum requis pour que le module ait un sens) et si et seulement s'il n'est pas un sous-module des résultats desquels il provient. On peut résumer l'algorithme à ce que pour chaque nœud n_1 du graphe G_1, on se positionne successivement sur chaque nœud n_2 du graphe G_2, de manière à parcourir simultanément les deux graphes (en utilisant des arcs avec le même label) afin de déterminer le sous-graphe commun maximum. Il se peut qu'il y ait à partir des nœuds n_1 ou n_2, plusieurs arcs ayant le même label qui conduisent à plusieurs possibilités, et ce sur des labels différents. Il est donc nécessaire d'élargir la recherche de manière à couvrir toutes les alternatives induites par l'arrangement des arcs de n_1 qui ont le même label que ceux de n_2. Enfin, si plusieurs arcs avec des labels différents ont mené à des alternatives différentes, il faut faire le produit cartésien des possibilités de chaque labels.

5.5 Analyse des modules

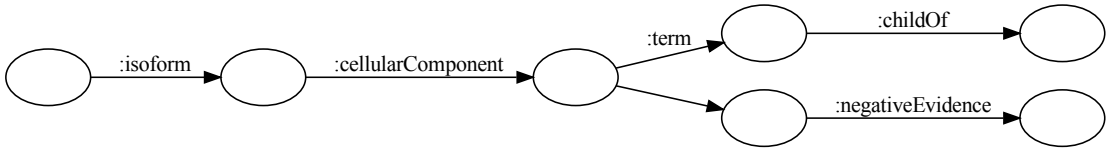


Figure 9. Exemple d'un module extrait du graphe de la Figure 8 (module161). Ce module correspond à la recherche du compartiment cellulaire dans lequel est localisée une protéine.

L'extraction a permis d'isoler 774 modules dans lesquels on peut trouver celui de la figure 9. Une majorité des modules semblent avoir un sens biologique cohérent tandis qu'une minorité

semble plus s'apparenter à du bruit, c'est-à-dire des modules qui seront sûrement éliminés ultérieurement.

5.5.1 Composition de modules

L'étude de la composition des modules a révélé qu'il y avait 179 feuilles parmi les 774 modules, c'est-à-dire qu'il y a 595 modules qui comportent au moins un sous-module. Il y a 8 requêtes parmi les 777 qui ne contiennent aucun module. Ceci s'explique par le fait qu'elles sont très spécifiques, soit qu'elles n'ont aucun élément en commun avec les autres requêtes. Quant à la présence de sous-modules, il y a un minimum de 0 sous-modules, une médiane de 2 sous-modules et un maximum de 43 sous-modules.

5.5.2 Association de modules

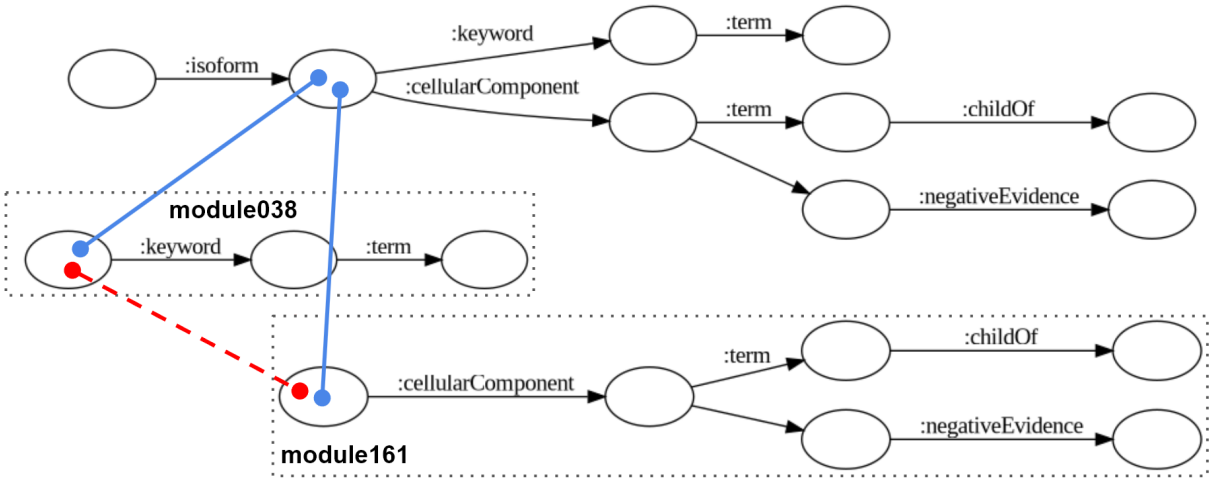


Figure 10. Exemple d'inférence d'association de deux modules à partir d'une requête. Les lignes bleues représentent le seul sommet du graphe de la requête sur lequel les deux modules se superposent. Le pointillé rouge représente l'association des sommets des deux modules déduite.

Associer les modules en utilisant l'intersection des sommets des requêtes sur lesquels ils se superposent a permis d'inférer 14602 paires de modules pouvant s'associer ; l'association de la figure 10 est un exemple tiré de nos résultats. Cela m'a également permis de voir qu'une paire de modules peut également s'associer de plusieurs manières.

6 Discussion

Optimisation de l'algorithme d'extraction des MCS Il existe dans la littérature de nombreux articles sur les sous-graphes maxima communs induits. Ils proposent plusieurs méthodes afin de réduire les coûts de recherches de MCS. Il y a notamment diverses méthodes utilisant les tables d'adjacence afin de réduire drastiquement les combinaisons de sommets à tester mais elles font appel à des heuristiques ce qui ne nous intéresse pas dans notre démarche. Néanmoins, il y a dans ces papiers des règles permettant de conclure directement qu'un ensemble de sommets ne peut être un MCS, ce qui constitue une optimisation du temps de calcul notoire [16, 17, 18].

Redondance au sein les modules extraits des requêtes L'analyse de la composition des 774 modules en sous-modules montre que 595 d'entre-eux (77 %) sont composés à partir de 179 feuilles (23 %), et cela sur plusieurs niveaux (on a des sous-modules eux-mêmes composés de sous-modules). Il est intéressant de constater que plutôt qu'un graphe de composition très plat, on obtient un graphe très profond. Dans le premier cas, on aurait plutôt une situation combinatoire où chaque requête est constituée de modules feuilles, alors que dans le cas de neXtProt il y a plusieurs niveaux imbriqués. Cela entraîne donc une certaine redondance dont il faut tenir compte lors de l'analyse de la fréquence des modules, puisque lorsqu'un module est présent dans une requête, tous ses sous-modules le sont aussi. Cela entraîne donc la question du bon niveau de composition, qui sera détaillé dans la section 7.

Couverture du schéma de données de neXtProt Au cours du stage, j'ai pu étudier la couverture du jeu de requêtes par les modules. Il aurait également été pertinent d'étudier dans quelle mesure le jeu de 777 requêtes couvre le schéma de données (dont une version simplifiée est présenté à la figure 3). Je n'ai pas eu le temps de mener cette étude faute de temps.

7 Perspectives

Assignation semi-automatique de mots-clefs aux modules L'assignation de mot-clefs aux modules est faites de manière semi-automatique, actuellement, j'annote automatiquement les modules en faisant l'intersection des mots-clefs des requêtes dans lequel on peut le trouver. Mais ceci a pour désavantage de rendre la méthode très dépendante aux annotations du jeu de données. Une amélioration serait de pouvoir corriger l'annotation de certaines requêtes à partir des modules, soit proposer une méthode d'autocorrection des requêtes.

Détermination d'un ensemble optimal de modules Les diverses associations et manières de s'associer des modules ne facilite pas la tâche de trouver un ensemble optimal de modules qui permettrait de recomposer toutes les requêtes du jeu de données initial (ou au moins en grande partie) mais également de composer de nouvelles requêtes. C'est pourquoi il serait intéressant d'utiliser un langage tel que ASP (*Answer Set Programming*) [19, 20] afin de pouvoir déterminer un sous-ensemble de modules de manière automatique. Cela requiert d'une part d'implémenter une méthode permettant de traduire des graphes dirigés en ASP et d'autre part de déterminer des règles logiques permettant de trouver ce sous-ensemble de modules.

Robustesse par rapport au jeu de requêtes initial Une autre question très importante est la robustesse. On peut se demander quel impact aurait le retrait de plusieurs requêtes du jeu de données sur les résultats ? D'ailleurs, une autre approche pour améliorer la robustesse serait d'exploiter des décompositions du schéma des données, ce qui permettrait à la fois de déduire des possibles modules manquants, de détecter les requêtes incohérentes et d'assurer que l'on couvre entièrement le graphe des données interrogeables (à l'échelle de la base de données).

8 Conclusion

Lors de ce stage, j'ai pu travailler avec un jeu de données non-conventionnel car les données étaient des requêtes annotées. J'ai pu réaliser une analyse quantitative sur ce jeu de données pour comprendre la nature des requêtes notamment par exemple si elles sont spécifiques puis j'ai créé un analyseur de requête qui permet de les transformer en graphe et de les afficher de manière claire à l'utilisateur. J'ai créé un algorithme qui essaye de répondre à un problème NP-difficile qui consiste à trouver dans deux graphes des sous-graphes en commun de taille maximale. Après avoir appliqué cet algorithme sur nos graphes en les comparant chacun deux à deux, j'ai pu isoler des modules candidats à partir desquels j'ai généré un graphe de composition qui permet de mettre en évidence la présence de modules au sein d'autres modules. Soit un aperçu d'un ensemble de modules qui peuvent s'assembler pour créer des requêtes.

Mes contributions sont ainsi de trois sortes : *(1) développement logiciel* (génération d'un parser de SPARQL avec ANTLR4, développement d'un visualiseur de requêtes, implémentation de l'extraction de modules), *(2) algorithmique* (adaptation des algirithmes classiques de MCIS pour identifier les modules) et *(3) analyse de données* (neXtProt + modules).

Bibliographie

[1] Daniel J Rigden et Xosé M Fernández. “The 2022 Nucleic Acids Research database issue and the online molecular biology database collection”. In : *Nucleic acids research* 50.D1 (2022), p. D1-D10. doi : https://doi.org/10.1093/nar/gkab848.

[2] Jacob Köhler. “Integration of life science databases”. In : *Drug Discovery Today : BIOSILICO* 2.2 (2004), p. 61-69. issn : 1741-8364. doi : https://doi.org/10.1016/S1741-8364(04)02392-3. url : https://www.sciencedirect.com/science/article/pii/S1741836404023923.

[3] Zachary D Stephens et al. “Big Data : Astronomical or Genomical?” In : *PLoS biology* 13.7 (2015), e1002195.

[4] Carole Goble et Robert Stevens. “State of the nation in data integration for bioinformatics”. In : *Journal of Biomedical Informatics* 41.5 (2008). Semantic Mashup of Biomedical Data, p. 687-693. issn : 1532-0464. doi : https://doi.org/10.1016/j.jbi.2008.01.008. url : https://www.sciencedirect.com/science/article/pii/S1532046408000178.

[5] Maulik R Kamdar et al. “Enabling Web-scale data integration in biomedicine through Linked Open Data”. In : *NPJ digital medicine* 2 (2019), p. 90. doi : https://doi.org/10.1038/s41746-019-0162-5.

[6] Maulik R Kamdar et Mark A Musen. “An empirical meta-analysis of the life sciences linked open data on the web”. In : *Scientific data* 8.1 (2021), p. 24. doi : https://doi.org/10.1038/s41597-021-00797-y.

[7] Jorge Pérez, Marcelo Arenas et Claudio Gutierrez. “Semantics and Complexity of SPARQL”. In : *ACM Trans. Database Syst.* 34.3 (sept. 2009), 16 :1-16 :45. issn : 0362-5915. doi : 10.1145/1567274.1567278. url : http://doi.acm.org/10.1145/1567274.1567278.

[8] Tim Berners-Lee, James Hendler et Ora Lassila. “The semantic web”. In : *Scientific american* 284.5 (2001), p. 34-43.

[9] Steve Harris, Andy Seaborne et Eric Prud’hommeaux. “SPARQL 1.1 query language”. In : *W3C Recommendation* (2013). url : https://www.w3.org/TR/sparql11-query/.

[10] Mehdi Sharifi Tabar et al. “Illuminating the dark protein-protein interactome”. In : *Cell reports methods* 2.8 (2022), p. 100275. doi : https://doi.org/10.1016/j.crmeth.2022.100275.

[11] Sam Hanash et Julio E Celis. “The Human Proteome Organization : a mission to advance proteome knowledge”. In : *Molecular & Cellular Proteomics* 1.6 (2002), p. 413-414.

[12] Lydie Lane et al. “neXtProt : a knowledge platform for human proteins”. In : *Nucleic acids research* 40.D1 (2012), p. D76-D83.

[13] Terence Parr. “The definitive ANTLR 4 reference”. In : *The Definitive ANTLR 4 Reference* (2013), p. 1-326.

[14] Luigi Pietro Cordella et al. “An improved algorithm for matching large graphs”. In : *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*. Citeseer. 2001, p. 149-159.

[15] Luigi P Cordella et al. “A (sub) graph isomorphism algorithm for matching large graphs”. In : *IEEE transactions on pattern analysis and machine intelligence* 26.10 (2004), p. 1367-1372.

[16] Giorgio Levi. “A note on the derivation of maximal common subgraphs of two directed or undirected graphs”. In : *Calcolo* 9.4 (1973), p. 341-352.

[17] Ciaran McCreesh, Patrick Prosser et James Trimble. “A Partitioning Algorithm for Maximum Common Subgraph Problems”. In : *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, p. 712-719. doi : 10.24963/ijcai.2017/99. url : https://doi.org/10.24963/ijcai.2017/99.

[18] James J McGregor. “Backtrack search algorithms and the maximal common subgraph problem”. In : *Software : Practice and Experience* 12.1 (1982), p. 23-34.

[19] Ilkka Niemelä. “Logic programs with stable model semantics as a constraint programming paradigm”. In : *Annals of mathematics and Artificial Intelligence* 25 (1999), p. 241-273.

[20] Vladimir Lifschitz. “What is Answer Set Programming ?” In : *Proceedings of the 23rd National Conference on Artificial Intelligence*. Sous la dir. d’AAAI Press. T. 3. 2008, p. 1594-1597.

Annexe 1 : Structure d'accueil

La structure d'accueil est le laboratoire IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires) de Rennes qui se situe sur le campus de Beaulieu. C'est un laboratoire qui comporte des équipes spécialisées dans : la bio-informatique, la sécurité des systèmes, les nouvelles architectures logicielles, la réalité virtuelle, l'analyse des masses de données et l'intelligence artificielle. Le directeur actuel est M. Guillaume Gravier.

CAMPUS DE BEAULIEU IRISA/INRIA RENNES
263 avenue du Général Leclerc
35 042 RENNES cedex
Entrée par le Bâtiment 12 F IRISA/Inria, allée Jean Perrin
Tel : +33 (0)2 99 84 71 00
Site internet : `https://www.irisa.fr/`

J'ai effectué mon stage parmi l'équipe DYLISS, dont le responsable est M. Olivier Dameron. C'est une équipe de recherche en bioinformatique sur l'automatisation de l'utilisation et de la découverte de connaissances formelles dans les sciences de la vie. L'équipe se consacre essentiellement à des applications à propos de la génomique et protéomique fonctionnelle grâce à leurs recherches sur l'analyse de séquence, les systèmes biologiques et le Web Sémantique.

Site internet : `https://www-dyliss.irisa.fr/`

Annexe 2 : Bilan personnel

Je me suis senti impliqué dans la recherche, j'ai eu l'occasion de prendre mes propres initiatives et de les mener. C'était d'ailleurs satisfaisant de pouvoir les mettre en place et d'interpréter mes résultats. J'ai pu abordé des thématiques qui me plaisent comme la théorie des graphes. Enfin, j'ai pu comprendre l'intérêt du Web Sémantique et la façon dont il a été conceptualisé.

Identification systématique de modules dans une collection de requêtes SPARQL

Résumé

La vitesse à laquelle les données scientifiques s'accumulent n'a cessé de croître, au point que les manières de les stocker et de les interroger ont dû s'adapter. Le Web Sémantique a pour but de rendre le Web accessible aux programmes et non plus uniquement aux humains. Il vise à établir de nouvelles connaissances à partir de données existantes. Ainsi, les graphes RDF et le langage de requête SPARQL ont été développés pour répondre à ces nouveaux enjeux. Cependant, un problème survient, il est nécessaire de connaître la structure RDF des données et le langage SPARQL afin de pouvoir les interroger. Or cela rend la tâche pénible pour un scientifique qui ne connaît pas ces technologies et cela freine l'utilisation de technologies adaptées pour les données du monde du vivant. C'est pourquoi ce stage se positionne sur l'identification systématique de modules, morceaux de code SPARQL qui peuvent s'assembler pour former n'importe quelle requête. Pour ce faire, nous utilisons une approche par l'extraction de modules en tant que sous-graphes communs maxima induits. L'extraction montre des résultats intéressants mais nécessitent d'être analysés avec la combinatoire afin de garder les meilleurs modules. Cela permettrait de déployer des interfaces proposant la génération automatique de code SPARQL et ainsi proposer des requêtes qui pourraient directement être soumise ou des structures de requêtes bien établies pour des utilisateurs confirmés.

Mots-clefs : composition de requêtes, Web Sémantique, RDF, SPARQL, neXtProt

Systematic identification of modules in a collection of SPARQL queries

Abstract

The speed at which scientific data accumulates has continued to increase, to the point that the ways of storing and querying it have had to adapt. One aspect of the Web is the Semantic Web, which aims to create new knowledge from existing ones. RDF graphs and the SPARQL query language were developped to meet these new challenges. However, you have to know the RDF structure of the data and the SPARQL language in order to craft new queries. This makes the task difficult for a scientist unfamiliar with these technologies and hinders the use of suitable technologies for life science data. That is why this internship focuses on the systematic identification of modules, pieces of SPARQL code that can be assembled to build any query. To do this, we use an approach based on the extraction of modules as maximum common induced subgraphs. The extraction shows interesting results, but needs to be analyzed with combinatorics to retain the best modules. This would enable the deployment of interfaces that allow for the automatic generation of SPARQL code and thus propose queries that could be submitted directly, or well-established query structures for experienced users.

Keywords : query composition, Semantic Web, RDF, SPARQL, neXtProt