Le Linked Data à l'université : la plateforme LinkedWiki

Karima Rafes

université PARIS-SACLAY

NNT : 2019SACLS032

Thèse de doctorat

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Le Linked Data à l'université
la plateforme LinkedWiki

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : réseaux, information et communications

Thèse présentée et soutenue à Gif-sur-Yvette,
le 25 janvier 2019, par

Karima Rafes

Composition du jury :

Philippe Pucheral Professeur, Université de Versailles-Saint-Quentin-en-Yvelines, Inria	Président
Cédric Du Mouza Maître de conférences HDR, Cnam Paris — Cédric	Rapporteur
Dan Vodislav Professeur, U. Cergy-Pontoise — ETIS	Rapporteur
Khalid Belhajjame Maître de conférences, Université Paris Dauphine — LAMSADE	Examinateur
Anne Doucet Professeure, Sorbonne Université — LIP6	Examinatrice
Sarah Cohen-Boulakia Professeure, U. Paris-Sud,U. Paris-Saclay — LRI	Directrice de thèse
Serge Abiteboul Directeur de Recherche Inria, ENS Paris	Co-directeur de thèse

Thèse de doctorat

de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

Le Linked Data à l'université : la plateforme LinkedWiki

Karima Rafes

supervisée par

Sarah Cohen-Boulakia
Professeure
Université Paris-Sud & Paris-Saclay — LRI

Serge Abiteboul
Directeur de Recherche Inria, ENS Paris

27 janvier 2019

Résumé

Le Center for Data Science de l'Université Paris-Saclay a déployé une plateforme compatible avec le Linked Data en 2016. Or, les chercheurs rencontrent face à ces technologies de nombreuses difficultés. Pour surmonter celles-ci, une approche et une plateforme appelée LinkedWiki, ont été conçues et expérimentées au-dessus du cloud de l'université (IAAS) pour permettre la création d'environnements virtuels de recherche (VRE) modulaires et compatibles avec le Linked Data. Nous avons ainsi pu proposer aux chercheurs une solution pour découvrir, produire et réutiliser les données de la recherche disponibles au sein du Linked Open Data, c'est-à-dire du système global d'information en train d'émerger à l'échelle du Web. Cette expérience nous a permis de montrer que l'utilisation opérationnelle du Linked Data au sein d'une université est parfaitement envisageable avec cette approche. Cependant, certains problèmes persistent, comme (i) le respect des protocoles du Linked Data et (ii) le manque d'outils adaptés pour interroger le Linked Open Data avec le langage SPARQL. Nous proposons des solutions à ces deux problèmes.

Afin de pouvoir vérifier le respect d'un protocole SPARQL au sein du Linked Data d'une université, nous avons créé l'indicateur SPARQL Score qui évalue la conformité des services SPARQL avant leur déploiement dans le système d'information de l'université.

De plus, pour aider les chercheurs à interroger le Linked Open Data, nous avons implémenté le démonstrateur SPARQLets-Finder qui facilite la conception de requêtes SPARQL à l'aide d'outils d'autocomplétion sans connaissance préalable des schémas RDF au sein du Linked Open Data.

Abstract

The Center for Data Science of the University of Paris-Saclay deployed a platform compatible with Linked Data in 2016. Because researchers face many difficulties utilizing these technologies, an approach and then a platform we call LinkedWiki were designed and tested over the university's cloud (IAAS) to enable the creation of modular virtual search environments (VREs) compatible with Linked Data. We are thus able to offer researchers a means to discover, produce and reuse the research data available within the Linked Open Data, i.e., the global information system emerging at the scale of the Internet. This experience enabled us to demonstrate that the operational use of Linked Data within a university is perfectly possible with this approach. However, some problems persist, such as (i) the respect of protocols and (ii) the lack of adapted tools to interrogate the Linked Open Data with SPARQL. We propose solutions to both problems.

In order to be able to verify the respect of a SPARQL protocol within the Linked Data of a university, we have created the SPARQL Score indicator which evaluates the compliance of the SPARQL services before their deployments in a university's information system.

In addition, to help researchers interrogate the Linked Open Data, we implemented a SPARQLets-Finder, a demonstrator which shows that it is possible to facilitate the design of SPARQL queries using autocompletion tools without prior knowledge of the RDF schemas within the Linked Open Data.

Remerciements

Je remercie mes parents, mes sœurs et frères, qui m'ont toujours soutenue dans ma volonté de devenir docteure en informatique.

Je remercie Jean Rohmer, qui m'a fait confiance pour enseigner dans son école en 2011 et m'a permis de présenter mes recherches à Cécile Germain, que je salue également, car, quelques années plus tard, c'est grâce à elle que cette thèse a pu voir le jour.

Je remercie aussi chaleureusement Sarah Cohen-Boulakia, ma directrice de thèse, et Serge Abiteboul, mon co-encadrant, qui m'ont aidée à terminer cette thèse. Cette expérience avec eux influencera profondément la direction de mes futurs travaux de recherche.

Je voudrais remercier particulièrement Sana Tfaili, maître de conférences à la faculté de pharmacie en chimie analytique de Paris-Sud, qui a eu le courage d'accepter d'être la première à expérimenter de bout en bout l'approche décrite dans cette thèse, afin d'améliorer les conditions de travail de son équipe, mais également de tous ses collègues au sein de son université.

Évidemment, je remercie tout autant Julien Nauroy, Balazs Kegl, Michèle Sebag, Marc Schoenauer, Guillaume Philippon, Catherine Delplanque, Valérie Cantonny, Denis Humbert, et tous les chercheurs, ingénieurs ou techniciens de recherche, ainsi que mes étudiants, qui ont soutenu ou participé aux expériences décrites dans cette thèse.

Pour terminer, je tiens à remercier tous les internautes, comme les Wikipédiens, qui participent, par leurs actions, à construire un Web non mercantile au service de tous. Ce sont eux qui donnent un sens à mes travaux et qui m'insufflent l'énergie pour les poursuivre depuis près de dix ans. Ma thèse n'est qu'une modeste contribution face à l'ampleur du travail qu'il nous reste à parcourir pour construire le Web dont nous avons besoin, afin de mieux vivre ensemble sur cette toute petite planète.

Table des matières

Chapitre 1

Introduction

Le Center for Data Science de l'Université Paris-Saclay a décidé de se doter d'une plateforme compatible avec le *Linked Data* à destination des chercheurs. J'ai réalisé cette plateforme dans le cadre d'un accord entre mon entreprise, BorderCloud, et l'Université Paris-Sud (UPS). Les logiciels que j'ai développés sont maintenus par BorderCloud.

Après le déploiement en 2016 de la première version de cette plateforme, nous avons analysé au travers de cette thèse les problèmes que les chercheurs rencontrent face à ces technologies. Nous avons ensuite pu proposer et expérimenter une approche globale pour faciliter le travail des chercheurs au sein du Linked Data. Cette approche, nommée *LinkedWiki*, préfigure ce que pourraient être les systèmes d'information nécessaires au fonctionnement du *Web sémantique* au sein des universités.

1.1 Le Web sémantique à l'Université

Après le succès du Web, le concept de Web sémantique [17] a été diffusé auprès de la communauté scientifique en 2001 par le W3C. Le premier objectif était d'aider les scientifiques à accélérer leurs recherches en leur offrant des logiciels capables de retrouver et de réutiliser les connaissances disponibles. Dix-sept ans plus tard, l'expression « Web sémantique » est toujours d'actualité mais les avis sont partagés sur la façon de l'implémenter. Pour les uns, le Web sémantique doit être construit autour de technologies permettant à la machine de comprendre le langage humain afin d'en extraire le sens pour le réutiliser. D'autres, comme les inventeurs du Web sémantique, pensent que sa mise en œuvre doit d'abord passer par la mise à disposition d'informations reliées, mieux définies et interprétables sans ambiguïté par les humains et les machines [94].

En 2006, afin de dissocier ces deux visions du Web sémantique, on nomma Linked Data [15] les données du Web sémantique qui seront correctement structurées et liées entre elles afin que les machines puissent un jour les utiliser directement. Malheureusement, cela n'a pas suffi à lever cette confusion entre la construction d'un Web interprétable par les machines et un Web dont le sens de son contenu sera accessible aux machines. Et en français, cette confusion est même amplifiée par la traduction tronquée du Linked Open Data (la version accessible par tous du Lin-

ked Data), devenu le Web des données, au lieu de le traduire, par exemple, par le Web des données liées. Pour éviter cette confusion dans cette thèse, nous utiliserons uniquement le terme Linked Data pour parler des données liées.

À cause de cette confusion persistante et du peu de données structurées mises à disposition depuis onze ans, de nombreux scientifiques sont arrivés à la conclusion qu'il n'était pas possible d'obtenir naturellement cette masse critique de données structurées indispensables pour faire naître ces nouveaux systèmes promis par le Web sémantique et qu'il fallait donc travailler en priorité sur l'extraction automatique du sens contenu au sein des documents du Web pour constituer cette masse critique. Cette approche n'est pas contradictoire avec l'approche d'origine, inchangée depuis 2001, qui consiste à aider et encourager les producteurs de connaissances à mieux partager leurs données afin de construire le Web sémantique. Ces deux approches sont complémentaires, bien que les techniques nécessaires pour les mettre en œuvre soient très différentes [94].

Du point de vue d'une communauté, comme une université, la mise en œuvre du Web sémantique au travers d'informations directement interprétables sans ambiguïté par les machines a au moins trois avantages. Le premier avantage est de mieux préserver, transmettre et faciliter la reproductibilité des travaux de recherche au travers des technologies du Linked Data. Le deuxième avantage est d'alimenter des algorithmes d'intelligence artificielle avec des données plus fiables. Une dernière raison est que ces communautés ne souhaitent plus payer pour accéder à leurs propres connaissances sur les réseaux [39] et voient dans le Linked Data une opportunité de s'affranchir des éditeurs scientifiques qui prennent également le virage du Web sémantique [160] en devenant des *data brokers* (fournisseurs/revendeurs des données pour la recherche).

Le travail que relate cette thèse s'inscrit dans cette volonté de mettre en œuvre le Linked Data au sein d'une université au service de la recherche.

1.2 Dans l'écosystème de l'Université Paris-Saclay

Bien que les universités soient très semblables en France dans leurs organisations, le cas de l'Université Paris-Saclay présente des particularités. Tout d'abord, elle est née en 2014 avec l'ambition de regrouper de nombreux établissements parmi les plus réputés en France, dont trois universités. Ces établissements se sont fédérés au sein de l'Université Paris-Saclay, en mutualisant les formations et une recherche au plus haut niveau international. L'un des chantiers, toujours en cours, est la volonté de créer une synergie de la recherche entre les établissements. Cette synergie a pris plusieurs formes. Dans le domaine de la science des données, c'est le CDS (*Center for Data Science*) qui a pour mission d'offrir des passerelles entre les laboratoires afin de faciliter le travail entre les chercheurs qui produisent et/ou qui consomment de la donnée.

Cette thèse s'inscrit dans le cadre de l'une des missions du CDS qui consiste à expérimenter le Linked Data avec des chercheurs de différents domaines. Nous avons ainsi travaillé avec le laboratoire Lip(Sys)^2 de la Faculté de Pharmacie, le laboratoire de Droit et Sociétés Religieuses de la Faculté de Droit ainsi que le La-

boratoire Atmosphères, Milieux, Observations Spatiales (LATMOS) de l'Institut de recherche IPSL, l'équipe en Intelligence Artificielle TAO (Inria-Saclay et LRI, Laboratoire en Recherche Informatique) et l'IPS2 (Institut des Sciences des Plantes de Paris-Saclay). De plus, cette expérimentation a lieu dans une université qui se caractérise par une organisation décentralisée et une volonté de mutualiser son infrastructure informatique, ce qui peut sembler antinomique. En réalité, nous allons voir que toutes ces différences et ces contradictions sont des atouts dans la perspective d'évaluer les nouvelles technologies du Linked Data.

Systèmes d'information hétérogènes. L'un des problèmes souvent rencontrés par les chercheurs qui manipulent des données est le problème de la conversion des données entre le format des producteurs et le format des consommateurs.

Pour y répondre :

— Les chercheurs sont souvent livrés à eux-mêmes.

— Les chercheurs bénéficiant de financements peuvent embaucher un ou des informaticiens pour réaliser une partie du travail. Ils ont rarement les moyens de payer pour un système d'information pérenne.

— Certains laboratoires travaillent sur des installations plus lourdes afin de traiter les données de la recherche au sein de leur domaine, mais ils sont rares.

Dans ce contexte, la mission du Center for Data Science prend tout son sens afin de partager l'expérience des laboratoires et des chercheurs qui savent déjà traiter la donnée avec ceux qui peinent encore à le faire.

Écosystème décentralisé sinon rien. Chacun des établissements dans l'université tient à rester libre dans ses recherches et dans la manière de les mener. Cela implique que la mise en œuvre d'une solution rigide pour la gestion des données est exclue. Ce problème de gouvernance n'est pas nouveau, mais les acteurs du Web ont su contourner ce problème en inventant le moyen de partager des documents de manière décentralisée, laissant ainsi aux acteurs la liberté de diffuser ou non leurs documents, comme ils le souhaitent. Cependant, il a fallu une vingtaine d'années pour y parvenir. Aujourd'hui, il n'existe pas vraiment de solution décentralisée pour la gestion des données de la recherche. La solution qui s'en rapproche le plus est le Linked Data, qui servira de système d'information aux logiciels du Web sémantique. Malheureusement, ce système d'information global décentralisé pour la gestion, l'interrogation et la traçabilité de données massives n'est pas totalement disponible. Pour le moment, les établissements mettent chacun en place leurs propres solutions pour gérer leurs données en restant conscients que ces solutions sont temporaires et devront converger pour faciliter leur travail de recherche.

Mutualisation des infrastructures. L'évolution technologique vers le calcul à haute densité rend obsolète l'informatique physiquement dispersée. Dans le cadre d'une action de l'Université Paris-Sud visant à une meilleure structuration de l'informatique pour la science, une expérimentation sur l'adoption du modèle de production *Cloud* a été conduite de façon systématique. Elle a utilisé l'infrastructure avancée *Virtual Data*, disponible sur le campus d'Orsay, et développée par le Laboratoire de l'Accélérateur Linéaire (LAL IN2P3 du CNRS), qui est un gros consommateur de temps de calcul, et qui a, en particulier, mis en œuvre un Cloud et des services

afférents. Le LAL a mis son expertise en exploitation informatique à la disposition de l'université pour cette action, en particulier en direction des laboratoires qui souhaitent accéder à cette technologie de façon transparente tout en restant dans un contexte académique.

En faisant la somme des caractéristiques de cette nouvelle université, on obtient un écosystème définitivement décentralisé où il existe des milliers de consommateurs et de producteurs de données avec la possibilité de construire un futur système d'information mutualisé. L'Université Paris-Saclay rassemble ainsi tous les ingrédients permettant d'expérimenter la mise en oeuvre d'une véritable solution pour la gestion de données scientifiques afin de faciliter le travail des chercheurs.

Pour lier tous ces ingrédients, l'université a demandé au CDS de travailler à la création de ce futur système d'information où l'utilisation des technologies du Linked Data était une évidence.

1.3 Objectifs

Nos objectifs étaient les suivants :

— Implémenter une plateforme de partage de données compatible avec le Linked Data pour faciliter la réutilisation des données au sein de l'université et ainsi encourager la synergie entre les producteurs et les consommateurs de données.

— Faciliter la réutilisation des données massives des laboratoires au travers des infrastructures Cloud et de la grille de calcul que l'Université met à disposition des chercheurs afin de permettre la reproductibilité de leurs résultats.

1.4 Contributions

Après le chapitre 2 qui présente un état de l'art du Linked Data et des pratiques des chercheurs que nous avons côtoyés, nos contributions sont décrites dans trois chapitres :

— Le chapitre 3 présente l'approche LinkedWiki, qui vise à offrir une solution pragmatique pour permettre aux chercheurs de travailler avec un système d'information compatible avec le Linked Data.

— Le chapitre 4 considère le problème de l'interrogation des données du Linked Data au sein des universités. Notre solution propose un mécanisme d'autocomplétion de requêtes SPARQL basé sur un algorithme de *clustering* hiérarchique.

— Le chapitre 5 adresse le problème du respect de l'interopérabilité des services SPARQL au sein du Linked Data. Nous introduisons un indicateur du niveau d'interopérabilité calculé avec une série de plus de 500 tests automatiques et une méthode d'alignement des protocoles des services SPARQL déployés au sein d'un système d'information.

Le chapitre 6 conclut ce travail.

Résultats publiés durant cette thèse

Articles publiés :
[117] « Designing scientific SPARQL queries using autocompletion by snippets ». Karima Rafes, Serge Abiteboul, Sarah Cohen-Boulakia et Bastien Rance. Dans : eScience, 2018 IEEE 14th International Conference on. IEEE. 2018.
[120] « Certifying the interoperability of RDF database systems ». Karima Rafes, Julien Nauroy et Cécile Germain. Dans : LDQ 2015-2nd Workshop on Linked Data Quality. 2. Springer. 2015.

Démonstrations publiées :
[118] « Une autocomplétion générique de SPARQL dans un contexte multi-services », Karima Rafes, Sarah Cohen-Boulakia et Serge Abiteboul. Dans BDA. 2017.
[119] « A platform for scientific data sharing ». Karima Rafes et Cécile Germain. Dans : BDA (Bases de Données Avancées). 2015.
[121] « TFT, Tests For Triplestores ». Karima Rafes, Julien Nauroy et Cécile Germain. Dans : Semantic Web Challenge, part of the International Semantic Web Conference. 2014.

Autres travaux

[96] Proposition de projet européen de recherche (H2020) : « Enabling Open Science : Wikidata for Research. ». Daniel Mietchen, Gregor Hagedorn, Karima Rafes et al. 2015.
[140] Poster : « Data acquisition for analytical platforms : Automating scientific workflows and building an open database platform for chemical analysis metadata ». Sana Tfaili, Diem Bui Thi, Karima Rafes et al. Dans Chimiométrie XVII. Poster. 2016.

Workshops :
[56] « Transforming Wikipedia into an Ontology based Information Retrieval Search Engine for Local Experts usinga Third-Party Taxonomy ». Gregory Grefenstette et Karima Rafes. Dans : Workshop on Language and Ontology & Terminology and Knowledge Structures LO2TKS. 2016.
[52] « The Grid Observatory 3.0 - Towards reproducible research and open collaborations using semantic technologies. ». Cécile Germain, Julien Nauroy et Karima Rafes. Dans : EGI Community Forum 2014. Mai 2015.

Chapitre 2

Le Linked Data

Sommaire

2.1 Introduction

Dès 2006, des bonnes pratiques [15, 14, 134] encadrées par le W3C sont proposées afin de construire un système global d'information pour faciliter la réutilisation des informations contenues au sein des documents sur le Web. Ce système d'information à l'échelle du Web qui commence à émerger se nomme le *Linked Data*, et sa partie publique se nomme le *Linked Open Data* [19].

La finalité du Linked Data est de devenir le système d'information du Web sémantique [15, 14, 134] et en 2018, il existe 84 recommandations du W3C qui concernent de près ou de loin le Web sémantique et/ou le Linked Data [155]. Certaines évoquent les bonnes pratiques concernant les protocoles d'interrogation des données, la manière de décrire les informations, la façon de relier les informations, d'améliorer la fiabilité des données, etc. Cela correspond à plusieurs centaines de bonnes pratiques à appliquer « en théorie ».

Dans ce chapitre, après un état de l'art du Linked Data au travers de son histoire, nous définissons ce que devrait être le Linked Data à l'université. Nous avons ainsi sélectionné onze bonnes pratiques parmi des centaines afin de clarifier les objectifs

du Linked Data et ainsi simplifier son déploiement au sein d'une université, afin que les chercheurs bénéficient plus rapidement de ces nouvelles technologies.

Pour mettre en œuvre ces bonnes pratiques au sein de l'Université Paris-Saclay, nous avons collaboré avec le laboratoire Lip(Sys)^2 de la Faculté de Pharmacie, le laboratoire de Droit et Sociétés Religieuses de la Faculté de Droit et le Laboratoire Atmosphères, Milieux, Observations Spatiales (LATMOS) de l'Institut de recherche IPSL. Nous avons ainsi pu observer et rassembler une partie des difficultés que les chercheurs rencontrent face à ces technologies.

Dans ce chapitre, nous décrivons le Linked Data au travers de son histoire (en 2.2) et de ses objectifs (en 2.3) afin de sélectionner les bonnes pratiques du W3C qui nous semblent indispensables afin de construire le Linked Data à l'université. Ensuite, au travers des pratiques des chercheurs (en 2.4), nous identifions les potentielles données de la recherche que ces bonnes pratiques permettront de structurer et lier, afin d'aider les chercheurs dans leurs activités. Puis, nous évoquons les freins et les verrous observés face au Linked Data dans les laboratoires avec lesquels nous avons collaboré (en 2.5). Enfin, nous concluons en précisant l'approche que nous utilisons afin de lever ces verrous au sein de cette thèse.

2.2 Historique

Nous retraçons l'histoire du Linked Data et de ses technologies en nous fondant sur l'article « Linked data : The story so far » [20].

Le Web a changé la façon dont nous partageons les connaissances en facilitant la publication et l'accès aux documents au sein d'un espace mondial d'information [16]. Les liens hypertextes permettent de parcourir cet espace d'information à l'aide de navigateurs Web, tandis que les moteurs de recherche indexent les documents, analysent la structure des liens [26] et observent nos comportements pour déterminer les documents les plus pertinents [59]. Ces fonctionnalités ont été rendues possibles par l'ouverture et l'extensibilité du Web [69], éléments clés de sa croissance [13].

Malgré les avantages indéniables que le Web apporte, jusqu'à récemment, les principes qui ont permis au Web des documents de s'épanouir n'ont pas été appliqués aux données. Encore aujourd'hui, les données publiées sur le Web sont plus souvent disponibles sous forme de fichiers bruts dans des formats tels que CSV ou XML, ou encore comme des tableaux HTML, sacrifiant ainsi une grande partie de leur structure et de leur sémantique. Dans le Web traditionnel, la nature de la relation entre deux documents liés est explicite uniquement pour l'être humain, car pour une machine, le format de données, c'est-à-dire le HTML4, n'est pas suffisamment expressif pour permettre aux entités décrites dans différents documents d'être explicitement connectées.

Cependant, de 2001 à 2011, le Web a évolué d'un espace mondial d'information de documents liés à un espace où documents et données sont liés. Cette évolution repose sur un ensemble de bonnes pratiques relatives à la publication et la structuration des données sur le Web, appelées *Linked Data* [15, 14]. L'adoption de ces bonnes pratiques a permis d'étendre le Web à un espace mondial de données reliant les

données de divers domaines [133], telles que les données associées aux publications et aux revues scientifiques, les données relatives aux médicaments et essais cliniques, les données statistiques, etc. Ces bonnes pratiques étaient portées par l'ambition de mettre en œuvre le Web sémantique explicité en 2001 [17] avec l'objectif d'accélérer la recherche en offrant aux chercheurs des machines en mesure de les aider à exploiter au mieux cette masse d'informations disponibles.

En appliquant les bonnes pratiques du Linked Data aux données partagées sur le Web [20, 19], les données structurées ont commencé à se lier non pas à des documents mais directement à d'autres données, en faisant directement référence à leurs identifiants [40]. Ce réseau global d'information publique naissant grâce au Linked Data a pris le nom du *Linked Open Data* (ou LOD) [19].

Dès 2006, pour faciliter la réutilisation de données structurées, les bonnes pratiques du Linked Data sont devenues progressivement des spécifications techniques précises au travers du W3C [134, 83, 25, 141, 84, 137].

La finalité des spécifications du Linked Data est de permettre aux applications de partager un espace de données accessibles via Internet, le LOD. Le LOD permettra de compléter les réponses des applications au fur et à mesure que de nouvelles sources de données apparaîtront dans son espace [20].

Aujourd'hui, le Linked Data désigne également la somme des spécifications techniques nécessaires pour mettre en oeuvre ce système global d'information publique qui se nomme le *Linked Open Data* (ou LOD). Cet espace d'information est maintenant une réalité [133] au travers de services tels que Wikidata, la base des données factuelles de Wikipédia [41] ou la base de connaissances de l'Institut européen de Bio-Informatique (EBI) [76]. Le LOD offre des données fraîches et de plus en plus fiables, rassemblant ainsi les conditions pour passer à l'étape suivante, c'est-à-dire concevoir des outils à destination des chercheurs [130] pour réutiliser l'ensemble des informations disponibles et ainsi les aider à créer plus rapidement de nouvelles connaissances.

Dans la section suivante, nous décrivons les technologies nécessaires pour construire au sein d'une université un système d'information en mesure de faire partie de cet espace global.

2.3 Les bonnes pratiques

Dans cette section, nous allons expliciter les bonnes pratiques en vigueur en 2018 qui visent à faire émerger un système d'information global et fiable.

Pour permettre à des machines d'appliquer des raisonnements fiables sur la base d'un système global d'information, il faut au préalable s'assurer que les informations qu'utilisent ces machines soient également fiables.

La fiabilité d'une information désigne souvent le degré de confiance qu'accorde un humain à cette information afin d'exécuter une tâche précise. Cette fiabilité dépend d'éléments interdépendants, tels que l'identification claire de l'origine de cette information ou sa cohérence par comparaison aux autres informations sur le

même sujet, et de critères en relation avec la tâche visée par celui qui va utiliser cette information [125]. Pour un chercheur dont la tâche est de créer de nouvelles connaissances à partir des précédentes, la validité des connaissances dépend aussi de leur fraîcheur afin de prendre en compte les connaissances les plus récentes.

Pour assurer la fiabilité des données au sein du Web sémantique, Tim Berners-Lee [15, 14, 134] propose un ensemble de « règles » relativement simples pour démarrer la constitution d'un système d'information à l'échelle du Web, pour faciliter la réutilisation des informations et pour pouvoir opérer des processus permettant de s'assurer de leur fiabilité.

A partir de ces premières règles, le W3C a pris le relais et recommandé officiellement de nombreuses technologies ; chacune d'entre elles repose sur une ou plusieurs bonnes pratiques qui permettraient de faire émerger un système global d'information fiable pour permettre au Web sémantique de fonctionner.

Parmi toutes les bonnes pratiques dispersées dans les recommandations officielles du W3C à propos du Linked Data (environ 84 recommandations depuis douze ans [155]), nous en avons sélectionné onze qui nous semblent indispensables pour répondre aux conditions minimales d'obtention d'un système d'information fiable dans un contexte décentralisé comme celui d'une université.

Comme nous l'avons vu, pour déterminer la fiabilité d'une information, il faut permettre la comparaison des informations pour faciliter la détection des contradictions par l'utilisateur de ces informations. Ainsi, pour qu'une machine ou un humain puisse comparer une information, il faut leur permettre de naviguer d'information en information (de lien en lien) et dans un format lisible pour qu'ils puissent les comparer. Voici les bonnes pratiques que nous avons sélectionnées pour y parvenir :

1. **Utiliser des IRI HTTP pour identifier les *entités* composants les informations décrites sur le Web** [12].

   IRI ou *Internationalized Resource Identifier* est une norme (RFC 3987) qui généralise les normes précédentes comme les URL et les URI. Ainsi, une adresse HTTP, une adresse email (avec mailto :) et d'autres formes d'identifiants sont également des IRI.

   Les IRI sont absolus ou relatifs comme pour les URL. Pour obtenir les *IRI absolus* à partir d'*IRI relatifs*, les *préfixes* des IRI relatifs doivent être préalablement définis.

2. **Représenter les informations en RDF avec le vocabulaire RDFS** [83].

   Le modèle RDF relie les entités par des arcs orientés afin de former un grand graphe d'information. Un arc de ce graphe est un *triplet* (*triple*, en anglais) qui prend la forme « sujet-propriété-objet » ou « sujet-propriété-valeur ». Un sujet, une propriété et un objet correspondent à des entités qui sont substituées par leurs IRI au sein du triplet. Lorsque la fin d'un triplet est une « valeur », il s'agit d'une chaîne de caractères, d'une date, d'un nombre, etc.

   On nomme « *vocabulaire* » en RDF la manière de décrire une information à l'aide de classes et d'instances en s'appuyant sur le vocabulaire RDFS [83]. On nomme « *ontologie* » les descriptions RDF plus complexes qui s'appuient sur les recommandations OWL du W3C et ses déclinaisons (OWL-Lite/DL/Full,

OWL2, etc.) [103]. Quand on parle de « *schémas RDF* », on fait référence à des ontologies ou à des vocabulaires.

3. **Relier les informations de sources différentes** [83].
   Si une autre source d'information évoque la même information, il faut essayer d'utiliser les mêmes entités (les mêmes IRI) quand c'est possible ou au moins décrire la relation entre ces entités similaires en RDF. Avec ces liens, une machine ou un humain peut naviguer entre les différentes sources d'information.
4. **Décrire les métadonnées en RDF des documents non RDF sur le Web** [137].
   La recommandation de 2015 du *Linked Data Platform* (LDP) a été conçue pour normaliser l'association de données RDF décrivant des documents non RDF sur le Web. De nouveaux services Web apparaissent afin d'héberger des documents non RDF dont leurs métadonnées sont automatiquement extraites et partagées avec le protocole LDP [65].
5. **Retourner un document décrivant une entité quand on adresse son IRI dans une requête HTTP. Le document obtenu doit être lisible par la machine ou l'humain à l'origine de la requête HTTP [15].**
   Le format du document sera en :
   - HTML quand le document RDF n'est pas explicitement demandé dans la requête HTTP car on suppose alors que c'est un humain qui utilise un navigateur Web pour comparer l'information en naviguant d'information en information (d'IRI en IRI) [131].
   - RDF/XML [9] ou Turtle [10, 83], JSON-LD [75] ou bien encore en HTML avec RDFa [45, 3]. Une machine doit explicitement demander le format du document RDF qu'elle peut supporter, dans la requête HTTP.
     Ces formats ont aussi été conçus pour être lisibles par des humains afin de permettre aux informaticiens d'écrire/corriger du RDF. Le format Turtle est le format le moins verbeux d'entre eux. Il est souvent utilisé pour fabriquer des suites de tests [153], partager des exemples en RDF dans les documentations et les recommandations du W3C (par exemple [84]).

Le degré de fiabilité d'une information est déterminé entre autres par son origine. Il est nécessaire de connaître la source d'une information, même si cette information est diffusée par une autre source. Pour y parvenir au sein des recommandations officielles et des notes des groupes de travail du W3C, nous avons sélectionné les bonnes pratiques suivantes :

6. **Créer des IRI avec le nom de domaine de la source qui a créé ces informations** [131].
   Le protocole DNS [64] permet d'octroyer un nom de domaine « pérenne » à un être humain ou à une organisation qui diffuse des informations. Une source devrait utiliser son nom de domaine dans toutes les nouvelles IRI dont elle aura besoin pour diffuser de nouvelles informations.
   Malheureusement, on a constaté que la stabilité des informations au travers du LOD étaient encore à améliorer et qu'elle baisse avec le temps [148]. Cette

bonne pratique ne peut donc s'appliquer que pour des organisations en mesure de maintenir seules et dans la durée leurs informations au sein du LOD.

Pour résoudre le problème de stabilité tout en garantissant l'origine des informations, des services comme Wikidata [41] ou DBpedia [91] sont apparus pour pouvoir partager des informations factuelles au sein du LOD avec des IRI stables associés à leurs références afin d'en déterminer leurs origines.

7. **Préserver les IRI d'origine dans un cadre décentralisé [30].**
   Les informations au sein d'un document RDF collecté puis diffusé doivent conserver l'IRI d'origine de ce document RDF.
   Pour ce faire, au sein des bases de données RDF, les documents RDF sont généralement sauvegardés dans des *graphes nommés* (en anglais, *graph named*) avec l'adresse d'origine des documents sur le Web [30] (avec un service SPARQL, on peut appeler le chargement d'un document RDF dans un graphe nommé avec la fonction `LOAD` par exemple).

Pour appliquer le critère de fiabilité qui détermine si une information est récente/fraîche, nous avons sélectionné cette bonne pratique du W3C :

8. **Retourner à partir de l'IRI d'une entité la dernière version du sous-graphe global d'information qui s'y rapporte.**
   Cette bonne pratique appliquée à la bonne pratique n°4 précise que le contenu du document RDF ou HTML ne devrait contenir que les dernières informations concernant cette entité.
   De plus, de nombreuses bases de données RDF utilisent ce mécanisme d'accès aux documents RDF à partir d'une IRI pour se mettre à jour [138, 102, 72].

Afin d'évaluer la fiabilité d'une information avant de l'utiliser pour certaines tâches, il est nécessaire d'interroger précisément et simultanément plusieurs sources d'information. Nous avons sélectionné cette bonne pratique du W3C pour y parvenir :

9. **Utiliser SPARQL pour interroger de manière transversale toutes les sources d'information** [141].
   Le W3C recommande d'associer un service SPARQL aux bases de données RDF, services LDP (Linked Data Platform) et tout autre conteneur de documents RDF afin de pouvoir interroger les données avec le langage SPARQL au travers du protocole SPARQL (basé sur HTTP). Une requête SPARQL est basée sur un patron de graphes RDF qui s'applique au graphe auquel le service SPARQL a accès [141].
   Cette technologie permet en théorie l'écriture d'une *requête fédérée* qui servira à interroger simultanément plusieurs services SPARQL. Même si les *requêtes fédérées* posent quelques problèmes (comme nous le verrons dans le chapitre 5), les bases de données RDF utilisent déjà largement des requêtes SPARQL pour permettre à leurs applications clientes de consulter et mettre à jour les données RDF (dans des architectures trois tiers) [51, 22, 138, 102, 73].
10. **Décrire les conditions de validation des graphes RDF en SHACL** [84].
    Le W3C a recommandé, en 2017, SHACL (*Shapes Constraint Language*) pour pouvoir décrire explicitement le schéma RDF attendu par les machines [84] (SHACL est pour le RDF l'équivalent du XML Schéma pour le XML [50]).

Cette technologie permet d'avertir les producteurs de données qu'ils ne respectent pas les conditions qu'ils ont eux-mêmes définies afin de pouvoir réutiliser leurs données avec une requête SPARQL précise. Quand un schéma RDF n'est pas respecté, l'écriture de requêtes SPARQL devient bien plus complexe et freine la réutilisation des données.

11. **Décrire les services SPARQL disponibles ainsi que l'ensemble des jeux de données réutilisables en RDF ou non avec le vocabulaire DCAT** [54, 93].

    DCAT (Data Catalog Vocabulary) est un vocabulaire RDF qui doit permettre le référencement et donc la découverte des données (RDF ou non) au sein du Linked Open Data. Cette bonne pratique permet aussi de référencer les services SPARQL disponibles et facilitera en principe la conception de requêtes fédérées.

Évidemment, la communauté du Web sémantique encourage l'utilisation de nombreuses autres bonnes pratiques au travers de nombreuses technologies.

Cependant, ces onze bonnes pratiques nous semblent déjà un objectif ambitieux qui devrait nous permettre de faire émerger le *Linked Data de l'université*, c'est-à-dire un système d'information décentralisé dont on pourra estimer la fiabilité des informations avant de les utiliser.

Une fois que le Linked Data de l'université sera fonctionnel, il sera possible de commencer à envisager le déploiement d'autres bonnes pratiques du W3C afin de faire émerger le Web sémantique.

Avec seulement quelques bonnes pratiques, il nous semble que la solution technique proposée par le W3C est maintenant suffisamment solide afin de constituer au niveau mondial un système global d'information au travers d'Internet. Cependant, dans les faits, les grandes organisations publiques ainsi que les chercheurs, ceux-là même les plus enclins à comprendre l'enjeu du Linked Data, tardent à le déployer et à l'utiliser. Dans la section suivante, pour comprendre ce décalage, nous allons étudier les pratiques des chercheurs et décrire les problèmes que nous avons rencontrés durant nos expérimentations lorsque nous avons essayé d'appliquer ces onze bonnes pratiques.

2.4 Les données structurées et liées de la recherche

Avant de déployer le Linked Data, il faut clairement identifier les processus et les données qui seront impactées durant la migration du système d'information qu'utilise actuellement le chercheur. Dans cette section, nous décrivons les pratiques scientifiques que nous avons observées au sein de quelques laboratoires de l'Université Paris-Saclay et la place du Linked Data dans la gestion de leurs données.

Besoins différents entre les chercheurs. Nous avons demandé à de nombreux chercheurs ce qu'ils désireraient améliorer dans la gestion de leurs données. Deux catégories de chercheurs sont apparues assez rapidement : les chercheurs parfaitement autonomes dans le traitement de leurs données, et ceux ayant besoin d'être

accompagnés. Un chercheur autonome n'a pas nécessairement envie d'alourdir le processus qu'il a mis en place lui-même et ne ressent pas la nécessité de changer ses méthodes pour le moment. Les chercheurs de la deuxième catégorie perçoivent que les méthodes qu'ils appliquent actuellement ne sont pas adaptées pour traiter la masse des données dont ils ont besoin pour travailler.

Nous nous sommes ainsi concentrée sur les chercheurs de la deuxième catégorie, qui ont besoin d'être accompagnés et d'accéder à des solutions concrètes rapidement. Ce sont généralement des scientifiques dans le domaine des *sciences expérimentales* qui collectent eux-mêmes leurs données au travers de machines (par exemple des spectromètres). De plus, ils doivent appliquer un processus expérimental souvent plus rigoureux et plus transparent que dans d'autres disciplines pour permettre la reproductibilité et la confirmation de leurs résultats, car ces résultats influencent, par exemple, la sortie de nouveaux médicaments ou les dosages dans certains traitements de maladies graves.

Dans les paragraphes suivants, nous allons décrire les différentes dimensions que recouvre le concept de processus scientifique.

Le processus scientifique. Le processus de recherche dans le domaine des sciences expérimentales [11] peut se résumer comme suit : les scientifiques (1) conçoivent une expérience, (2) réalisent cette expérience, (3) collectent des *données brutes*, (4) proposent une analyse, (5) comparent leur analyse à l'état de l'art, et (6) partagent leurs résultats, leurs nouvelles connaissances, si ceux-ci sont intéressants.

Le crédit scientifique. Quand c'est possible, les scientifiques souhaitent publier les nouvelles connaissances au sein de revues scientifiques ou sous forme de brevets pour les valoriser. Les scientifiques sont ainsi disposés à partager leurs données brutes et les résultats qu'ils produisent, mais, généralement, seulement sous certaines conditions pour conserver le bénéfice de leur effort de production de données comme un avantage concurrentiel dans la découverte scientifique.

Pour garder la trace de l'origine des travaux scientifiques, un outil particulièrement important est utilisé aux côtés des articles publiés. Il s'agit du *cahier de laboratoire* [124], qui est utilisé dans la plupart des laboratoires pour enregistrer les conditions expérimentales quotidiennes, les remarques et les résultats. Ces cahiers en papier contiennent des données cruciales pour permettre la reproductibilité des expériences, et constituent également des preuves difficilement falsifiables [68].

La prise en compte des contraintes liées au crédit scientifique influence profondément la gestion des données de recherche, comme nous allons le voir dans les paragraphes suivants.

L'environnement de recherche virtuel. La gestion des données des scientifiques nécessite l'utilisation de nombreux outils différents pour le traitement et l'analyse des données. Plus les expériences sont proches, plus les outils utilisés sont réutilisables entre chercheurs. Ainsi, avec la numérisation des méthodes de travail au sein de la recherche est apparu le concept d'environnement virtuel de recherche (en anglais, VRE pour *Virtual Research Environment*) [29]), qui vise à faciliter le travail durant les différentes étapes composant un *workflow* scientifique.

Un VRE est constitué d'un ensemble d'outils et vise à rendre réutilisables les

données entre les chercheurs dans l'objectif de faciliter les processus de collaboration afin d'accélérer leurs recherches.

Quand deux chercheurs utilisent chacun de leur côté le même VRE et qu'ils sont en mesure de réutiliser les mêmes données sans que cela nécessite un post-traitement sur ces données, on dit souvent que les données sont « interopérables au sein de ce VRE » [29].

De nombreux VRE sont conçus comme des systèmes de « bureau à distance » sécurisés et fermés à cause des contraintes liées au crédit scientifique. Ces VRE sont centrés sur les chercheurs et leurs logiciels. On y intègre juste assez d'interopérabilité pour transmettre les problèmes et les résultats d'un logiciel à un autre, facilitant le travail de leurs utilisateurs, mais sans pour autant faciliter la transmission des données à d'autres chercheurs [85].

Pour permettre la réutilisation des données à l'échelle du Web, les VRE doivent dissocier clairement la dimension métier que contient la gestion des données au sein d'un processus scientifique afin que ces données puissent être réutilisables par n'importe quelle machine, c'est-à-dire par n'importe quel chercheur.

Les différentes dimensions des VRE. Pour dissocier la dimension métier de la représentation des données au sein du processus scientifique, on peut distinguer trois dimensions au sein d'un VRE :

1. La **dimension logicielle** inclut tous les outils logiciels associés aux protocoles de recherche (des outils de collecte jusqu'aux outils d'analyse et de diffusion).
2. La **dimension infrastructure** inclut le matériel utilisé pour l'acquisition et l'analyse des données, par ex. la grille de calcul, le stockage de masse, ainsi que les dispositifs expérimentaux et les instruments de mesure.
3. La **dimension des données** inclut toutes les informations (au sein du Linked Data), processus de production, données nécessaires à la reproduction des mêmes analyses dès lors que l'on dispose des mêmes données expérimentales brutes, des mêmes logiciels et de la même infrastructure.

Nous allons maintenant faire un zoom sur la dimension des données.

Les différents types de données de la recherche. Durant notre collaboration, nous avons distingué six types de données qui peuvent être fabriquées, utilisées, déplacées et consultées dans le processus scientifique :

— Les **données expérimentales** :
  - **brutes** : ce sont les données d'acquisition avant filtrage du chercheur [90]. Toutes les données brutes sont rarement intégralement conservées car, par exemple, il y a eu des erreurs d'acquisition ou que leur stockage serait trop coûteux. Cependant, ces données se révèlent généralement les plus difficiles à falsifier (volontairement ou non) par rapport à tous les autres types de données.
  - **sélectionnées** : c'est un sous ensemble des données expérimentales brutes qui sont considérées comme fiables et qui seront conservées pour être utilisées dans les analyses ultérieures. Elles se révèlent également difficilement falsifiables, mais si ce premier filtrage est mal effectué, il peut fausser complètement ou partiellement les analyses [90].

— Les **artefacts** [62, 63] sont des données ayant subi une transformation, même minime, par l'être humain. Nous distinguons ceux qui sont :
  - **fabriqués manuellement** : ils ont été calculées manuellement à partir des données expérimentales. Les artefacts sont les tableaux de résultats ou les graphiques utilisés dans des publications ou des cours pour justifier des analyses.
  - **fabriqués automatiquement** : ce sont des vues de données ou des graphiques qui ont été calculés automatiquement à partir des données expérimentales et de scripts (en Python, R, etc.) afin de justifier des analyses.

— Les **données de protocole** décrivent toutes les données nécessaires pour reproduire le processus scientifique avec les mêmes logiciels et la même infrastructure. Cela inclut entre autres les paramètres des instruments de mesure, de la grille de calcul et la description des calculs permettant de reproduire les artefacts fabriqués manuellement ou bien les scripts générant les artefacts fabriqués automatiquement qui justifient les résultats obtenus. Ce sont les mêmes informations que l'on retrouve dans les cahiers de laboratoire [67].

— Les **ressources** sont des données extérieures à l'expérience (en lecture seule, comme celles du LOD par exemple). Elles sont suffisamment génériques et fiables pour être réutilisées dans un processus scientifique. Elles peuvent être des données expérimentales, des artefacts, de protocoles et même des logiciels (la version exécutable d'un logiciel étant un document comme un autre sur le Web). Les ressources permettent de réutiliser les connaissances acquises au travers d'autres recherches afin d'obtenir de nouvelles connaissances. (On réutilise et étend le terme « *resource* » de l'anglais, qui, dans un contexte RDF, correspond à tout ce qu'un graphe RDF peut décrire. [154])

De nombreux laboratoires travaillent en réseau pour diminuer les coûts liés aux données. Ainsi ils mettent en place un échange de données qui permet de partager des données expérimentales fiables entre laboratoires. Chaque chercheur peut ensuite appliquer leurs processus scientifiques avec, cependant, une seule restriction qui concerne le partage des données.

Le niveau d'accès des données. On peut distinguer trois niveaux d'accès concernant les données de la recherche :

— **Données privées** : quand un chercheur travaille seul, toutes les données restent généralement confidentielles jusqu'à la diffusion des résultats.

— **Données en accès restreints** : durant la production des artefacts et la définition des analyses, des chercheurs peuvent essayer de collaborer en limitant l'accès aux données le temps d'obtenir une analyse commune. Cet accès restreint peut même être formalisé quand il s'agit de données sensibles et/ou qu'il s'agit de faire profiter ces données en priorité aux laboratoires qui ont participé aux coûts financiers de leur collecte.

— **Données ouvertes** (ou *Open Data*) : après la publication des analyses qui s'appuient sur les artefacts, les chercheurs ouvrent généralement leurs données, si on le leur demande explicitement(par email par exemple).

Des journaux scientifiques, face à la crise de reproductibilité, commencent à encourager l'ouverture des artefacts ainsi que de toutes les données facilitant la reproductibilité des artefacts.

Face à certaines restrictions qu'imposent certains journaux scientifiques grâce à leurs monopoles, des pays commencent à légiférer pour imposer aux chercheurs d'ouvrir leurs publications et leurs données dès lors qu'ils sont financés par des fonds publics [122].

Le Linked Data dans la gestion des données de la recherche. Il y a une grande diversité de mise en œuvre possibles du Linked Data pour les données de la recherche.

— Les **données expérimentales** sont généralement des données massives. Le Linked Data ne sert ici qu'à contextualiser et enrichir ces données à l'aide de métadonnées pour faciliter leurs échanges. Ces métadonnées en RDF au sein de base de données serviront à naviguer à distance plus simplement dans ces données expérimentales stockées sur le serveur de stockage sécurisé et définitif (NAS) à proximité des grilles de calcul.

— Les **artefacts** peuvent être sauvegardés dans des documents RDF pour pouvoir être plus facilement réutilisés.

— Les **données de protocole** sont probablement les données les plus intéressantes à mettre en RDF car elles répondent aux questions nécessaires à la description des concepts décrits par les données tout au long du processus scientifique. De plus, la description et la localisation des logiciels, la taille des infrastructures ainsi que tous les paramètres et les identifiants des appareils électroniques utilisés sont autant de données essentielles qui décrivent le contexte des données expérimentales et donc leur fiabilité.

— Améliorer l'accès aux **ressources** extérieures est l'objectif principal du Linked Data en mettant à disposition un système global d'information fiable.

Concernant la gestion du niveau d'accès aux données, le Linked Data n'impacte pas la sécurité des données. On sécurise l'accès d'une bases de données RDF avec un service SPARQL de la même manière qu'on sécurise l'accès à un service Web.

Dans la section suivante, nous décrivons les freins que nous avons rencontrés en déployant le Linked Data dans les laboratoires qui voulaient avancer sur ces questions.

2.5 Les freins

Suite au déploiement du Linked Data au sein de quelques laboratoires de l'Université Paris-Saclay, nous explicitons les freins qu'ont rencontrés les chercheurs en essayant d'utiliser ces technologies.

2.5.1 Durant la découverte de données

Le premier contact du chercheur avec le LOD commence généralement par une recherche sur les données disponibles au sein de cet océan de données (*data lake*).

Comment découvrir des données au sein du LOD ?

Le LOD est par nature un système d'information global décentralisé. L'une des conséquences est que tant qu'il n'y aura pas de moteurs de recherche comme ceux de Google ou de solution alternative au sein du LOD, nous serons face aux mêmes problèmes qu'aux premières années du Web. Le premier de ces problèmes est « Comment trouver une donnée qui existe dans le LOD ? » et immédiatement suivra : « Comment découvrir la donnée dont j'ai besoin au sein de mes recherches ? ».

De plus, ces problèmes doivent prendre en compte le fait que les données des chercheurs ne doivent être visibles que s'ils le souhaitent.

Pour faciliter cette découverte, il est nécessaire de relier les données entre elles. En supposant qu'un chercheur produise des données RDF, la question est de savoir à quelles sources de données du LOD il convient de lier ses données.

A quelles données du LOD dois-je lier mes propres données ?

De nouvelles sources de données peuvent apparaître en permanence au sein du LOD. Une connaissance exhaustive du LOD afin de relier simplement les données de la recherche entre elles devient impossible.

De plus, les données d'un chercheur ne sont généralement pas accessibles sans sa permission ; il est donc le seul en mesure de décrire ses propres données et donc le seul à pouvoir les relier au LOD.

Il faut donc imaginer de nouvelles bonnes pratiques ou de nouveaux outils pour aider les chercheurs à relier simplement leurs données au LOD sans connaissance exhaustive du LOD.

Pour relier au LOD des données encore privées, il faut aussi résoudre la question suivante.

De quelle manière relier des données au LOD sans avoir besoin de les divulguer ?

Pour permettre la découverte de données au sein de l'université, il faut arriver à créer une IRI HTTP pérenne pour les données afin d'y associer les *métadonnées* nécessaires pour décrire leur contenu en RDF en les reliant au LOD. De plus, toutes ces métadonnées ne doivent pas sortir de l'université ou même du laboratoire tant que les données ne sont pas rendues publiques.

Cela implique d'imaginer un système d'information qui permettra à un chercheur de travailler sur ses données en interne sans les divulguer tout en interrogeant les données du LOD nécessaires à ses recherches.

Dès lors qu'un chercheur sait trouver des données dans le LOD, il souhaite immédiatement savoir les utiliser et recherche naturellement des exemples pour y parvenir.

Comment trouver des exemples de requêtes ?

Les requêtes SPARQL, noyées dans le code d'une application, sont difficiles à découvrir et à réutiliser, même si l'application est *open source*, c'est-à-dire dont le code est accessible. Les utilisateurs sont obligés d'analyser le code avec peu d'indications sur l'objectif des requêtes utilisées.

Il faut trouver un moyen simple pour partager des exemples de requêtes afin d'aider les utilisateurs de SPARQL à monter en compétence sur l'utilisation des données qui les intéressent.

Quand un chercheur sait interroger le LOD, la question se pose de pouvoir visualiser les données pour déterminer leur qualité.

Comment visualiser le contenu des données sans les télécharger ?

Le LOD n'étant pas téléchargeable, il faut des solutions permettant d'aider les chercheurs à visualiser et naviguer au travers des données RDF pour évaluer la qualité des données avant de les utiliser.

Ce problème et les précédents constituent autant de freins qui empêchent la découverte des données au sein du LOD, mais aussi au sein du Linked Data de l'université. Dans la section suivante, nous abordons les problèmes qui surviennent quand le chercheur commence à travailler avec les technologies du Linked Data.

2.5.2 Durant la production de données

Après une courte formation, un chercheur peut concevoir des données RDF, mais cela n'est pas suffisant pour intégrer le Linked Data dans son processus scientifique. Le premier frein est de modifier son VRE afin d'intégrer le Linked Data de l'université.

Comment travailler avec les données du Linked Data ?

Nous avons vu que les VRE sont des systèmes relativement refermés sur les communautés qui les ont conçus au grés de leurs besoins. La transversalité des échanges de données entre les VRE et le LOD est une question particulièrement difficile. Les technologies du Linked Data impliquent d'ouvrir une réflexion sur la mise en œuvre de systèmes d'information qui supporteront les communications des données de recherche entre les logiciels interdépendants (au sein des VRE).

Nous avons, au travers de cette thèse, pu travailler avec des chercheurs qui ne possédaient pas encore de VRE, et ainsi nous avons pu construire un VRE compatible avec le Linked Data.

Avec des VRE compatibles au sein du Linked Data de l'université, la question se pose ensuite naturellement sur la manière d'ouvrir progressivement ces données au sein du LOD en respectant les contraintes du chercheur.

Comment ouvrir progressivement ses données ?

Il y a une telle confusion entre les concepts de données ouvertes et de Linked Data que souvent les chercheurs pensent que leurs données seront immédiatement en ligne dès lors qu'on les rend compatibles avec le LOD. Quand les données sont liées au LOD, il faut encore les rendre accessibles au travers d'Internet et cela peut passer par les niveaux d'accès que le chercheur peut définir.

De plus, aujourd'hui, le gain de l'ouverture des données est relativement faible pour un chercheur. Cela implique que pour encourager cette mise en ligne, il faut que l'effort pour partager ses données soit quasiment nul, c'est-à-dire sans avoir besoin d'effectuer un post-traitement pour les mettre à disposition. Cela suppose que les données soient compatibles avec le LOD dès leur production au sein du VRE.

Pour cela, il faut que le chercheur sache décrire ses données en RDF.

Comment concevoir son propre schéma RDF pour décrire ses propres données ?

Il est simple de construire un schéma RDF avec l'un des nombreux logiciels que nous avons fait tester à des chercheurs [139], comme (Web)Protégé [145] ou encore TopBraid Composer [31]. Malheureusement, ces outils nécessitent de passer par un contrôleur de code source dès que plusieurs chercheurs sont impliqués dans la conception de la même ontologie, ce qui est trop complexe pour la majorité des chercheurs.

Cependant, des solutions existent pour travailler à plusieurs sur la même ontologie, comme Wikibase (le logiciel de Wikidata) [168], l'extension Semantic MediaWiki [87] ou Web Protégé [145]. Malheureusement, ces solutions n'intègrent pas les contrôles d'accès exigés par les chercheurs.

De plus, ces solutions s'appuient sur des schémas RDF (codés au cœur des logiciels) et imposant aux chercheurs des contraintes dans la conception de leurs propres schémas RDF qui peuvent leur sembler trop contraignantes.

La complexité de travailler à plusieurs, le manque de contrôle sur le niveau d'accès aux données, les contraintes de conception d'un schéma RDF sont autant de raisons techniques qui font que généralement, les chercheurs abandonnent souvent l'écriture de leur schéma RDF après seulement quelques jours.

Sans solution mieux adaptée à leurs besoins, les chercheurs ne peuvent maîtriser le processus de description et d'interrogation de leurs propres données, ce qui est clairement un point bloquant pour l'adoption du Linked Data comme solution technique d'échange et de réutilisation de leurs données.

Dans la section suivante, nous nous concentrerons sur les problèmes de la réutilisation des données des chercheurs, car même si un chercheur arrive à décrire ses données et à les collecter au sein de son VRE, encore faut-il qu'il soit en mesure de les réutiliser lui-même.

2.5.3 Durant la réutilisation des données

Les chercheurs qui ont commencé à produire leurs données RDF sont immédiatement confrontés à la difficulté d'utiliser SPARQL.

Comment simplifier la conception d'une requête ?

Rédiger une requête SPARQL peut se révéler fastidieux [18], y compris pour des utilisateurs expérimentés, et ce, pour plusieurs raisons, incluant la maîtrise imparfaite par l'utilisateur des schémas RDF impliqués pour décrire les connaissances, la nécessité de suivre une syntaxe souvent ressentie comme lourde, et la difficulté à réutiliser des requêtes écrites par des tiers.

Les réponses à ces problèmes se concentrent trop souvent sur la conception d'interfaces qui masquent le langage d'interrogation, par exemple au travers de mots-clés, de questions en langage naturel, de pseudo-requêtes, de requêtes schématiques ou construites à partir d'exemples de résultats attendus [71]. L'inconvénient de ces méthodes est double pour SPARQL. D'abord, l'utilisateur ne progresse pas dans l'apprentissage du langage et, par conséquent, il n'est pas en mesure d'améliorer ses

compétences au rythme des progrès des applications qui l'intéressent, ou de l'augmentation des données disponibles au sein du LOD. Ensuite, l'utilisateur est limité dans sa recherche non seulement par le schéma RDF supportée par l'interface, mais aussi et surtout par les fonctions SPARQL supportées par l'interface.

SPARQL étant une technologie récente, les outils pour faciliter la conception de requêtes par les utilisateurs eux-mêmes restent à imaginer et à inventer. Pour l'heure, c'est un point bloquant rédhibitoire pour la plupart des chercheurs qui souhaitent savoir interroger eux-mêmes leurs données pour pouvoir les analyser.

Une fois qu'un utilisateur sait interroger le LOD avec SPARQL, il est confronté à la nature du LOD, qui est décentralisé dans sa distribution et sa description. En d'autres termes, les schémas RDF peuvent changer et ces changements impacteront inévitablement les méthodes de réutilisation des données.

Comment utiliser des données évoluant en permanence ?

L'utilisation de requêtes SPARQL offre de nombreux avantages mais également des inconvénients, comme le fait que les applications deviennent plus difficiles à maintenir, car les schémas RDF peuvent évoluer indépendamment des applications qui les utilisent.

Il existe des solutions comme SPARQL2Git [95], qui gère les versions des requêtes au travers de GitHub pour faciliter la maintenance des applications qui sont impactées par la modification de ces requêtes quand leurs schémas RDF évoluent. Malheureusement, l'utilisation de technologies comme Git auprès d'utilisateurs non-informaticiens est difficile à envisager et nécessite d'autres solutions.

Dans la section suivante qui conclut ce chapitre, nous évoquons l'approche que nous appliquerons au travers de cette thèse afin de travailler sur les freins et les verrous que nous avons identifiés.

2.6 Conclusion

Dans ce chapitre, nous avons décrit le Linked Data au travers de son histoire. Puis, nous avons décrit l'objectif que doit permettre d'obtenir le Linked Data au sein de l'université et les bonnes pratiques nécessaires du W3C pour y parvenir. En appliquant ces bonnes pratiques dans plusieurs laboratoires de différents domaines, nous avons identifié des freins et des points bloquants qui empêchent l'adoption du Linked Data.

Nous avons ainsi présenté dans ce chapitre deux contributions.

La première contribution est la description de ce que devrait être le « *Linked Data d'une université* » au travers de la sélection précise de onze bonnes pratiques du W3C, dans l'objectif de construire un système d'information décentralisé et fiable au service de la recherche compatible avec le Linked Data.

Notre deuxième contribution est la liste des premières questions que se posent les chercheurs qui désirent utiliser ces technologies et qui devront trouver une réponse pour qu'ils utilisent réellement le Linked Data de l'université.

A travers la tentative de mettre en place le Linked Data dans une université, nous percevons que les problèmes sont trop interdépendants pour être résolus séparément. Par exemple, si un chercheur ne sait pas réutiliser ses propres données en RDF, il n'utilisera pas de données RDF dans ses activités et, s'il n'en utilise pas régulièrement, il ne saura pas interroger le LOD quand il en aura besoin.

C'est pour cette raison que nous avons construit l'« approche LinkedWiki », qui prend simultanément trois directions pour essayer de lever les verrous de :

— l'intégration du Linked Data au sein du VRE des chercheurs,

— la découverte des données RDF,

— et la réutilisation des données RDF par les chercheurs eux-mêmes.

Ainsi, dans le chapitre suivant, nous décrivons les solutions qui nous semblent lever la majorité des verrous que nous avons rencontrés en intégrant le Linked Data au cœur du processus scientifique. Les chapitres 4 et 5 aborderont les problèmes les plus complexes que nous avons rencontrés en essayant de lever ces verrous.

Résumé du chapitre 2

Dans ce chapitre, nous avons décrit ce que devrait être selon nous le Linked Data à l'université, c'est-à-dire un système d'information décentralisé et fiable au service de la recherche et compatible avec le Linked Data.
Au travers de la sélection de onze bonnes pratiques du W3C, nous avons décrit les objectifs techniques à atteindre pour concevoir le Linked Data dans une université.
En appliquant ces bonnes pratiques dans plusieurs laboratoires de différents domaines, nous avons identifié les premiers verrous qui empêchent l'adoption du Linked Data. De ces expériences, nous avons défini l'« approche Linked-Wiki » afin de résoudre ces verrous.
Le chapitre 3 décrit cette approche et nos solutions. Les chapitres 4 et 5 abordent les verrous les plus complexes.

Chapitre 3

LinkedWiki : gestion des données avec le Linked Data

Sommaire

3.1 Introduction

La promesse du Web sémantique pour les chercheurs est d'accélérer leurs recherches en exploitant la diversité et la richesse des données disponibles et en réutilisant les connaissances existantes mises à disposition dans le LOD (Linked Open Data).

Dans le chapitre précédent, nous avons motivé le choix d'utiliser les technologies du Linked Data pour la mise en œuvre d'un système d'information en mesure de supporter les applications du Web sémantique dont les chercheurs seront les premiers bénéficiaires.

Le présent chapitre est consacré à la présentation d'une solution de partage des données capable de faciliter la réutilisation des données tout en respectant les contraintes qu'implique la mise en œuvre du Linked Data. Le glissement progressif des données des chercheurs dans le Linked Data de l'université puis dans le LOD sera ainsi simplifié.

Notre solution s'applique à toutes les données de la recherche (structurées en RDF ou non), petites ou massives, fichiers ou bases de données. Ele répond à trois besoins :

— améliorer la découverte des données de la recherche en les reliant au LOD ;

— permettre le déploiement d'environnements virtuels de recherche (en anglais, VRE) qui alimenteront le Linked Data interne de l'université et permettront une meilleure gestion des données ;

— faciliter la réutilisation des données (la reproduction des analyses) du Linked Data de l'université et du LOD par les chercheurs, qui produiront à leur tour de nouvelles données.

Nous avons proposé notre solution à plusieurs équipes de recherche de l'Université Paris-Saclay. Une de ces équipes a accepté de l'expérimenter afin de construire son propre VRE. Nous avons ainsi été amenée à créer :

— la plateforme LinkedWiki [114] pour faciliter la création d'un environnement de travail compatible avec le Linked Data propice à la production, au référencement et la gestion des données scientifiques, ainsi qu'à la reproduction des analyses (des artefacts) ;

— l'extension LinkedWiki [116] pour MediaWiki afin d'aider les chercheurs à construire leurs propres schémas RDF dans leurs propres bases de connaissances connectées au LOD ;

— des outils facilitant la découverte des données produites sur des services tiers comme un plugin pour Wikipédia [109] ou encore des clients SPARQL pour PHP [79] et JavaScript/TypeScript [78].

Nous décrivons dans ce chapitre notre approche en commençant, en section 3.2, par les solutions que nous avons adoptées face aux problèmes abordés dans le chapitre 2. Ensuite en 3.3, nous détaillons l'implémentation des contributions qui résultent de cette solution. En section 3.4, nous évaluons quantitativement nos contributions à l'aide des données d'usage disponibles, et qualitativement avec les cher-

cheurs qui participent à l'expérimentation de notre solution pour construire leur propre VRE. Enfin, nous évoquons l'avenir de ces travaux en 3.5.

3.2 Solutions adoptées

Dans cette section, nous reprenons les problèmes recensés dans le chapitre 2 en leur apportant des solutions.

3.2.1 Solutions pour la découverte de données

Dans cette section, nous nous focalisons sur les solutions adoptées relatives à la découverte des données de la recherche.

3.2.1.1 Découvrir en naviguant dans Wikipédia

Avec notre solution, la découverte des données de la recherche peut s'effectuer en naviguant sur Wikipédia.

Observation des chercheurs. Quand les chercheurs recherchent des données, ils utilisent souvent un moteur de recherche tel que Google, qui propose généralement des pages Wikipédia en tête des résultats. Ce résultat est dû au fait que Google classe les pages par popularité et que les pages Wikipédia répondent souvent aux besoins de la majorité des internautes. Ainsi, Wikipédia est aussi en tête des résultats pour des sujets scientifiques précis [88]. On observe donc qu'un chercheur trouve très simplement la page Wikipédia qui concerne le sujet en relation avec les jeux de données qui l'intéressent. On en conclut que la mise en œuvre d'une solution permettant d'utiliser Wikipédia comme point d'entrée pour trouver les données de la recherche faciliterait la découverte des données de la recherche au travers du Web.

Observation sur l'organisation de Wikipédia. Wikipédia [167] est une grande encyclopédie collaborative multilingue qui contient des informations générales, spécialisées, des almanachs, des répertoires géographiques, etc. Cependant, s'agissant des connaissances scientifiques, Wikipédia ne permet pas d'accéder aux jeux de données qui ont permis d'obtenir ces connaissances.

Pour se retrouver dans les millions de pages (5 millions en anglais et 2 millions en français en juillet 2018), les contributeurs de Wikipédia ajoutent des catégories générales « pour regrouper des pages sur des sujets similaires » [166].

Lier les catégories de Wikipédia aux données de la recherche : une fausse bonne idée. Les chercheurs utilisent généralement des taxonomies pour organiser leurs résultats et leurs données. Par exemple, l'Association for Computing Machinery (ACM) publie sa propre taxonomie [129] qui sert à annoter des articles dans le domaine de la recherche en informatique.

Le fait de vouloir rapprocher les taxonomies des catégories de Wikipédia pour naviguer dans les données au travers de Wikipédia est donc naturel. Il est d'ailleurs possible de transformer les catégories de Wikipédia en une taxonomie [105, 171].

Figure 3.1 – Après authentification et installation de notre *plugin* dans Wikipédia (lien Web pour l'installer), si des données sont disponibles et liées au sujet de la page Wikipédia, un onglet « Research » (Recherche) permet l'affichage de ces données. Exemple avec la page « Herschel (télescope spatial) » qui permet de découvrir le portail officiel des données de ce satellite et voir la fiche synthétique de ces données (lien Web vers l'article Wikipédia).

Néanmoins, les taxonomies sont difficiles à fusionner avec ces catégories « parce que les concepts entre taxonomies sont de granularités différentes, de structures différentes, ambiguës et partiellement incompatibles » [5].

Relier les concepts de Wikipédia directement aux données. Wikidata, un autre projet de la fondation Wikimédia (que nous décrivons en détail un peu plus loin), associe un ID unique à chacun des millions de concepts décrits au travers de Wikipédia. En utilisant Wikidata comme pivot entre Wikipédia et les données de la recherche, il devient possible de relier les données aux concepts sans avoir besoin de rapprocher des taxonomies partiellement incompatibles.

Utiliser Wikipédia comme moteur de recherche en reliant les données à Wikidata. Nous avons implémenté (1) un système décrivant les données des chercheurs en utilisant les ID de Wikidata et (2) un *plugin* dans Wikipédia pour permettre de s'en servir comme moteur de recherche afin de découvrir les jeux de données des chercheurs.

Bien que Wikipédia soit déjà utilisé pour découvrir des documents [169] ou des experts [35], les solutions actuelles de partage des données de la recherche [42, 146, 54, 6, 49] ne l'utilisent pas pour relier les données aux articles Wikipédia décrivant leurs sujets de recherche. Notre contribution diffère donc, à notre connaissance, des travaux précédents.

Impact sur l'expérimentation. Notre solution prend la forme d'un onglet supplémentaire sur les pages de Wikipédia. Un chercheur, quelle que soit sa langue, en passant par Google puis en naviguant sur Wikipédia, peut maintenant découvrir

des données de l'Université Paris-Saclay en cliquant sur cet onglet supplémentaire, comme l'illustre la figure 3.1. Cet onglet n'est visible que par les chercheurs authentifiés dans Wikipédia et après avoir installé manuellement au sein de Wikipédia le *plugin* [110] que nous avons développé.

Le chercheur peut pour chaque jeu de données : lire une description sommaire, accéder aux données via un lien Web et accéder à une fiche synthétique regroupant les différents formats disponibles et des exemples pour les réutiliser, comme l'illustre la figure 3.2.

3.2.1.2 Lier les données au LOD avec Wikidata

Dans la section précédente, nous avons utilisé Wikidata pour indexer librement les données de la recherche aux concepts décrits dans Wikipédia. Dans cette section, nous montrons que cette indexation va permettre de résoudre *de facto* le problème consistant à relier les données de la recherche au LOD sans avoir besoin de divulguer ces données.

Observation sur le projet Wikidata. La fondation Wikimédia, qui chapeaute Wikipédia, mène des recherches dans le domaine du Linked Data afin de simplifier le travail de ses contributeurs, par exemple pour leur éviter d'écrire des faits redondants entre les articles de langues différentes [99]. Les résultats de ces recherches sont intégrés au sein du projet Wikidata.

Wikidata comprend, en 2018, 49 millions d'objets [162] (avec 851 classes [127] et 4 909 propriétés [128]). Quiconque (humain ou machine) peut modifier les données, proposer de nouvelles propriétés et ajouter des références pour chaque fait au sein de cette base de connaissances. Si un concept scientifique est absent ou incorrect, un chercheur peut facilement l'ajouter ou le rectifier.

De plus, ces concepts sont accessibles et mis à jour en temps réel au travers d'un service SPARQL qui supporte des millions de requêtes journalières [163].

Wikidata est devenu un excellent point d'appui pour que d'autres services se connectent au LOD.

Relier les données de la recherche à Wikidata pour les connecter au LOD. Des projets importants tels que Freebase [24] ou DBpedia [91] intègrent déjà leurs connaissances à Wikidata ou l'utilisent comme pivot pour se connecter au LOD. De plus, Wikidata collabore fortement avec les chercheurs (exemple de proposition de projet européen en 2015 entre l'Université Paris-Saclay et Wikidata : *Wiki4R* [96]).

En liant les jeux de données des chercheurs aux identifiants des concepts dans Wikidata, la plateforme les relie aux LOD et en même temps, à l'aide de notre *plugin* vu dans la section précédente, Wikipédia devient un moteur de recherche pour les données des chercheurs.

Relier les données au LOD au travers d'une passerelle. Notre système crée et sauvegarde un *graphe RDF* décrivant les liens entre les données et Wikidata. Ces liens sont accessibles au travers d'un service SPARQL et, ainsi, ce service constitue une passerelle entre le LOD, dont Wikidata fait partie, et les données de la recherche.

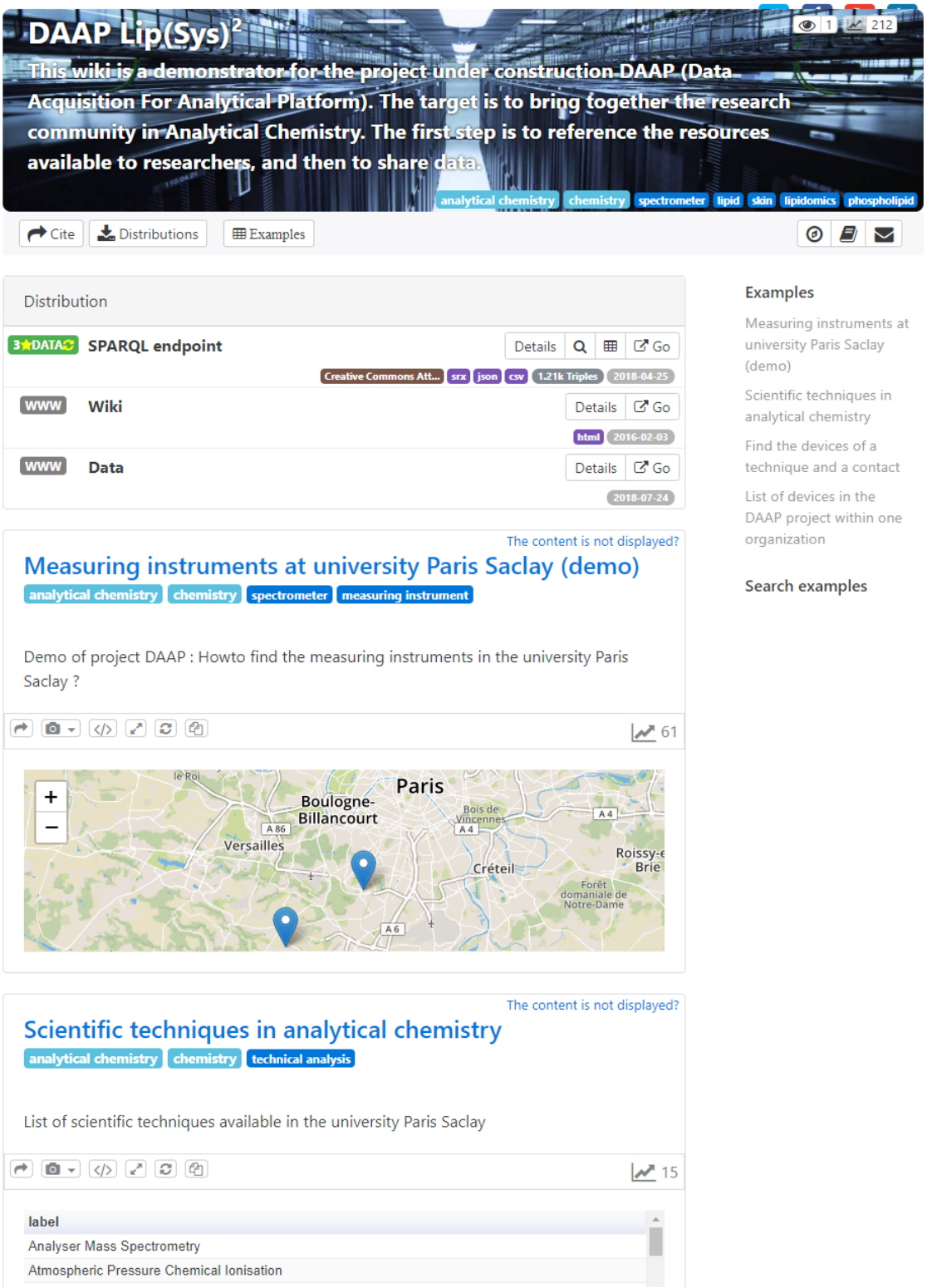


Figure 3.2 – Sur la fiche synthétique d'un jeu de données, on retrouve les modes de distribution de ces données et des exemples de leurs réutilisations si les données sont aussi distribuées au travers d'un service SPARQL (lien Web).

Cette passerelle permet de :

1. surmonter la difficulté du manque de pérennité des URL des jeux de données en y associant une URL pérenne ;
2. avertir le propriétaire quand ses données ne sont plus accessibles ;
3. définir un niveau d'accès aux données pour permettre à leur propriétaire de les ouvrir progressivement.

Dans la section suivante, nous expliquons la méthode que nous adoptons pour créer les liens entre les données des chercheurs et les identifiants de Wikidata.

3.2.1.3 Lier manuellement les données

La solution la plus simple à implémenter pour lier les données au LOD est de laisser les utilisateurs créer eux-mêmes les liens entre leurs données et le LOD. Cependant, cette liberté ne convient pas à tous les utilisateurs. Dans cette section, nous allons décrire les trois méthodes que nous avons expérimentées pour lier les données au LOD : par les humains, par les machines [56] ou bien par un processus spécifique mettant en jeu humains et machines. A partir de ces expériences, la méthode qui consiste à laisser les chercheurs indexer librement leurs données au travers d'un processus manuel a été retenue.

Créer les liens manuellement. En 2016, avec le laboratoire Lip(Sys)^2 de la Faculté de Pharmacie et l'IPS2 (Institut des Sciences des Plantes), nous avons expérimenté le moyen de retrouver les appareils de mesure des laboratoires de l'Université Paris-Saclay au travers de Wikipédia avec un *plugin* [110]. Néanmoins, l'effort nécessaire pour calculer automatiquement les concepts décrivant le montage des appareils sur les paillasses d'un laboratoire est largement supérieur à celui de demander directement aux chercheurs quelques mots-clés. De plus, le recensement de ces quelques mots-clés uniquement évite aux chercheurs de divulguer leurs cahiers de laboratoires qui décrivent leurs montages et restent des documents très sensibles.

Nous avons également expérimenté avec le LRI (Laboratoire de Recherche en Informatique) la recherche au travers de Wikipédia des informations relatives aux thèses encadrées dans le laboratoire [112] et aux chercheurs [113]. La mise en relation entre Wikipédia et les données du LRI a été faite manuellement au travers d'un formulaire. Cela a permis de relier simplement et définitivement les identifiants de Wikidata à l'ontologie/taxonomie qu'avaient commencé à construire certains chercheurs pour décrire leurs thèmes de recherche. Même s'il est possible d'imaginer un algorithme pour effectuer/faciliter cette opération, il n'aurait pas été possible de l'appliquer, car les données nécessaires se trouvent dans des bases de données confidentielles du LRI.

On peut conclure de ces expériences que lorsque les informations nécessaires pour relier les données au LOD ne sont pas déjà ouvertes, numérisées et organisées, il est préférable d'appliquer une méthode simple en offrant des outils aux utilisateurs pour effectuer cette tâche eux-mêmes s'ils le souhaitent.

Pour confirmer cette conclusion, dans le paragraphe suivant, nous décrivons notre expérience concernant la génération automatique de liens.

Créer des liens automatiquement Nous avons fait une autre expérience en développant un autre *plugin* qui permet de découvrir les équipes de l'Inria (Institut national de recherche en informatique et en automatique) au travers de Wikipédia [56]. Automatiquement, nous avons relié 3 123 pages de rapports Inria de 2014 à 129 499 articles de Wikipédia. Ce résultat a été ensuite transféré dans un service SPARQL afin de les rendre accessibles par notre *plugin* [111].

Les résultats sont relativement positifs car on peut généralement retrouver les équipes Inria au travers de leurs thématiques dans Wikipédia. Cependant, il existe toujours une marge d'erreur qui va aboutir à associer des équipes Inria à des thématiques qui ne sont pas les leurs. Corriger ce problème nécessitera un post-traitement avant de déployer en production ce traitement. Ce type de post-traitement, mettant en jeu des experts humains, est lourd à implémenter et ainsi, deux ans après, ce traitement n'a pu être mis en production.

Cette expérience nous enseigne que ce traitement automatique n'a pu être implémenté que parce que les données étaient ouvertes, numérisées et organisées. De plus, pour être considéré comme intéressant dans la durée, ce type de traitement doit intégrer dès sa conception un processus permettant à des humains ou à d'autres machines d'améliorer dans la durée les liens créés, c'est-à-dire permettant la correction des liens dans la durée (lien mort, lien incorrect, etc.).

Suggérer des liens. Après avoir expérimenté la création de liens manuels et automatiques, nous avons testé le nouveau service *Mix'n'match* [115] de Wikidata qui permet la suggestion de liens entre les concepts de Wikidata et d'autres bases de données.

Pour tester ce service, nous avons y avons injecté environ 2 300 termes ACM [115] pour essayer de les lier automatiquement à Wikidata. Puis, nous avons développé un autre *plugin* pour réutiliser ces liens afin de permettre la recherche au travers de Wikipédia dans des archives d'articles scientifiques en informatique référencés depuis plusieurs décennies par ACM [108].

Durant cette expérience, environ 300 termes ont été reliés automatiquement, mais les contributeurs de Wikidata doivent confirmer ou supprimer chacune des relations suggérées. Pour les 2 000 autres termes non reliés par le service, les contributeurs peuvent maintenant les associer manuellement à un identifiant de Wikidata.

Le service *Mix'n'match* démontre que Wikidata cherche à simplifier au maximum la création de liens entre ses concepts avec les autres bases de connaissances. Malgré cela, ce travail reste long et fastidieux, mais il est définitif.

Les traitements automatiques restent possibles pour lier des données dans Wikidata mais si une erreur est détectée par un humain, tous ces liens peuvent être annulés. Ainsi, il est préférable d'appliquer les résultats de ces traitements avec une marge d'erreur (même minime) au travers du service *Mix'n'match*, qui permet la validation définitive des liens par les contributeurs.

Wikidata est maintenant l'une des bases de connaissances les plus importantes et son contrôle systématique par des humains de tous les traitements automatiques fait partie des éléments qui ont conduit à ce succès. Wikidata est probablement un exemple à suivre dans la constitution de bases de connaissances durables au sein du Linked Data.

Impact dans notre expérimentation. La majorité des sources décrivant des données de la recherche accumulent souvent les inconvénients suivants : elles sont non ouvertes, non numérisées ou encore non organisées. Par exemple, les publications de recherche sont une source d'information potentielle sur les données, mais elles ne contiennent que rarement les informations permettant à une machine de les récupérer.

Un traitement suggérant automatiquement les liens vers le LOD n'est envisageable qu'à la condition de disposer des données associées à leurs descriptions. Pour le moment, seuls les chercheurs peuvent fournir (manuellement) ces informations.

C'est la raison pour laquelle nous avons implémenté une solution pour insérer manuellement toutes les métadonnées décrivant un jeu de données. Quand le système disposera de suffisamment d'informations pour suggérer des liens automatiquement, un chercheur, avec notre solution, pourra corriger simplement et définitivement ces suggestions (comme le fait Wikidata avec ce type de traitement automatique).

En attendant de pouvoir calculer des suggestions automatiques, le chercheur peut lier très simplement ses données au LOD au travers d'une recherche par mots-clés parmi les millions de concepts contenus dans Wikidata.

Avec ces solutions, notre expérimentation a ainsi pu constituer un catalogue opérationnel des données de la recherche au sein de l'Université Paris-Saclay. Dans la section suivante, nous traiterons de l'opportunité de créer également un catalogue de requêtes SPARQL vers les données RDF des chercheurs.

3.2.1.4 Catalogue de requêtes

Pour résoudre le problème de la découverte du contenu des données et de leurs requêtes (vu en section 2.5.1), nous proposons un catalogue de requêtes pour naviguer au travers des données.

Partager les vues interactives générées par les requêtes. Sans le Linked Data, un scientifique recherchant des données utilisables n'a d'autre choix que de télécharger une partie des données pour en évaluer la qualité.

Avec des données RDF au sein d'un service SPARQL, notre système permet de naviguer dans les données au travers d'illustrations, comme le montre la figure 3.2, et facilite la découverte de ces données.

Un chercheur en partageant ses requêtes SPARQL permet à d'autres chercheurs de dupliquer puis de modifier ses requêtes pour partager de nouvelles requêtes à son tour.

Un catalogue de requêtes et de données qui s'enrichissent mutuellement. Notre système constitue un catalogue de requêtes pour faciliter la découverte des données. D'autres catalogues de requêtes ont été développés. Par exemple, LSQ (*Linked SPARQL Queries dataset*) [126] décrit des requêtes extraites des logs de services SPARQL publics, et SPARQL2Git [95] offre un catalogue et permet la co-conception de requêtes au travers de services comme GitHub.

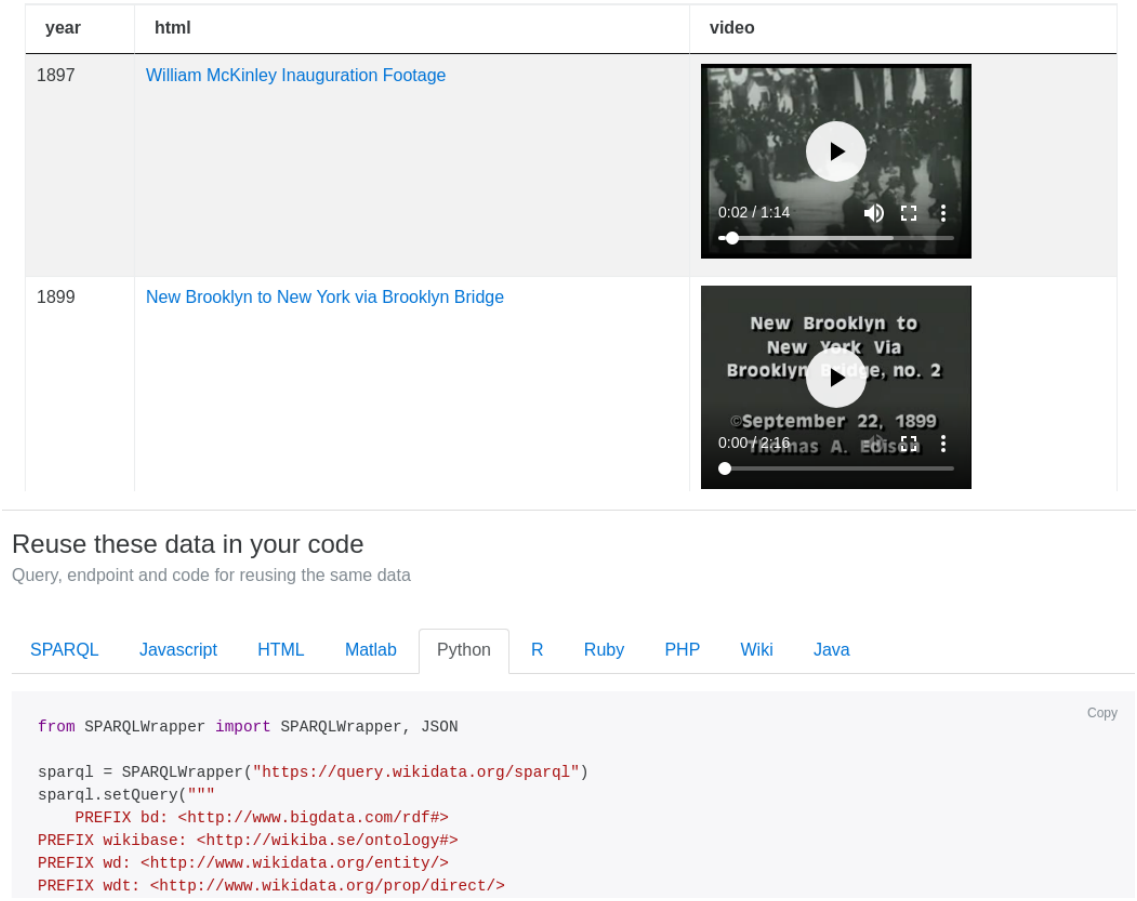


Figure 3.3 – Sur le site de la plateforme LinkedWiki, un utilisateur peut naviguer dans les données au travers d'illustrations interactives. Pour réutiliser une requête, il suffit de cliquer sur l'onglet qui correspond à son langage de programmation et copier le code qui apparaît ; ici un extrait du code en Python. (lien Web).

Cependant, aucun de ces catalogues n'intègre un catalogue de requêtes à un catalogue de données permettant de s'enrichir mutuellement de la manière suivante.

1. L'écriture de requêtes augmente le nombre de vues interactives sur les données.
2. Les vues interactives permettent de naviguer dans les données.
3. La navigation dans les données facilite leurs découvertes et leur réutilisation, comme le décrit la figure 3.3.
4. La réutilisation des données déjà partagées encourage les utilisateurs à partager d'autres données ainsi que des requêtes pour décrire leurs schémas RDF.
5. De nouvelles données permettent d'écrire de nouvelles requêtes (retour à 1).

Ci-dessous, nous décrivons nos solutions afin de créer un environnement de travail compatible avec le Linked Data pour la recherche.

3.2.2 Solutions pour l'analyse des données

Dans cette section, nous exposons nos choix d'implémentation pour simplifier le travail des chercheurs afin de pouvoir utiliser les données du LOD et y partager leurs propres données.

3.2.2.1 Travailler au sein d'un VRE compatible avec le LOD

Nous traitons, ici, le problème des environnements virtuels de recherche (VRE) pour les rendre capables de travailler avec le LOD.

Proposition d'un environnement de travail ouvert. En 2015, nous avons travaillé à une proposition de projet de recherche à laquelle Wikidata ainsi que de nombreuses universités ont participé [96]. Le projet visait à créer des environnements de travail (VRE) complètement ouverts avec comme épine dorsale une ontologie partagée pluridisciplinaire comme Wikidata. L'objectif était de créer entre autres un VRE compatible avec le Linked Data pour permettre la consommation et la production de données du LOD.

Malheureusement, le projet, qui suggérait de bouleverser radicalement les pratiques professionnelles des chercheurs, n'a pas été financé.

Un VRE compatible avec le Linked Data. En travaillant avec des chercheurs qui comprennent les avantages de la science ouverte mais qui vivent la difficulté de maintenir leurs recherches dans un contexte concurrentiel international fort, nous avons pu concevoir une approche qui diffère des précédentes, permettant simultanément :

1. d'ouvrir les données de la recherche uniquement si le chercheur le souhaite ;
2. d'offrir les outils nécessaires pour consommer les données du LOD ;
3. et d'intégrer les technologies du Linked Data dès la conception des protocoles de recherche jusqu'à la publication des résultats.

Nous avons ainsi créé un prototype de VRE au sein de notre expérimentation pour permettre la consommation et le transfert sans effort des données du LOD, mais sous le contrôle du chercheur, qui prendra lui-même la décision d'ouvrir ou non ses données.

Gestion des données au travers d'un VRE modulable. Un environnement de recherche se compose généralement de plusieurs composants. Les besoins entre chercheurs sont différents, mais on peut généraliser un certain nombre des composants de VRE pour mutualiser l'effort de leur conception pour ensuite les mettre à disposition de tous les chercheurs.

Pour constituer le VRE de notre expérimentation, nous avons créé des composants génériques de VRE au sein de machines virtuelles que notre plateforme LinkedWiki met à disposition des chercheurs. LinkedWiki est donc une plateforme PAAS (plateforme comme un service) et elle repose sur l'IAAS (infrastructure comme un service) de l'Université Paris-Saclay.

Un chercheur peut ainsi solliciter un composant générique de VRE en un clic ou, s'il n'existe pas encore, demander à des ingénieurs systèmes de l'université de le

construire pour le mettre à disposition. Ainsi, un processus peut se mettre en place pour contrôler les composants de VRE avant leurs mises à disposition au sein de l'Université afin de s'assurer qu'elles puissent produire et consommer les données compatibles avec le Linked Data.

Ainsi pour contrôler nos composants de VRE qui contiennent des bases de données RDF et leurs services SPARQL, nous avons implémenté un indicateur d'interopérabilité qui informe les chercheurs sur la compatibilité réelle au Linked Data du composant de VRE qu'ils souhaitent utiliser. La figure 3.4 montre l'interface de sélection des composants de VRE. Nous reviendrons sur les motivations et la mise en oeuvre de cet indicateur dans le chapitre 5.

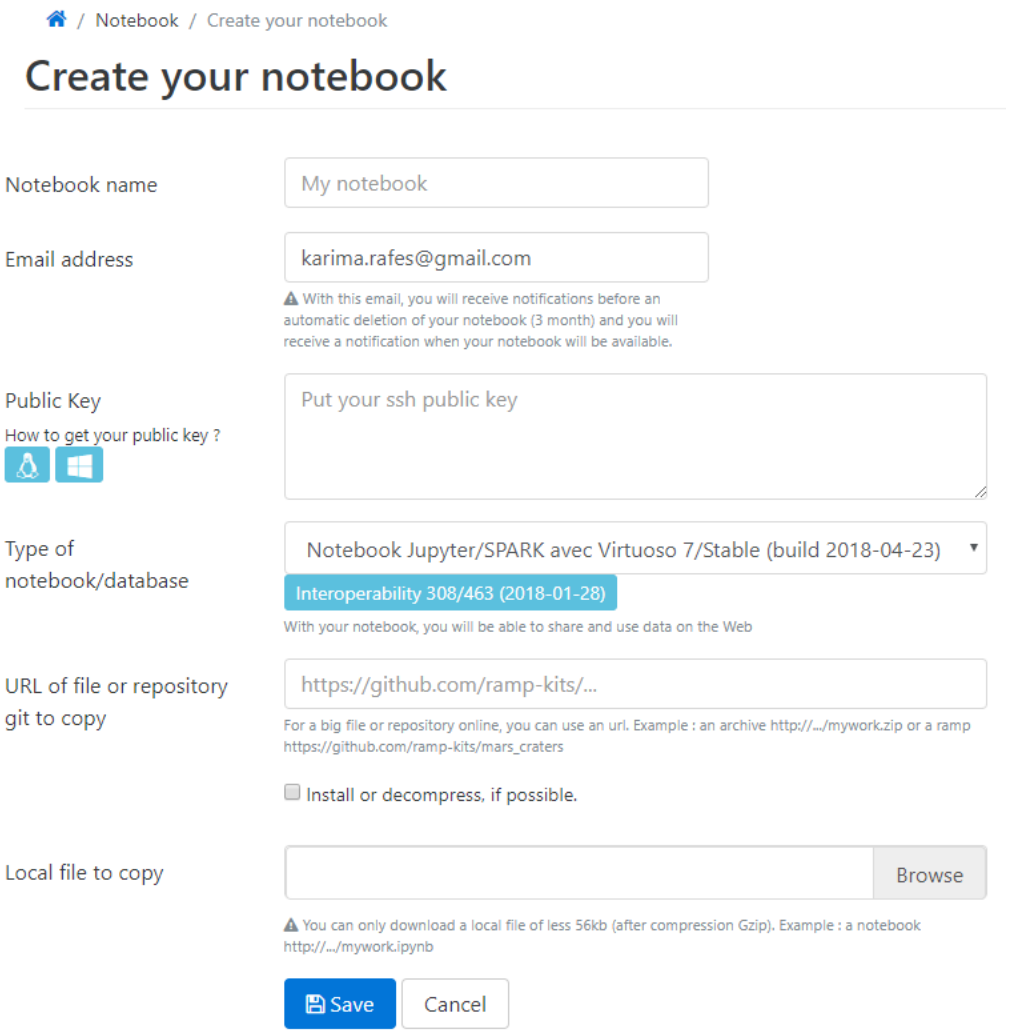


Figure 3.4 – La figure 3.4 montre l'interface de sélection des composants de VRE au travers du site Web de la plateforme LinkedWiki. Le composant de VRE, sélectionné ici, contient un notebook Jupyter, un client SPARK (pour se connecter à la grille de calcul du VRE) et une base de données RDF Virtuoso avec un service SPARQL dont l'interopérabilité est évaluée à 308 sur 463 par rapport aux standards fixés au sein du Linked Data de l'université (plus de détails en chapitre 5).

Dans la section suivante, nous décrivons de quelle manière un chercheur va pouvoir ouvrir progressivement ses données.

3.2.2.2 Permettre l'ouverture progressive des données

Bien que l'ouverture des données soit possible au sein d'un VRE, il faut encore définir la manière de les ouvrir en fonction des collaborations et des résultats publiés. Dans cette section, nous décrivons les niveaux d'accès que nous avons implémentés pour permettre aux chercheurs de contrôler l'ouverture de leurs données.

Distinguer plusieurs niveaux d'accès pour progressivement ouvrir les données. Généralement, la confidentialité des données est sous la responsabilité directe des scientifiques. Leurs données sont généralement privées avant la publication de leurs résultats, mais ils sont souvent disposés à les partager avant avec leurs collègues afin de faciliter la collaboration.

Ainsi, la plateforme distingue plusieurs niveaux d'accès : privé, restreint (par exemple, accessibilité aux chercheurs de l'université et à leurs partenaires) ou ouvert (à tous).

Dans la section suivante, nous abordons le problème de production des données structurées en RDF, c'est-à-dire du schéma RDF, directement par le chercheur afin qu'il puisse décrire ses connaissances lui-même.

3.2.2.3 Concevoir librement son schéma RDF en équipe

Dans cette section, nous décrivons la solution au problème, vu en section 2.5.2, qui permet aux chercheurs de construire leurs premières bases de connaissances à l'aide d'un simple wiki.

Partager les connaissances au travers d'un wiki. Lorsqu'un scientifique introduit des informations dans un cahier de laboratoire, nous aimerions connecter ses informations aux bases de connaissances disponibles. Cela conduirait à des cahiers de laboratoires plus modernes, plus appropriés pour décrire les flux de données, et en particulier les données de protocole ainsi que les résultats obtenus. Cela encouragerait la collaboration entre les scientifiques [70].

Dans notre expérimentation, le cahier de laboratoire prend la forme d'un wiki, MediaWiki, l'un des wikis, open source, les plus populaires où nous avons implémenté une extension pour y intégrer plusieurs outils.

Si un schéma RDF a été conçu au travers d'un logiciel extérieur, il peut être recopié dans le wiki (au format RDF/Turtle) pour permettre à plusieurs chercheurs simultanément de l'étendre (en passant par des logiciels extérieurs si nécessaire).

Détecter les incohérences avec le LOD pour les corriger. L'extension que nous avons implémentée permet de créer des modules (en Lua), comme une infobox, afin de visualiser et sauvegarder simultanément des données dans une base de données RDF au travers de son service SPARQL qui autorise les requêtes en écriture.

Figure 3.5 – L'extension LinkedWiki pour MediaWiki permet de transférer les données vers une base de données RDF et de visualiser les incohérences entre bases de connaissances. Ici, une infobox affiche en rouge les divergences avec les données de Wikidata (lien Web).

L'extension peut comparer les données contenues dans le wiki à d'autres données dans le LOD, comme avec Wikidata via SPARQL comme l'illustre la figure 3.5.

Les utilisateurs détectent plus facilement, au travers de ce wiki, les connaissances manquantes et les incohérences.

Au travers de ce wiki, avec quelques notions en Lua et Turtle (RDF), le chercheur peut commencer à construire sa base de connaissances afin de décrire les données qu'il souhaite collecter dans son processus expérimental.

Concevoir un schéma RDF sans contrainte extérieur tout en s'assurant de son intégrité. Pour maintenir ce schéma RDF « bien formé », notre wiki vérifie les contributions RDF en Turtle 1.1 avant de les enregistrer.

Pour maintenir le schéma RDF « valide » pour que des machines au sein du VRE puissent le réutiliser dans la durée, notre wiki utilise des règles d'intégrité pour générer un rapport de ces règles non respectées dans une page spéciale du wiki. Ces règles d'intégrité sont écrites avec le vocabulaire SHACL [84] en Turtle au sein des pages du wiki.

De plus, le rapport des règles d'intégrité porte sur la base de connaissances que constitue le wiki, mais aussi sur toutes les données RDF privées ou publiques accessibles par le service SPARQL au sein du VRE (ceci inclut les métadonnées extraites en RDF des données expérimentales et qui ne figurent pas dans les pages du wiki).

Contrôler les niveaux d'accès tout en ne bridant pas les utilisateurs dans leur capacité à améliorer les données. On distingue dans le wiki le contenu en langage naturel et le contenu en RDF (en Turtle). Il est possible de faire cohabiter les deux contenus sur la même page du wiki.

Cependant, des chercheurs ont exprimé le besoin de limiter la capacité des utilisateurs à modifier leurs schémas RDF partagés, car ces données sont réutilisées au sein de leurs VRE par leurs machines et une erreur peut entraîner de lourdes conséquences.

Ainsi, nous avons créé un espace sécurisé au sein du wiki pour rendre invisible la gestion des schémas RDF pour les utilisateurs du wiki n'ayant pas la nécessité de les modifier. Cet espace permet à chaque page principale du wiki (écrite en langage

naturel) de disposer d'un onglet « Data » où l'utilisateur autorisé peut rajouter (automatiquement ou manuellement) des données RDF en Turtle en relation avec cette page principale du wiki.

Même sans les droits de modifications les utilisateurs peuvent continuer à suggérer des améliorations en modifiant les pages principales du wiki. Ces différences entre les pages écrites en langage naturel et les données RDF sont regroupées dans une page de catégorie du wiki pour être traitées par les chercheurs, qui veilleront à la mise à jour de leurs bases de connaissances à partir des améliorations suggérées.

Dans la section suivante, nous traitons des problèmes que rencontrent les chercheurs quand ils souhaitent utiliser leurs données RDF, mais également les autres données du Linked Data.

3.2.3 Solutions pour la réutilisation des données

Après avoir implémenté la plateforme et commencé à former les chercheurs afin qu'ils produisent leurs propres données structurées, le besoin de **savoir réutiliser** leurs données RDF s'est révélé être un point bloquant à l'adoption de notre solution. Ainsi, nous avons travaillé sur les problèmes de la conception d'une requête et sur celui de la stabilité des applications utilisant des requêtes qui peuvent reposer sur des schémas RDF mouvants comme celui de Wikidata.

3.2.3.1 Abonnement aux requêtes et aux schémas RDF

Dans cette section, nous allons décrire comment notre solution répond aux problèmes que posent les requêtes SPARQL au sein d'applications qui deviennent plus difficiles à maintenir, car les schémas RDF dans le Linked Data peuvent évoluer indépendamment des applications qui les utilisent (vu en section 2.5.3).

S'abonner aux schémas RDF en s'abonnant aux requêtes. Les illustrations au sein de notre catalogue de données sont construites à partir des résultats de requêtes SPARQL recalculées régulièrement. Un chercheur changeant son schéma RDF devra donc rafraîchir ses requêtes pour que ses illustrations puissent continuer à s'afficher correctement dans ce catalogue.

En s'appuyant sur ce comportement, nous avons implémenté la possibilité de s'abonner aux requêtes pour être alerté de leurs modifications. Ainsi, les abonnés à ces requêtes sont immédiatement avertis si leurs schémas RDF ont changé. Les utilisateurs qui réutilisaient la requête modifiée de notre catalogue dans leur application peuvent ainsi corriger immédiatement leur application avec une requête adaptée à la nouvelle version du schéma RDF.

A notre connaissance, cette contribution qui permet de s'abonner aux requêtes SPARQL est la première qui aborde de cette manière le problème de la stabilité des applications qui consomment des schémas RDF pouvant évoluer.

Dans la section suivante, nous nous décrirons la solution adoptée pour faciliter la réutilisation des requêtes.

3.2.3.2 Tester et réutiliser simplement les requêtes

Afin d'aider les utilisateurs occasionnels de SPARQL (problème vu en section 2.5.3), nous nous sommes concentrée sur la manière de tester avant usage les requêtes SPARQL et la manière de les réutiliser sans effort.

Tester avant usage les requêtes. Le catalogue de requêtes que nous avons implémenté relance régulièrement les requêtes pour s'assurer du bon fonctionnement des services SPARQL.

De plus, l'utilisateur peut dupliquer une requête de notre catalogue et l'ouvrir directement dans notre éditeur SPARQL pour la modifier et la tester avant de la réutiliser.

Générer des exemples de code. Après avoir choisi une requête à l'aide des illustrations associées au catalogue de requêtes, un utilisateur (occasionnel) peut simplement dupliquer une requête existante pour la modifier en fonction de ses besoins. Une fois celle-ci modifiée, il pourra réutiliser la requête immédiatement dans son code, comme le montre la figure 3.3, en utilisant un exemple de programme généré par notre système intégrant cette requête.

Il nous faut maintenant aborder le problème de la conception des requêtes SPARQL.

3.2.3.3 Conception d'outils d'autocomplétion de requêtes

Rédiger une requête SPARQL pouvant se révéler difficile (*cf.* en 2.5.3), nous décrivons dans cette partie nos outils d'*autocomplétion* qui servent à guider un utilisateur dans la rédaction de sa requête par la proposition de portions de code SPARQL.

Quatre types d'autocomplétions sont essentielles. Pendant un an, nous avons réalisé une expérimentation [118] sur l'utilisation de SPARQL auprès de deux types d'utilisateurs : 103 étudiants issus de formations variées (en informatique, en administration des entreprises, du niveau L3 au M2), dans le cadre de travaux pratiques de SPARQL, et 60 professionnels au travers de formations internes et événements proposés au sein du *Center for Data Science* de l'Université Paris-Saclay.

Nous avons observé que les erreurs courantes des utilisateurs SPARQL sont par ordre d'importance : les erreurs de syntaxe, les préfixes inconnus et la connaissance imparfaite des schémas RDF impliqués dans la requête. Cette dernière catégorie touche tous les types d'utilisateurs, des novices jusqu'aux experts.

A partir de cette expérience, une taxonomie et un rapide état de l'art des différents types d'autocomplétion ont été réalisés [118] (détails en annexe A). Nous avons choisi d'en implémenter quatre nous-même, comme l'illustre la figure 3.6 :

— **Autocomplétion des préfixes.** Certains services SPARQL autorisent l'exécution de requêtes avec des préfixes implicites, ce qui est une source d'erreur lorsque l'on réutilise ces requêtes au travers d'autres services. Nous avons implémenté la correction automatique des préfixes manquants en réutilisant la connaissance des préfixes au sein de notre catalogue de requêtes.

— **Autocomplétion des IRI via mots-clés.** Dans le cas où un utilisateur ne connaît pas le schéma RDF qu'il interroge, nous avons implémenté une

autocomplétion des IRI via mots-clés dans la langue maternelle de l'utilisateur, ce qui ne fonctionne actuellement qu'avec Wikidata.

— **Autocomplétion par modèle.** Nous avons implémenté ce type d'autocomplétion mais nous n'y avons inséré que deux modèles que nous pensons indispensables pour notre expérimentation. Le cas 3, dans la figure 3.6, permet d'accéder aux labels des entités Wikidata, et le cas 4 évite de réécrire dans les requêtes le filtre `langMatches` afin de filtrer les labels dans une langue précise (ce qui est souvent utile quand on interroge d'autres services que celui de Wikidata).

— **Suggestion de snippets.** Cette autocomplétion consiste à suggérer des morceaux de requêtes (des snippets) à l'utilisateur quand il conçoit sa requête en se fondant sur d'autres requêtes similaires. Cela se révèle très utile pour adapter rapidement les exemples de requêtes sans avoir besoin de maîtriser tous les schémas RDF des services SPARQL. La figure 3.6 illustre cette suggestion de snippets. Le chapitre 4 décrit précisément les algorithmes sous-jacents à cette technique d'autocomplétion.

Etat de l'art pour ces quatre types d'autocomplétions. Nous avons étudié les éditeurs existants SPARQL dans [123] et plusieurs autres afin de vérifier si ces autocomplétions existent déjà et s'il était envisageable de les réutiliser au sein de notre système.

Nous avons recensé huit éditeurs qui incluent au moins une des autocomplétions attendues par nos utilisateurs, et qui peuvent être testés au travers d'un navigateur Web. Les résultats sont résumés dans le tableau 3.1.

Les deux éditeurs les plus connus sont Flint SPARQL Editor [144] et YASGUI [123]. Flint propose des modèles de requête. YASGUI a agrégé plusieurs fonctionnalités avancées telles que l'*autocomplétion des déclarations des préfixes* en utilisant l'API du service Prefix.cc [36]. Gosparqled [27], qui se base sur YASGUI, fournit aussi des autocomplétions d'IRI par mots-clés.

Une autre catégorie d'éditeurs provient des fournisseurs de bases de données RDF proposant des éditeurs connectés à un seul service SPARQL. Ces éditeurs sont souvent basiques. Une exception est iSPARQL [101] où l'autocomplétion par modèle est proposée (parmi beaucoup d'autres fonctionnalités).

D'autres solutions sont basées sur des domaines spécifiques tels que BioCarian [170], fournissant des modèles adaptés aux requêtes sur la biologie moléculaire.

L'éditeur de Wikidata [164] propose non seulement des modèles, mais aussi des *autocomplétions en utilisant des IRI relatifs* via des mots-clés sur la base de données Wikidata. Fait intéressant, Gosparqled [27] et Wikidata supportent la recherche par mots-clés d'une manière complètement différente. Le premier le fait via des requêtes SPARQL en utilisant la fonction REGEX. L'avantage de cette méthode est qu'elle est applicable à tous les services SPARQL. L'inconvénient est que le temps de réponse d'une requête SPARQL avec REGEX devient inacceptable quand la base de données est trop importante. De son coté, Wikidata a implémenté une API spécifique pour permettre à son éditeur de rechercher ses IRI via des mots-clés.

En ce qui concerne la fonctionnalité d'*autocomplétion par snippets*, nous pouvons

Table 3.1 – Les types d'autocomplétions dans les éditeurs SPARQL

Fonctionnalités	Flint Editor[a]	iSPARQL[b]	LODatio+[c]	BioCarian[d]	Gosparqled[e]	Wikidata Query[g]	YASGUI[h]	**LinkedWiki editor**[i]
IRI relatif par mots-clés	-	-	-	-	+	+	-	+
Déclaration des préfixes	-	-	-	-	+	-	+	+
Modèle	+	+	+	+	-	+	-	+
Snippets	-	-	+[j]	-	-	-	-	+

a http://bnb.data.bl.uk/flint-sparql Flint SPARQL Editor v1.0.4 [144]

b https://www.openlinksw.com/isparql OpenLink iSPARQL v2.9 [101]

c http://lodatio.informatik.uni-kiel.de LODatio+ (testé en mai 2018) [55]

d http://www.biocarian.com BioCarian SPARQL editor (testé en mai 2018) [170]

e http://scampi.github.io/gosparqled/ Gosparqled (testé en mai 2018) [27]

g https://query.wikidata.org Wikidata Query (testé en mai 2018) [164]

h http://doc.yasgui.org/ YASGUI v2.7.27 [123]

i https://io.datascience-paris-saclay.fr/exampleInsertUpdate.php **LinkedWiki platform v2.0.0**

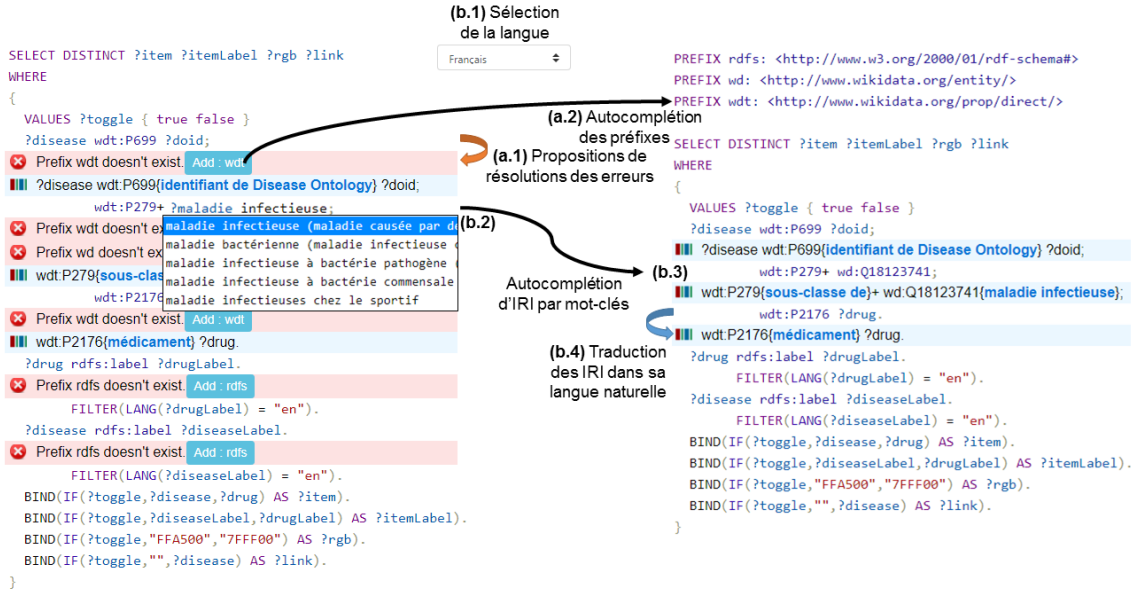
j Uniquement des snippets d'un seul patron de triplets.

mentionner LODatio+ [55]. Ce n'est pas un éditeur mais un moteur de recherche. Il est capable de trouver des sources de données pertinentes pour une requête sur la base de ses patrons de triplets qui la relient à une combinaison spécifique de types et/ou de propriétés RDF. LODatio+ indexe différents schémas dans une base de connaissances et propose de compléter une requête au travers de suggestions qui ajoutent ou suppriment un seul patron de triplets à la fois. Cette fonctionnalité est celle que nous avons trouvée la plus proche de notre définition d'*autocomplétion par snippets*. Cependant, elle est très limitée dans la mesure où elle ne suggère qu'un seul patron de triplets à la fois.

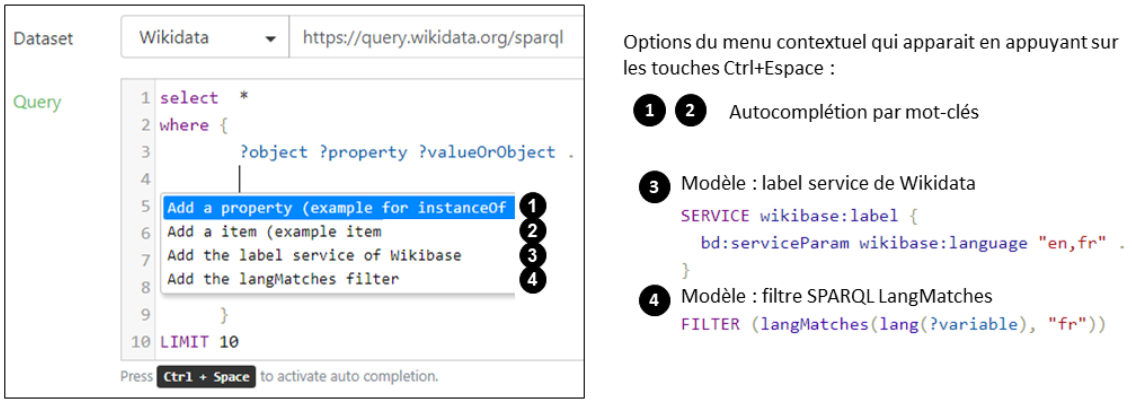
En résumé, ces éditeurs supportent certaines fonctionnalités d'autocomplétions intéressantes, mais de façon très limitée. Aucun d'entre eux ne supporte les quatre types d'autocomplétion les plus attendues par nos utilisateurs et l'autocomplétion par snippets est presque inexistante bien qu'elle soit la fonctionnalité la plus demandée.

L'autocomplétion par snippets est la fonction la plus attendue de nos utilisateurs et la plus originale vis-à-vis de l'état de l'art. Elle sera décrite en chapitre 4.

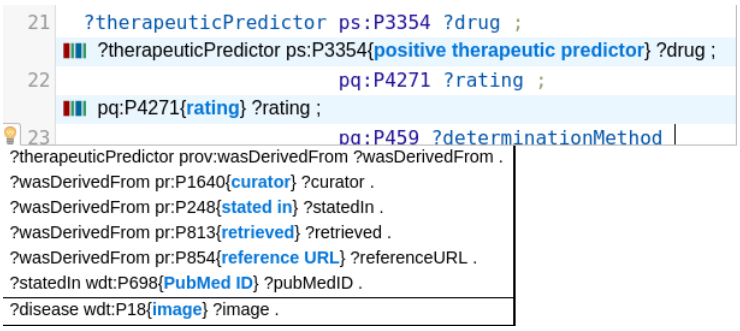
Dans la section suivante, nous décrivons notre implémentation des solutions adoptées que nous venons de présenter.



(a) Autocomplétion des préfixes inconnus, (b.1-3) autocomplétion par mots-clés dans la langue de l'utilisateur, ici pour Wikidata et (b.4) insertion de lignes explicitant les IRI en langage naturel (lien Web).



En pressant Ctrl+Espace, l'utilisateur choisit entre une autocomplétion par mots-clés pour rechercher (1) une propriété ou (2) un type ou une instance, ou bien pour rechercher un modèle pour (3) appeler le service *label* de Wikidata, ou pour (4) écrire le code SPARQL pour choisir un tag *lang* (lien Web).



Une icône d'« ampoule » apparaît dans la marge de gauche afin de proposer des snippets à l'utilisateur pour l'aider à terminer l'écriture de sa requête (détails en chapitre 4).

Figure 3.6 – Les autocomplétions dans l'éditeur SPARQL de LinkedWiki (lien Web).

3.3 Implémentation

Les principaux composants logiciels que nous avons développés sont les suivants :

— Une plateforme avec une interface Web pour publier des données et les requêtes des chercheurs en utilisant le Linked Data et pour déployer leurs VRE [114], ainsi qu'un éditeur SPARQL qui intègre des outils d'autocomplétions.

— Une extension pour MediaWiki au sein de composants génériques de VRE pour permettre aux chercheurs de construire leur propre base de connaissances privée ou publique avec leurs propres schémas RDF [116].

— Des outils, open source, de réutilisation des données comme un *plugin* pour découvrir les données de scientifiques en utilisant Wikipédia comme un moteur de recherche [109] et des librairies permettant d'afficher des données et des graphiques à partir d'une requête SPARQL sans avoir besoin de transférer ces données à des services extérieurs [78, 79].

Ces trois points font l'objet des trois prochaines sous-sections.

3.3.1 Plateforme LinkedWiki

Dans cette section, nous décrivons notre plateforme qui doit être en mesure de gérer les VRE compatibles avec le Linked Data et qui servira de plateforme de publication des jeux de données (produits au travers des VRE ou non) au sein de l'université.

Une instance de cette plateforme est déployée à l'Université Paris-Saclay (à cette adresse : `https://io.datascience-paris-saclay.fr`). Elle est la seule qui permette également le déploiement des composants des VRE des chercheurs, car cela nécessite une IAAS que seule une organisation de la taille d'une université, par exemple, peut maintenir et mettre à la disposition de milliers de personnes.

Notre plateforme offre un service SPARQL pour interroger librement les méta-données publiques des jeux de données référencés en son sein (dont voici le point d'accès : `https://io.datascience-paris-saclay.fr/sparql`).

Une autre instance de la plateforme (version 1.0) est déployée à destination de personnes intéressées qui ne font pas partie de l'université. (Son adresse est la suivante `http://linkedwiki.com`.)

Nous discutons ici de l'implémentation de la plateforme LinkedWiki (version 2.0 déployée) à l'Université Paris-Saclay.

Gestion des VRE compatibles avec le Linked Data. Pour permettre la création de VRE, la plateforme utilise l'IAAS de l'Université Paris-Saclay. L'université utilise les logiciels OpenStack pour fournir ce service. Ainsi, la plateforme se connecte aux API d'OpenStack pour déployer et gérer des machines virtuelles, c'est-à-dire les composants de VRE des chercheurs.

Nous avons donc implémenté une plateforme PAAS (plateforme comme un service) pour mettre à disposition les composants génériques de VRE.

Tout laboratoire ou chercheur peut maintenant construire ses propres VRE (compatible avec le Linked Data ou non) et les mettre à disposition de toute l'université en passant par cette plateforme.

Pour le moment, nous limitons le nombre de machines virtuelles à une par utilisateur, mais rien n'empêche de supprimer cette limitation pour gérer plusieurs composants de VRE simultanément afin de gérer un VRE complet au travers de la plateforme. En attendant de lever cette limitation, si un chercheur a besoin de plus de ressources, il doit :

1. demander à son laboratoire les ressources dont il a besoin pour son VRE pour créer son propre projet OpenStack qui hébergera tous les composants de son VRE, et
2. se servir de la plateforme pour créer et transférer les composants de VRE dont il a besoin dans son propre projet OpenStack.

Pour notre évaluation qualitative où nous avons commencé à expérimenter un VRE complet, nous avons déployé quatre images de machines virtuelles avec différentes bases de données RDF (et donc différents moteurs d'inférence). Pour qu'une image soit visible dans la liste des composants de VRE disponibles dans l'interface de la plateforme, il faut attacher à l'image quelques métadonnées et déplacer l'image dans un projet spécifique d'OpenStack. Si cette image offre un service SPARQL, on pourra dans les métadonnées préciser le logiciel et sa version pour afficher son indicateur d'interopérabilité à destinations des futurs utilisateurs (détails en chapitre 5). L'objectif est de familiariser les chercheurs à ces notions d'interopérabilité pour qu'ils bénéficient des technologies du Linked Data.

La figure 3.7 illustre la manière dont la plateforme instancie le VRE d'un chercheur. Puis, à l'issue de la publication de ses recherches, le chercheur peut transférer son VRE vers une infrastructure permanente afin de mettre à disposition définitivement ses travaux. Ensuite, pour faciliter la réutilisation des données disponibles au sein des VRE, la plateforme sert à référencer, interroger et illustrer les données des chercheurs.

Déclaration des jeux de données. Avec la plateforme LinkedWiki, les scientifiques déclarent leurs jeux de données comme des données ouvertes ou pour une utilisation restreinte au sein de l'université.

Les métadonnées associées publiées sont facilement réutilisées grâce à un service SPARQL.

Pour partager ces métadonnées, nous avons utilisé l'ontologie DCAT ([93], bonne pratique n°11) qui a été créée à cet effet au sein du W3C. Cependant, nous avons dû étendre cette ontologie avec deux nouvelles propriétés :

— la propriété « keywordConcept » : sa valeur est l'IRI d'une entité de Wikidata. Elle joue le même rôle que la propriété « keyword » dans DCAT, à ceci près qu'elle lie réellement le jeu de données au LOD au travers de Wikidata. De plus, l'entité décrite dans Wikipédia offre une description bien plus précise qu'un simple mot-clé. Nous avons cependant conservé la propriété d'origine « keyword » pour assurer l'interopérabilité avec l'ontologie DCAT d'origine.

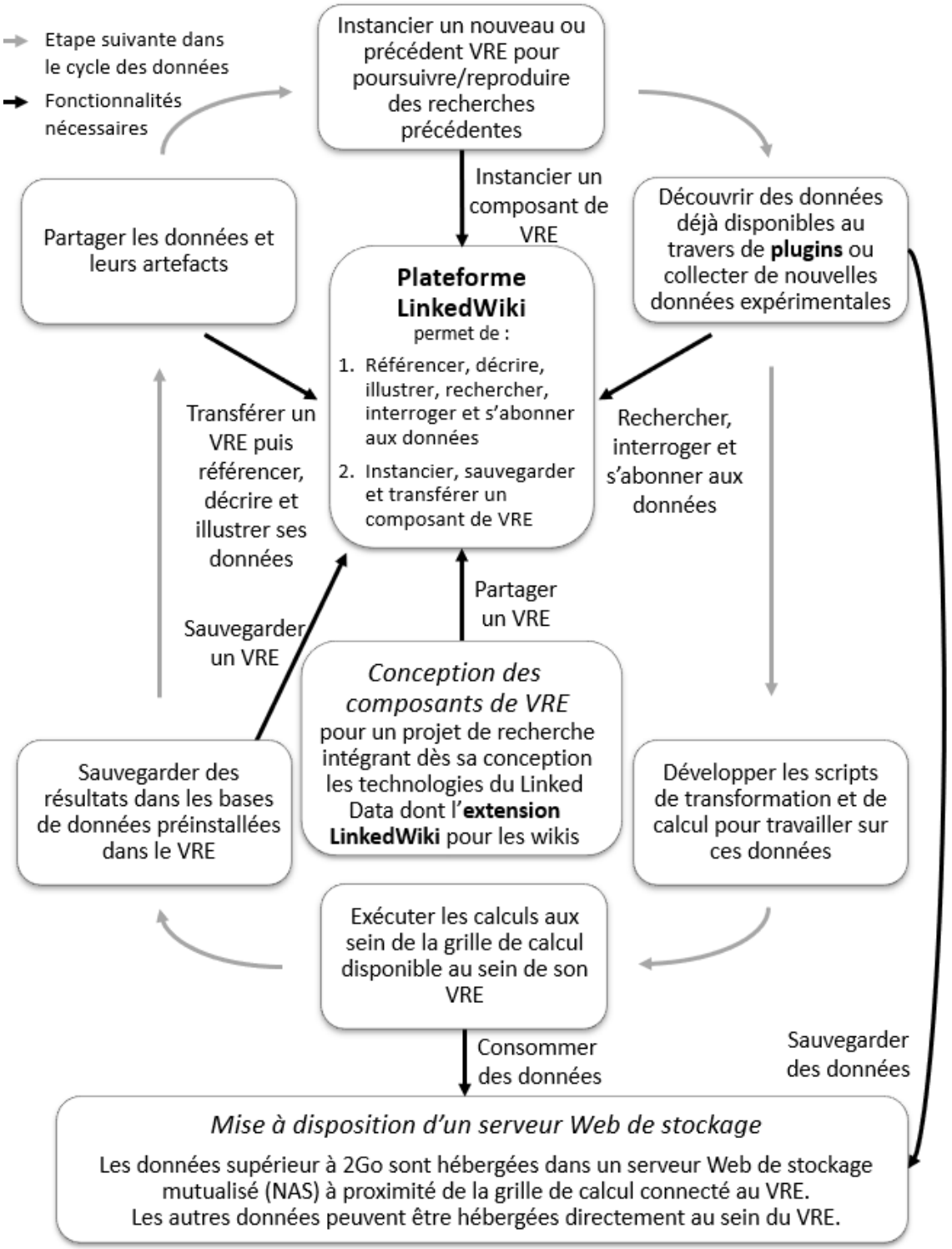


Figure 3.7 – L'approche LinkedWiki met en place un cercle vertueux où un chercheur peut instancier un environnement virtuel de recherche (VRE).

— la propriété « theme » : sa valeur est encore une IRI d'une entité de Wikidata qui correspond au contexte d'utilisation. Par exemple, le jeu de données du *télescope spatial Herschel* sera lié au thème *cosmologie* qui est le domaine de recherche dans lequel ces données seront particulièrement utiles.

En résumé, les « *keywordConcepts* » décrivent le « quoi » et les « *themes* » décrivent le « pourquoi » de ces jeux de données. Ces métadonnées permettent en même temps d'alimenter une recherche à facettes pour les utilisateurs et de relier les jeux de données au LOD pour les machines.

Déclaration des distributions d'un jeu de données. Dans la recommandation DCAT, nous avons également implémenté le concept de *distribution*.

> Une *distribution* représente une forme spécifique disponible d'un jeu de données. Chaque jeu de données peut être disponible sous différentes formes. Ces formes représentent différents formats de données ou différents points d'accès. Les formes de distribution incluent le format de fichiers CSV, une API comme un flux RSS ou bien un service SPARQL, etc.

Le chercheur peut décrire quatre types de distributions, via :

— une page Web donnant accès au jeu de données, par exemple, au travers d'un portail de données comme dans le cas du télescope spatial Herschel (on n'utilisera que la propriété `dcat:landingPage`) ou bien un tutoriel ;

— un fichier qui contient la distribution des données dans un format quelconque ;

— un *torrent*, c'est-à-dire un fichier sur un serveur Web, qui sert à configurer un client P2P (pair à pair) afin de télécharger plus rapidement le jeu de données avec l'aide des autres utilisateurs de ces données ;

— un point d'accès au travers d'une API comme un service SPARQL ou tout autre protocole.

Les métadonnées nécessaires pour construire l'interface de ces distributions vont bien au-delà de l'ontologie DCAT. Nous n'étendons l'ontologie d'origine et ne rendons accessibles ces métadonnées que si elles sont effectivement utilisées à l'extérieur de la plateforme. Ainsi, pour l'heure, seules les distributions de service SPARQL sont partagées par la plateforme à l'aide de l'ontologie DCAT associée à l'ontologie SPARQL Service Description [142].

Conception, partage et réutilisation des requêtes. L'éditeur SPARQL et la librairie Sgvizler2 (client SPARQL) que nous avons spécialement implémentés pour la plateforme LinkedWiki permet de concevoir et exécuter une requête directement dans un navigateur Web. La librairie d'origine Sgvizler [136] (2012) a été réécrite intégralement pour supporter et reporter à l'utilisateur les erreurs provenant des services SPARQL en cas d'erreurs de protocole ou dans la requête en cours de conception. Il est à noter que le protocole SPARQL 1.1 (2013) ne traite pas les messages d'erreurs. La gestion des erreurs est donc chaotique, voire impossible quand il s'agit d'une requête fédérée. Sgvizler2 offre un début de solution pour résoudre ce problème, en attendant une version plus riche du protocole.

Une fois écrite, une requête peut être affichée à l'aide d'une vue graphique (détails en section 3.3.3) au travers de l'éditeur SPARQL et attachée au jeu de données afin de l'illustrer. Chaque requête, comme chaque jeu de données, est associée aux métadonnées permettant de faire une recherche par mots-clés et à facettes sur les requêtes.

De plus, les requêtes collectées permettent de constituer une base de connaissances que les outils d'autocomplétion de notre éditeur SPARQL peuvent utiliser pour aider les utilisateurs à concevoir leurs requêtes SPARQL. Dans le chapitre 4, nous reviendrons plus longuement sur une contribution qui offre la première approche d'autocomplétion (à notre connaissance) capable de suggérer des snippets (fragments de la requête SPARQL) en fonction des requêtes précédentes écrites par des utilisateurs, enrichissant ainsi l'expérience de l'utilisateur.

Simplifier la conception des liens vers le LOD. Les scientifiques rencontrent généralement des difficultés pour décrire leurs données. A l'aide de notre approche, ils relient leurs jeux de données, distributions et requêtes aux IRI de Wikidata sans effort. Pour sélectionner des sujets scientifiques qui existent bien dans Wikidata, la plateforme LinkedWiki utilise l'API de Wikidata, permettant aux scientifiques de sélectionner des IRI par l'intermédiaire d'une recherche par mots-clés. Cependant, un scientifique qui veut sélectionner un concept précis qui n'existe pas encore dans Wikidata pourra le créer immédiatement en (i) créant une nouvelle page dans Wikipédia et (ii) déclarant ce nouveau concept dans Wikidata. Ce nouveau concept deviendra alors visible dans l'interface pour y être sélectionné et attaché comme métadonnée aux données du scientifique.

Coté mise en œuvre. Les métadonnées des requêtes et des données dans la plateforme sont stockées dans une base de données SQL. Une vue en RDF des données réellement réutilisées et un service SPARQL sont pris en charge. À l'heure actuelle, seules certaines métadonnées des jeux de données avec le niveau d'accès « ouvert à tous » sont exportées en RDF dans le service SPARQL. Pour l'heure, le besoin d'accéder depuis un autre service aux métadonnées en accès limité n'a pas été jugé nécessaire. Cependant, si un jour ce besoin devait apparaître, il suffirait de configurer la base de données Virtuoso [150], qui permettra de fournir un service SPARQL sécurisé avec un contrôle d'accès et une gestion des niveaux d'accès appropriés.

Parmi les composants de VRE que la plateforme peut déployer, le composant qui permet la constitution d'une base de connaissances est un des composants les plus complexes à mettre en place. Dans la section suivante, nous décrivons la manière la plus simple pour un chercheur de créer sa première base de connaissances.

3.3.2 Transformer un wiki en base de connaissances RDF

Dans cette section, nous décrivons l'extension qui permet aux chercheurs de construire leurs premières bases de connaissances.

Notre extension LinkedWiki s'installe sur les wikis de type MediaWiki [116] (qui a débuté en 2008) et qui est en licence libre dans la forge de la fondation Wikimédia.

Une nouvelle version a été totalement repensée en 2016, dont voici les principales fonctionnalités.

Visualiser les données. L'extension permet aux chercheurs d'interroger leurs données contenues dans des services SPARQL privés ou publics et d'afficher le résultat des requêtes au sein des pages du wiki. Pour améliorer les performances et diminuer la sollicitation des bases de données RDF, le résultat de la requête peut être stocké dans le cache du wiki et rafraîchi régulièrement.

Concevoir un schéma RDF. La conception de modules (codés en Lua) au sein du wiki permet avec notre extension d'écrire directement dans un des graphes (publics ou privés) au sein d'une base de données RDF via SPARQL. Il est aussi possible d'écrire du RDF en Turtle 1.1 directement dans le wiki. En installant avec ce wiki le logiciel RAPPER [34], le wiki peut rejeter automatiquement les contributions RDF qui sont syntaxiquement incorrectes.

Pour vérifier l'intégrité au sein de toutes les données RDF du VRE, il est possible d'écrire en RDF/Turtle des contraintes SHACL [84]. Notre extension avec le logiciel RDFUnit [4] crée une page spéciale dans le wiki afin de générer le rapport des règles d'intégrité non respectées dans la base de données RDF qu'alimente le wiki ainsi que les outils de collecte de données structurées au sein du VRE.

Contrôler mais sans bloquer l'accès. L'extension contrôle les niveaux d'accès tout en ne bridant pas les utilisateurs dans leur capacité à critiquer le schéma RDF pour permettre de l'améliorer. Ainsi, l'extension crée des droits supplémentaires au sein du wiki pour rendre invisible la gestion du schéma RDF pour les utilisateurs du wiki n'ayant pas la nécessité de le modifier. Même sans les droits de modification, les utilisateurs peuvent continuer à suggérer des améliorations en modifiant les pages du wiki. Ces différences entre les pages écrites en langage naturel et les schémas RDF sont regroupées dans une page « catégorie » du wiki pour être traitées par les chercheurs, qui veilleront à la mise à jour de leurs bases de connaissances à partir des améliorations suggérées.

Dans les sections suivantes, nous décrivons les logiciels que nous avons implémentés pour notre plateforme et notre extension.

3.3.3 Réutiliser les données du Linked Data

Dans cette section, nous décrivons les logiciels open source et réutilisables que nous avons implémentés et qui servent à interroger les services SPARQL du LOD depuis Wikipédia, un site Web ou directement depuis un navigateur Web.

Réutiliser des données depuis Wikipédia. Nous avons implémenté en Javascript un simple *plug-in* (« gadget » dans la culture Wikipédia) pour accéder aux jeux de données directement depuis Wikipédia [109]. Ce *plugin* est illustré dans la figure 3.1. Une fois installé, ce *plugin* s'exécute sur tous les projets de la fondation Wikimédia (Wikipédia dans toutes les langues, Wiktionnaire, etc.). Pour le réaliser, nous avons utilisé les technologies standard de Wikipédia [165]. Le code du *plugin* fonctionne selon les étapes suivantes : (i) trouver l'identifiant Wikidata du concept traité par la page en cours, (ii) générer une requête SPARQL, (iii) afficher les jeux de

données dans les résultats (sous forme de liens cliquables) quand l'utilisateur clique sur l'onglet « Research ».

Réutiliser des données depuis un site Web. D'après [159], 83% des sites Web sont développés en PHP. Nous avons également développé la plateforme LinkedWiki en PHP pour bénéficier de cet écosystème. De plus, nous avons participé à cet écosystème en implémentant et partageant : notre librairie client PHP SPARQL 1.1 [79] de la plateforme et notre extension LinkedWiki pour MediaWiki (aussi en PHP).

Réutiliser des données depuis son navigateur pour afficher des graphiques, des tableaux ou des cartes. Dans le passé, pour aider les utilisateurs à partager leurs requêtes SPARQL sur la plateforme, le wiki et les réseaux sociaux, nous intégrions la librairie Sgvizler [136] (codée avec Javascript), qui utilise les graphiques générés au travers des services de Google, c'est-à-dire en transférant préalablement les données à Google. Dans un VRE, il n'est pas possible d'utiliser les graphiques de Google car les données sont confidentielles tant qu'elles ne sont pas partagées.

Nous avons donc réécrit intégralement cette librairie en TypeScript (successeur probable de Javascript). Sgvizler2 [78] intègre les graphiques de Google pour la rétro-compatibilité, mais nous avons commencé à l'étendre avec des librairies qui n'imposent pas de transférer les données vers l'extérieur du VRE, comme les librairies D3.js (graphiques), DataTables (tableaux) et Leaflet avec un serveur OpenStreetMap locale (cartes).

Dans la partie suivante, nous évaluerons quantitativement et qualitativement les logiciels de notre plateforme LinkedWiki.

3.4 Evaluation

Dans cette section, nous évaluons quantitativement les logiciels qui composent notre solution par rapport aux informations collectées sur leurs usages. Nous les évaluons ensuite qualitativement auprès de scientifiques qui participent à notre expérimentation, toujours en cours.

3.4.1 Evaluation quantitative

La plateforme LinkedWiki pour la description et l'interrogation des données. Après avoir étudié l'audience, nous constatons une utilisation régulière de la plateforme LinkedWiki par les mêmes utilisateurs.

Les pages les plus consultées sont celles qui contiennent des exemples de requêtes SPARQL et qui permettent de les modifier.

Les données d'usage de la figure 3.8 laissent à penser qu'une partie de ces utilisateurs réguliers se servent principalement de la plateforme pour concevoir des requêtes SPARQL afin d'interroger le LOD.

LinkedWiki compte entre 20 et 50 utilisateurs uniques par semaine.

En résumé, le référencement des jeux de données publiques reste faible sur la plateforme et on constate que les utilisateurs l'utilisent principalement pour concevoir

leurs propres requêtes SPARQL afin d'interroger les jeux de données décrits dans la plateforme et les jeux de données du LOD. Dans la section 4.4.2, nous approfondirons l'évaluation de l'éditeur SPARQL et de ses fonctionnalités.

La plateforme LinkedWiki pour la gestion des données. La plateforme permet le déploiement des composants temporaires de VRE depuis janvier 2018. On constate qu'une vingtaine de machines virtuelles temporaires ont été déployées.

Un de ces composants de VRE sert déjà à héberger des données expérimentales d'un projet de recherche en cours. Ce composant sera transféré dans un projet OpenStack permanent afin de rendre ce VRE permanent dans l'IAAS de l'université (nous en reparlerons dans l'évaluation qualitative).

Durée de la session	Uni. Paris Saclay	LinkedWiki.com
0-10 secondes	3 635	2 362
11-30 secondes	343	215
31-60 secondes	268	173
61-180 secondes	485	258
181-600 secondes	459	303
601-1800 secondes	399	313
1801+ secondes	196	200

(a) Nombre de sessions regroupées par durée pour les deux instances déployées de la plateforme pour la période 01/05/2016-24/07/2018 (Source Google Analytics). On constate que de nombreuses sessions dépassent plusieurs dizaines de minutes.

Instances	**Uni. Paris-Saclay**	**LinkedWiki.com**
Utilisateurs enregistrés	90	76
Jeux de données publics	55	25
Exemples de requêtes publiques	534	78

(b) Statistiques des instances datant du 25/07/2018. Les utilisateurs ne sont pas obligés de s'inscrire pour utiliser les instances ni pour enregistrer une requête. L'utilisateur a besoin de s'inscrire uniquement lorsqu'il désire référencer un nouveau jeu de données ou travailler en privé sur ses propres requêtes SPARQL.

Figure 3.8 – La durée des sessions, relativement longue des utilisateurs (a) et un nombre d'utilisateurs enregistrés supérieur au nombre de jeux de données référencés (b) indiquent que les utilisateurs utilisent les instances pour travailler sur des requêtes SPARQL au travers de notre éditeur.

Le *plugin* Wikipédia pour découvrir les données. Concernant l'usage du *plugin* permettant de découvrir les jeux de données de l'instance de l'université, nous avons fait une recherche dans le projet « Wikidata », « meta » et « en.wikipedia » de la fondation Wikimédia pour compter le nombre de pages qui indiquent l'installation du *plugin* « DisplayResearchArtefact.js ». Le 28/07/2018, nous avons trouvé 64 comptes d'utilisateurs avec ce *plugin*. Ce nombre est encourageant si on tient compte du fait que ce *plugin* :

— ne peut être installé que manuellement, et

— ne connecte Wikipédia qu'à seulement 55 jeux de données, c'est-à-dire les 55 jeux de données décrits dans la plateforme LinkedWiki de l'Université Paris-Saclay.

Ce résultat montre un réel usage de ce *plugin*, qui pourrait être étendu en y affichant également les centaines d'exemples de requêtes hébergées par toutes les plateformes (la plateforme publique et celle de l'université).

L'extension pour MediaWiki pour construire une base de connaissances. Nous avons déployé cette extension dans six wikis différents, dont trois pour des chercheurs au sein de l'Université Paris-Saclay.

A l'extérieur de l'université, nous avons eu connaissance du fait que des organisations l'avaient déployé comme, par exemple, dans la « Platform Linked Data Nederland » [104] qui utilise l'extension sur de nombreuses pages. Leur wiki sert à partager les connaissances, les expertises sur le Linked Data et à explorer les possibilités d'application. Ils organisent des réunions, proposent des publications qu'ils référencent au sein de ce wiki. Cette installation témoigne que des experts internationaux du Linked Data ont fait le choix d'installer l'extension LinkedWiki.

Un autre exemple est un wiki privé où l'extension a été déployé (associé à l'extension Semantic MediaWiki) par des chercheurs afin d'analyser les métadonnées des événements scientifiques en informatique et qui ont pu générer directement leurs artefacts graphiques (avec Sgvizler2) pour leur article [44]. Ainsi, notre extension commence à être utilisée par des chercheurs qui travaillent sur les bases de connaissances à leurs dispositions dans le LOD.

Il existe plusieurs manières d'installer une extension MediaWiki. L'une d'entre elles passe par l'outil de téléchargement « distributor » de MediaWiki, qui garde les traces des téléchargements. Ainsi, durant l'année écoulée, l'extension LinkedWiki dans sa dernière version (avec l'intégration de l'extension Sgvizler2) a été téléchargée 24 fois, c'est-à-dire autant de fois que l'extension Semantic MediaWiki sur la même période (source : statistique entre le 01/08/2017 au 30/07/2018 au travers de l'outil « distributor » de MediaWiki lien web). Ce nombre peut sembler limité, mais cela signifie que l'extension LinkedWiki est maintenant autant déployée que l'extension Semantic MediaWiki, qui est la référence dans le domaine depuis plusieurs années.

S'il n'est pas simple de déterminer quelles sont les fonctionnalités les plus utilisées, tous ces indicateurs montrent que cette extension répond à un vrai besoin.

Statistiques de réutilisation des briques logicielles. Voici leurs données d'usages : le client SPARQL 1.1 pour PHP a été déployé 1 103 fois (source : le 15/08/2018 sur packagist.org lien web) et la librairie Sgvizler2 a été déployée 2 014 fois (source : le 15/08/2018 sur npm-stat.com lien web).

Ces chiffres montrent également que ces librairies, open source, sont réutilisées par les développeurs du Linked Data.

Dans la section suivante, nous évaluons notre plateforme qualitativement en restituant l'avis des chercheurs qui l'utilisent pour construire leur VRE.

3.4.2 Évaluation qualitative

Nous évaluons notre approche auprès de scientifiques qui participent à l'expérimentation d'un VRE que nous décrivons dans cette section. Concernant l'éditeur SPARQL, son évaluation qualitative figure en section 4.4.2.

3.4.2.1 Expérimentation

Nous avons travaillé avec plusieurs équipes de chercheurs depuis 2015. Nous avons développé pour eux des démonstrateurs en 2016. Nous les avons aidés à fabriquer leurs premiers schémas RDF. En 2017, le laboratoire Lip(Sys)[2] de la Faculté de Pharmacie de l'Université Paris Saclay a démarré un nouveau projet de recherche et décidé de construire son propre VRE.

Le VRE idéal : DAAP. Le VRE en cours de conception se nomme DAAP, « Data Acquisition for Analytical Platform »(détails en section 3.4.2). C'est un VRE destiné à la chimie analytique.

La figure 3.9 modélise ce VRE. Ce schéma, dont chacune des étapes est décrite plus bas, modélise le processus au sein de ce VRE pour partager au sein du LOD les connaissances qu'ils produisent à l'issue de leurs analyses. Ce VRE est mis à disposition auprès des chercheurs au travers de notre plateforme qui crée et transfère chacun des composants de ce VRE vers un hébergement et un stockage définitif dans l'IAAS de l'université.

Bien que cette figure décrive un VRE particulier, elle n'en reste pas moins assez générique pour que tous les chercheurs puissent y transposer leur propres méthodes de travail et ainsi imaginer leur propre VRE qui sera ainsi compatible avec le Linked Data.

Voici le détail des 9 étapes illustrées en figure 3.9 qui utilisent les mêmes numéros que dans cette liste :

1. Un ou des chercheurs souhaitent mener de nouvelles recherches en respectant un protocole confidentiel : (i) les chercheurs déploient un wiki privé dédié à leur projet, (ii) ils coconstruisent ou étendent le schéma RDF de leur domaine librement afin de décrire leurs nouveaux protocoles, et (iii) ils partagent ce nouveau schéma RDF uniquement avec les logiciels et appareils qu'ils utilisent dans leur laboratoire pour collecter les données expérimentales définies dans leur protocole.
2. Durant la collecte des données expérimentales, leurs outils de collecte consultent le protocole et déplacent/transforment automatiquement les données collectées au sein du serveur de stockage afin de protéger ces nouvelles données et faciliter les futures collaborations.
3. Si le format des données expérimentales le permet, les métadonnées sont extraites et reliées au schéma RDF privé, mais également au schéma RDF partagé au sein de leur laboratoire et au schéma RDF public de leur domaine.
4. La visualisation des connaissances structurées utilisées par les machines au sein de ce VRE passe par la création de modules au sein des wikis qui affichent

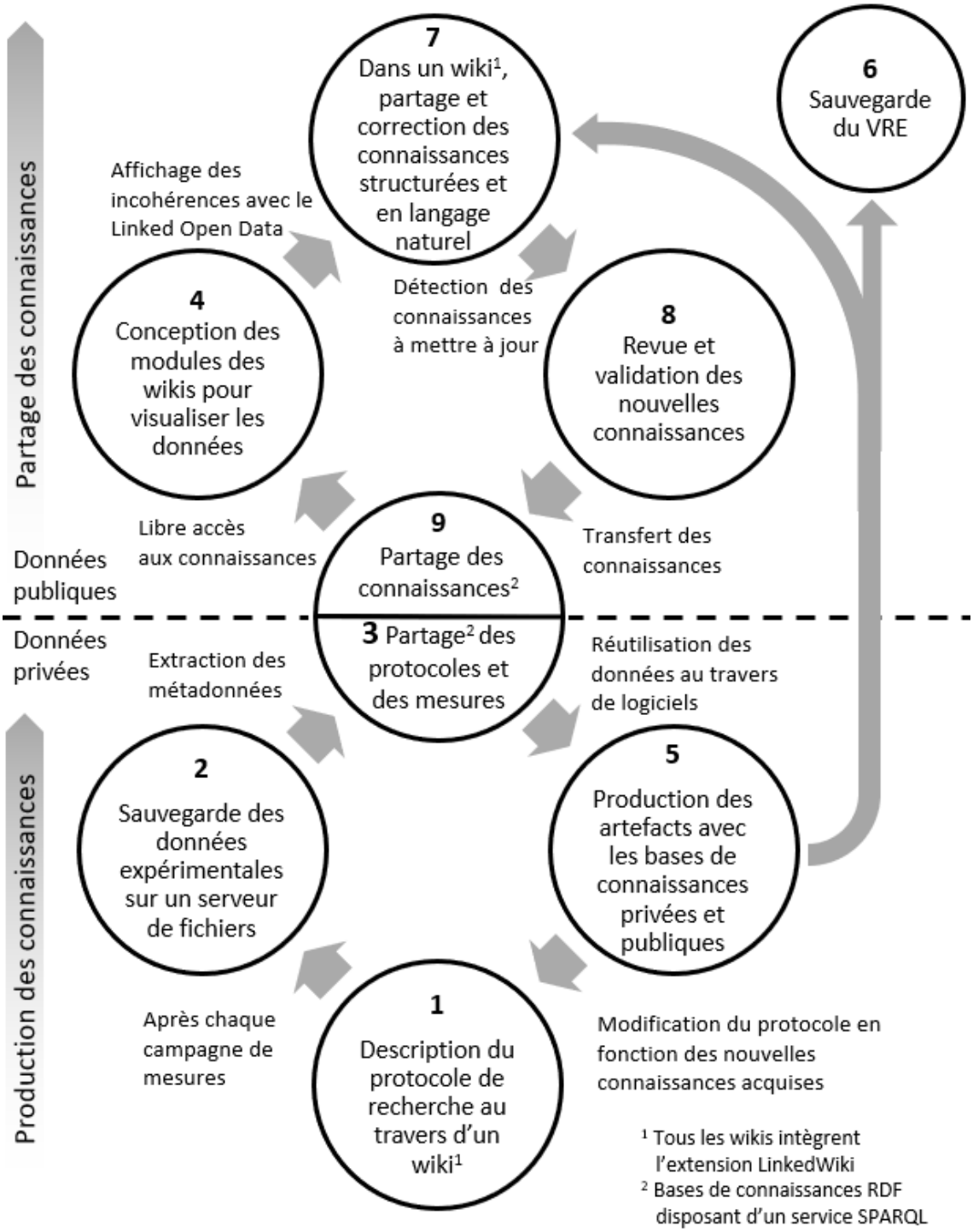


Figure 3.9 – Cette figure est une vue abstraite de l'expérimentation en cours, qui vise à mettre en œuvre un environnement de recherche virtuel (VRE) en y intégrant les technologies Linked Data. Avec ce VRE, les chercheurs travaillent librement au sein du Linked Data de leur université et ouvriront progressivement leurs recherches (après leur publication) au sein du LOD.

des vues des bases de connaissances du VRE et détectent les incohérences avec les connaissances au sein du LOD.

5. Avec leurs logiciels (Matlab, Jupyter, R, etc.), les chercheurs produisent leurs artefacts en interrogeant les données RDF décrivant leurs données expérimentales collectées et les bases de connaissances publiques de leurs domaines. Si un chercheur détecte une erreur sur un fait contenu dans la base de connaissances publique de son VRE, il est en mesure de modifier le wiki public ou privé de son VRE pour en avertir ses collègues.
6. Une fois les artefacts publiés, les chercheurs peuvent demander la sauvegarde de leur VRE de manière permanente pour permettre à d'autres de reproduire ces artefacts. En d'autres termes, il s'agit de sauvegarder les données et toutes les machines virtuelles (qui composent ce VRE dans l'IAAS) et qui contiennent les systèmes d'exploitation, les logiciels et les scripts ayant permis d'obtenir ces artefacts.
7. Les connaissances structurées ou non (en langage naturel) au sein des artefacts du wiki privé peuvent ensuite être transférées automatiquement au sein d'un wiki public (au sein du laboratoire, de l'université ou de leur domaine) organisé par leurs pairs ou par eux-mêmes.
8. Régulièrement, les chercheurs, en charge d'étendre leurs bases de connaissances, détectent les incohérences au sein de leur wiki public avec les bases de connaissances publiques disponibles, la leur et celles des autres, c'est-à-dire avec le LOD. Ils peuvent ainsi corriger les informations contenues dans leurs wikis avant de transférer ces informations structurées dans leur base de connaissances publique et ainsi permettre aux autres bases de connaissances dans le LOD de détecter leurs corrections pour se mettre à jour.
9. Les chercheurs et leurs machines disposent ainsi d'une base de connaissances à jour et fiable compatible avec le Linked Data leur permettant d'interroger simultanément des données privées et des données publiques. Les chercheurs peuvent publier de nouvelles connaissances dans leur VRE.

Avec ce VRE, les chercheurs accèdent et exploitent le Linked Data de leur université et ouvrent progressivement leurs recherches (publiées) au sein du LOD.

Expérimentation en cours. L'université a mis à disposition des chercheurs un NAS (serveur de stockage) privé mutualisé qui sert à stocker leurs données expérimentales.

Un logiciel Windows (en C#) a été développé pour consommer via SPARQL les données RDF du wiki privé qui décrit le protocole. Il permet de transférer chaque fichier de mesure dans le bon répertoire au sein du NAS et de s'assurer que les règles de nommage décrites dans le protocole ont été respectées. A la fin de chaque campagne de mesures, le logiciel permet de contrôler si tous les fichiers de mesures attendus ont bien été collectés.

En juin 2018, les chercheurs de Lip(Sys)^2 publient leurs résultats et ouvrent leurs données. Ils utilisent alors la plateforme LinkedWiki pour déployer un serveur de fichiers Web afin de les consulter via leur wiki public. Pour terminer l'implémentation de l'expérimentation, il faut encore permettre le transfert automatique des

pages entre le wiki privé et le wiki public. Bien que ce VRE ne soit pas encore complètement terminé, les chercheurs l'utilisent déjà depuis 2017. Ils peuvent donc faire une première évaluation de notre approche.

3.4.2.2 Évaluation des chercheurs

Nous restituons ici un entretien organisé avec les chercheurs qui participent à l'expérimentation de ce VRE compatible avec le Linked Data :

> *« Avec ce VRE, nous avons commencé par décrire notre protocole expérimental en langage naturel dans le wiki privé. Nous n'étions pas en mesure de concevoir seul, "from scratch", notre schéma RDF dans le wiki. La raison est due au fait que nous n'avions jamais réalisé auparavant de schéma RDF. Avec de l'aide, nous avons pu définir le schéma RDF afin de décrire la première méthode générale de notre recherche et définir les modules du wiki pour enregistrer les métadonnées des données expérimentales que nous devions collecter.*
>
> *Après plusieurs campagnes en suivant cette méthode générale, un stagiaire a pu dupliquer en quelques heures cette méthode générale et l'adapter pour lancer de nouvelles campagnes de mesures. Il est à noter que ce stagiaire s'était familiarisé aux technologies de MediaWiki environ 2 semaines avant de faire cette manipulation. Nous avons pu ensuite lire sans difficulté les données RDF (au format Turtle) décrites au sein du wiki avant de démarrer une nouvelle campagne.*
>
> *Pour modifier le schéma RDF, il sera nécessaire d'imaginer une interface plus ergonomique au sein du wiki, ou bien de développer un plugin pour nous permettre d'importer et sauvegarder du RDF au sein du wiki au travers d'un logiciel existant comme Protégé ou TopBraid Composer.*
>
> *La modification des modules (codés en Lua) du wiki nécessite l'intervention d'un informaticien pendant quelques heures et environ deux fois par an à chaque changement de protocole. Cet investissement est largement rentabilisé par le gain de temps qu'apporte le logiciel installé sur les machines d'acquisition qui transfère et contrôle les données expérimentales sur le serveur de stockage sécurisé et définitif (NAS) de l'Université Paris-Sud.*
>
> *Les outils que nous utilisons pour concevoir les artefacts ne sont pas encore adaptés au Linked Data. Nous avons avec ces données expérimentales générés nos artefacts et publiés nos recherches. Nous avons confié à un stagiaire le travail d'obtenir les mêmes artefacts en sélectionnant les données expérimentales à l'aide de leurs métadonnées stockées dans les bases de connaissances du VRE. L'objectif est de produire la documentation nécessaire pour nous permettre de le faire nous-même durant la prochaine campagne de mesures.*
>
> *La publication des données expérimentales s'est faite en utilisant un composant de VRE au travers de la plateforme LinkedWiki qui a déployé le serveur de fichiers. Pour transférer les fichiers du NAS privé*

à ce serveur, nous avons utilisé Filezilla (via SFTP). Les données du protocole expérimentale du wiki privé sont prêtes à être transférées mais l'extension LinkedWiki nécessite d'être encore étendue pour faire cette opération de manière automatique (amélioration prévue avant décembre 2018).

Le référencement des données ouvertes via la plateforme se fait sans difficulté et les exemples de requêtes nous permettent de nous familiariser avec ces technologies.

La conclusion de cette première expérience est que nous allons poursuivre cette expérimentation avec les prochaines campagnes de mesures prévues pour début 2019. La raison de ce choix est dû principalement à la simplification de la gestion nécessaire à la pérennisation des données sur plusieurs années. Sur cette durée, le respect du même protocole entre chercheurs se succédant est une condition indispensable pour obtenir des résultats pharmaceutiques fiables (reproductibles), et ce VRE nous permet de répondre à cet objectif et à l'ouverture de nos données auprès de notre communauté. »

Cette restitution d'utilisateurs montre que notre contribution est très prometteuse. Les chercheurs l'utilisent déjà car cela simplifie la gestion de leurs données, et les données déjà compatibles avec le LOD pourront progressivement être ouvertes sans effort.

Nous concluons dans la section suivante.

3.5 Conclusion

Nous avons décrit la solution LinkedWiki pour le management des données de la recherche, basée sur les technologies du Linked Data. Elle permet aujourd'hui de publier, découvrir et réutiliser plus simplement les données partagées par des chercheurs. Elle est encore en cours d'expérimentation.

Nous avons présenté les trois contributions principales de notre approche.

La première contribution est une plateforme permettant de référencer toutes les données de la recherche et de les relier au LOD, offrant ainsi de nouvelles manières de rechercher et naviguer vers ces données. Nous avons mis cet aspect en évidence avec le *plugin* qui transforme Wikipédia en moteur de recherche vers les données ouvertes au sein d'une université.

La seconde est un système de gestion des informations et des connaissances dans le cloud d'une université. Ce système simplifie le travail du chercheur, qui peut déployer à la demande son propre VRE.

La dernière contribution est un environnement permettant de concevoir, tester, partager, réutiliser et s'abonner à des requêtes SPARQL au travers de la plateforme. Cet outil aide les utilisateurs à interroger leurs données et le LOD, mais facilite également la maintenance de leurs applications qui utilisent des schémas RDF mouvants comme ceux de Wikidata.

De cette première implémentation de la solution LinkedWiki, nous pouvons déjà tirer quelques enseignements.

1. La découverte des données (au sein de l'université) est grandement facilitée. À long terme, nous espérons encourager une analyse automatique du système d'information de l'université pour identifier automatiquement les données scientifiques disponibles qu'il suffit de référencer pour les mettre à disposition au travers du Linked Data de l'université, et peut-être les ouvrir à tous dans un deuxième temps.
2. Les scientifiques qui utilisent et testent nos outils perçoivent l'intérêt de rendre leurs données compatibles avec le Linked Data et commencent à réfléchir sur la manière de travailler avec ces technologies.
3. L'utilisation par les scientifiques de schémas RDF connectés à Wikidata conduit à améliorer tous les schémas RDF et à diffuser le Linked Data dans tous les systèmes d'information. Plus les liens entre les systèmes d'information augmenteront, plus les données des scientifiques seront interopérables et plus les machines pourront jouer leur rôle qui consiste à aider les chercheurs. Cet aspect n'est pas encore évident pour la plupart des scientifiques, qui y voient plus de contraintes que d'avantages, même si l'on peut s'attendre à ce qu'ils soient les premiers bénéficiaires des résultats qui découleront de la diffusion de ces technologies. Ce point est encore à investiguer.

A l'avenir, nous poursuivrons le développement de LinkedWiki pour le rendre encore plus simple à utiliser. Plus particulièrement, nous voulons améliorer la quantité et la qualité des métadonnées automatiquement associées par le système.

Dans les chapitres suivants, nous décrivons, en chapitre 4, nos travaux sur l'aide à la conception de requêtes SPARQL et en chapitre 5, nos travaux sur l'interopérabilité réelle des services SPARQL dans le Linked Data.

Résumé du chapitre 3

Dans ce chapitre, nous présentons une nouvelle solution pour permettre une gestion durable des données de la recherche tout en étant compatible avec le Linked Data.
La première de nos contributions est une plateforme permettant de cataloguer des données RDF ou non pour faciliter leur découverte, comme au travers de Wikipédia, et pour faciliter leur réutilisation au travers d'un catalogue de requêtes SPARQL.
La seconde contribution est une solution permettant aux chercheurs de déployer un VRE modulaire et compatible avec le Linked Data au sein du cloud de l'université.
La dernière contribution est un environnement permettant de concevoir, tester, partager, réutiliser et faciliter la maintenance des requêtes SPARQL qui utilisent des schémas RDF mouvants comme ceux de Wikidata.
La plateforme est en cours d'expérimentation au sein du *Center for Data Science* (CDS) de l'Université Paris-Saclay dans le laboratoire partenaire Lip(Sys)^2 de la Faculté de Pharmacie. Les premiers résultats confirment que notre approche facilite le travail des chercheurs ainsi que la mise à disposition de leurs données au sein du Linked Data de l'université.

Résultats publiés du chapitre 3

Démonstrations :
[118] « Une autocomplétion générique de SPARQL dans un contexte multi-services », Karima Rafes, Sarah Cohen-Boulakia et Serge Abiteboul. Dans BDA. 2017.
[119] « A platform for scientific data sharing ». Karima Rafes et Cécile Germain. Dans : BDA (Bases de Données Avancées). 2015.

[140] Poster : « Data acquisition for analytical platforms : Automating scientific workflows and building an open database plat-form for chemical analysis metadata ». Sana Tfaili, Diem Bui Thi, Karima Rafes et al. Dans Chimiométrie XVII. Poster. 2016.

Chapitre 4

SPARQLet-Finder : autocomplétion par snippets pour SPARQL

Sommaire

4.1 Introduction

La quantité de données liées disponibles sur le Web augmente constamment. Interroger ces données est un besoin crucial afin d'exploiter cette richesse. Lorsque les données liées sont représentées en RDF, SPARQL est le langage de requête le plus populaire. Cependant, écrire une requête SPARQL peut être fastidieux, même pour les utilisateurs expérimentés. Les raisons sont nombreuses, mais les deux principales sont la connaissance imparfaite du *schéma RDF* (la structure des données) impliqué dans la requête et la nécessité de maîtriser une syntaxe informatique.

Pour répondre à ces problèmes, les systèmes proposent souvent des interfaces masquant le langage SPARQL : l'utilisateur est invité à dessiner des schémas [37,

32], écrire en langage naturel [46, 98], sélectionner des exemples de résultats attendus [71], ou bien sélectionner des requêtes préfabriquées [33].

L'inconvénient de ces approches est double. Tout d'abord, les utilisateurs ne progressent pas dans l'apprentissage du langage SPARQL, bien que le nombre d'applications intéressantes à fabriquer augmente constamment. Deuxièmement, les utilisateurs sont limités dans leurs recherches, non seulement par le schéma RDF supporté par l'interface, mais aussi et surtout par les fonctions supportées par l'interface. En d'autres termes, les utilisateurs sont limités dans le sens où ils ne peuvent interroger les données qu'au travers des requêtes imaginées par les développeurs des systèmes.

La solution que nous proposons permet aux utilisateurs d'exploiter toute la puissance et l'expressivité du langage SPARQL tout en les guidant dans la conception de nouvelles requêtes, en proposant un large panel de fonctionnalités d'*autocomplétion*.

Le point de départ de la contribution que nous décrivons ici est une étude de deux ans que nous avons menée avec des utilisateurs d'éditeurs SPARQL. Cette étude a été réalisée dans le cadre de l'utilisation de la plateforme LinkedWiki (*cf.* section 3.2) par différents concepteurs de requêtes, novices ou experts, issus de diverses disciplines (informatique, biologie, chimie, physique, sciences humaines et sociales, etc.) [119]. Cette étude a souligné la nécessité de mettre en œuvre une fonctionnalité qui serait très utile pour les utilisateurs avancés, à savoir l'*autocomplétion par snippets*, qui consiste à exploiter les fragments de requêtes partagées par des utilisateurs précédents afin de compléter une nouvelle requête déjà écrite partiellement. Le présent chapitre décrit en détail notre solution.

L'autocomplétion par snippets n'est pas considérée par les éditeurs SPARQL existants (*cf.* section 3.2). Cependant, cette fonctionnalité a été prise en compte par d'autres éditeurs de langages de requêtes [58, 82, 1, 43, 143] et, plus généralement, elle a été étudiée dans le contexte de l'exploration de données [66].

Dans ce chapitre, nous introduisons la première autocomplétion (à notre connaissance) à l'aide de snippets pour SPARQL. Pour cela, nous extrayons une représentation des sous-patrons des graphes contenus dans des requêtes précédentes (sous la forme de « linegraphs ») et construisons une classification hiérarchique de tels sous-patrons pour générer les snippets les plus adaptés à la partie de la requête que l'utilisateur est en train d'écrire. Ces snippets sont ensuite suggérés à l'utilisateur, via une autocomplétion, ce qui rend le processus d'édition d'une nouvelle requête nettement plus simple et plus rapide.

Ce chapitre est organisé comme suit. La section 4.2 introduit le contexte dans lequel s'est inscrit notre travail et fournit un exemple concret du domaine biomédical. La section 4.3 introduit SPARQLets-Finder, la première approche d'autocomplétion à l'aide de snippets pour SPARQL. La section 4.4 évalue notre approche à la fois quantitativement, en fournissant des mesures obtenues sur un grand nombre de cas d'utilisation, et qualitativement, sur la base de requêtes présentées dans la section 4.2.

4.2 Contexte de l'expérimentation

Contexte général. Depuis 2015, nous aidons des utilisateurs à écrire, partager et découvrir de nouvelles requêtes SPARQL en utilisant deux instances de la *plateforme LinkedWiki* : une instance dédiée aux scientifiques et étudiants de l'Université Paris-Saclay [114] et une autre instance accessible à tous [106]. Nous avons réalisé une étude des besoins d'autocomplétion lors de la rédaction des requêtes, en collectant au cours des deux dernières années les souhaits des utilisateurs, des novices jusqu'aux experts. La plupart des utilisateurs étaient des scientifiques de différents domaines (notamment, astronomie, sciences de la vie, sciences sociales) qui peuvent avoir peu de connaissances dans le domaine de la gestion des données. L'une des conclusions de cette étude est la suivante : quel que soit le domaine des scientifiques, il est urgent de disposer d'un outil d'autocomplétion par snippets pour les aider à écrire leurs propres requêtes en bénéficiant des requêtes écrites par leurs pairs.

L'approche présentée dans ce chapitre d'une autocomplétion de requêtes a été conçue pour répondre à ce besoin et dans l'optique de fonctionner dans n'importe quel domaine (scientifique ou autres). Le contexte du Center for Data Science de l'Université Paris-Saclay s'est avéré approprié pour capturer cette généralité. Pour illustrer notre propos, nous considérerons dans ce qui suit une illustration de ce besoin dans le domaine biomédical.

Dans le domaine biomédical. Le cas d'utilisation que nous considérons se situe dans le contexte de la médecine personnalisée, et plus précisément dans l'étude de la variabilité de la réponse à plusieurs médicaments anticancéreux en fonction du patient (c'est-à-dire en fonction de ses variants génétiques pour des gènes spécifiques, en fonction des mutations génétiques associées à ses tumeurs, etc.). Ici, les connaissances relatives aux réponses aux médicaments sont dispersées dans des centaines d'articles scientifiques indexés dans le référentiel PubMed (une bibliothèque publique de publications biomédicales contenant des millions d'articles, voir [97]), dont beaucoup sont référencés dans Wikidata, qui permet d'interroger ce référentiel via son service SPARQL [18].

Lors de la conception d'une requête afin de trouver des informations pertinentes, un médecin connaissant SPARQL rencontre plusieurs difficultés. Il a besoin d'avoir une connaissance approfondie de l'ontologie sous-jacente de Wikidata (l'organisation de l'information), et, plus généralement, une connaissance exhaustive des informations attachées aux médicaments ainsi que des relations entre les mutations génétiques et leurs références dans PubMed [97].

Pour faire cette requête, le médecin recherche d'abord les maladies qui sont des formes de cancers, et trouve les médicaments utilisés pour traiter ces cancers. Bien que cette première (sous-)requête soit relativement simple, le médecin doit déjà connaître la structure de l'ontologie de Wikidata (par exemple, le type `rdfs:subClassOf`) et, particulièrement, il doit avoir une connaissance approfondie des identifiants des « éléments » contenus dans Wikidata comme « cancer » (identifié par `wd:Q12078`), « médicament », et des prédicats comme « *maladie traitée par le médicament* » (identifié par `wdt:P2175`), etc. Sans outil d'autocomplétion et sans exemple de requêtes, il est peu probable qu'un médecin, même expérimenté en re-

quêtes SPARQL, réussisse à écrire une requête appropriée rapidement.

Pour compléter ces informations, le médecin cherche à associer les variantes génétiques associées à un médicament qui donne les meilleurs résultats thérapeutiques, et les références concernant la provenance de ces résultats. Encore une fois, sans l'aide de fonctionnalités d'autocomplétion, une telle requête semble très compliquée à concevoir et peut même être au-delà des compétences du médecin.

Ce cas d'utilisation (la recherche de ce type d'infomation) fait partie des requêtes classiques. En particulier, la recherche de l'origine des informations (références, provenance) a probablement déjà été effectuée par des utilisateurs précédents. Le fait de suggérer des extraits de telles requêtes, c'est-à-dire des snippets, simplifierait énormément la tâche des utilisateurs ayant besoin d'écrire de nouvelles requêtes.

C'est dans ce contexte que nous avons développé une nouvelle approche, que nous décrivons dans la partie suivante, afin de capter et mettre à disposition des utilisateurs les structures réutilisables des requêtes.

4.3 L'algorithme

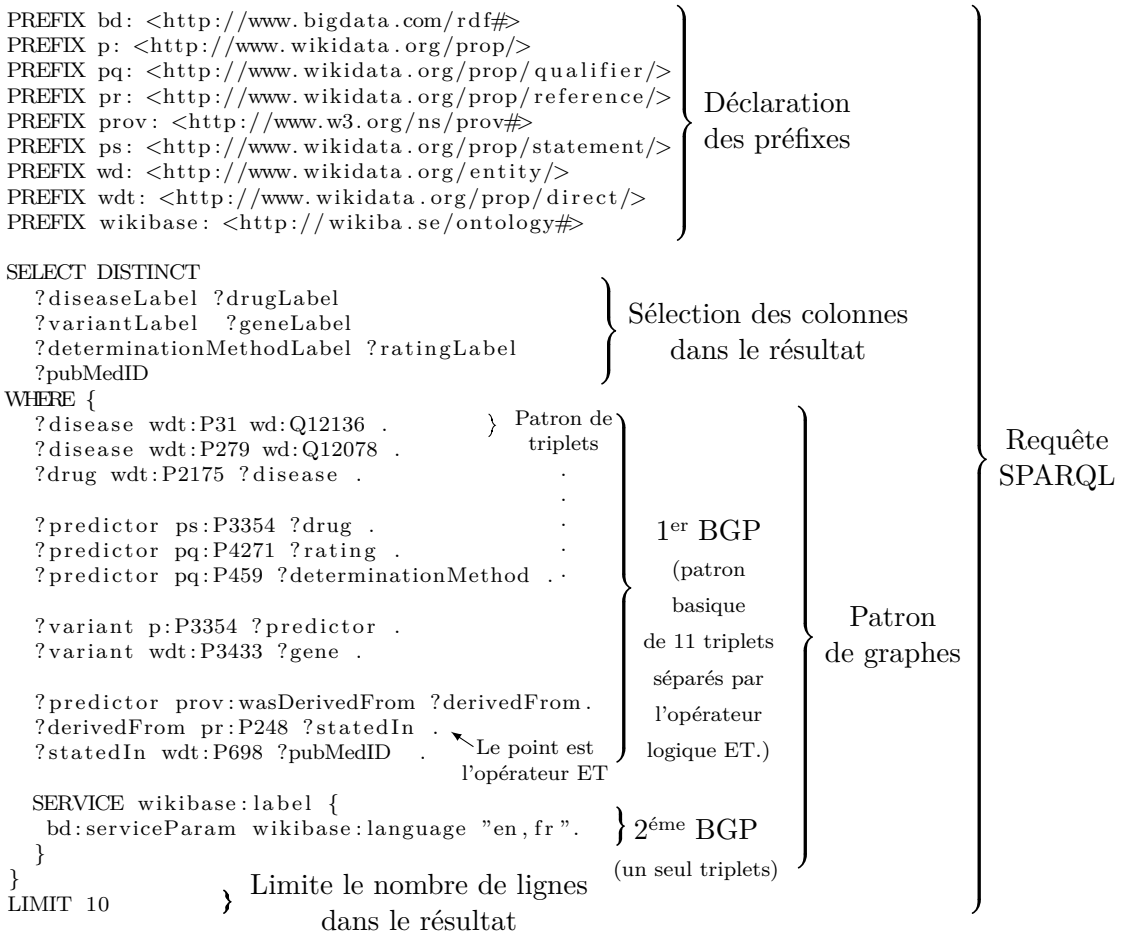
Dans cette section, nous présentons la première solution (à notre connaissance) d'autocomplétion à base de snippets. Après la présentation d'une vue d'ensemble de l'algorithme, nous décrirons l'extraction des sous-patrons présents dans les requêtes connues (sous la forme de « linegraphs ») afin de construire une classification hiérarchique (*hierarchical clustering*) des sous-patrons de requêtes. Ensuite, nous montrons comment générer les snippets adaptés à la partie de la requête en train d'être écrite par un utilisateur.

4.3.1 Vue d'ensemble

Une requête SPARQL est basée sur un patron de graphes qui s'applique à un graphe de connaissances. La figure 4.1 montre une requête qui contient un tel patron de graphes. Ce patron est écrit en combinant plusieurs sous-patrons de différentes tailles. Le plus petit de ces patrons, le *patron basique de graphes*, que l'on abrégera BGP (comme en anglais, *Basic Graph Pattern*), est un ensemble de *patrons de triplets* impliquant à la fois des variables et des constantes. Par conséquent, la structure du graphe représentant un BGP aura des sommets représentant des variables et des constantes.

Dans les requêtes SPARQL écrites par des utilisateurs, nous constatons de nombreuses similitudes entre les BGP au sein des requêtes appliquées à différents domaines. Cela nous a amené à l'idée de suggérer aux utilisateurs des extraits de requêtes précédentes. Pour découvrir ces extraits, nous utilisons une classification hiérarchique basée sur une distance de similarité entre les BGP. L'utilisation d'un tel regroupement hiérarchique a été conçue dans l'esprit de [61, 23, 132, 8].

L'efficacité de cette méthode est clairement conditionnée à la qualité de la distance de similarité utilisée. Définir une telle distance entre des patrons de graphes n'est pas trivial [38] et cela nous a demandé de nombreuses expérimentations avant de trouver la bonne méthode.



(a) Exemple d'une requête SPARQL où nous avons supprimé tous les sucres syntaxiques pour afficher clairement les séquences de patrons de triplets dans chaque BGP (la figure 4.10 reprend cette requête avec les sucres syntaxiques).

diseaseLabel	drugLabel	variantLabel	geneLabel	determinationMethodLabel	ratingLabel	ArticlePubMed
subependymal giant cell astrocytoma	temozolomide	NRAS Q61L	NRAS	CIViC evidence level C	CIViC 2-star trust rating	https://www.ncbi.nlm.nih.gov/pubmed/?term=21576590
lung oat cell carcinoma	cisplatin	HSPH1 NUCLEAR EXPRESSION	HSPH1	CIViC evidence level D	CIViC 1-star trust rating	https://www.ncbi.nlm.nih.gov/pubmed/?term=26943774
Hodgkin's lymphoma, mixed cellularity	cisplatin	XRCC1 Q399R	XRCC1	CIViC evidence level B	CIViC 4-star trust rating	https://www.ncbi.nlm.nih.gov/pubmed/?term=16875718
bladder cancer	cisplatin	XRCC1 Q399R	XRCC1	CIViC evidence level B	CIViC 4-star trust rating	https://www.ncbi.nlm.nih.gov/pubmed/?term=16875718
breast cancer	cisplatin	HSPH1 NUCLEAR EXPRESSION	HSPH1	CIViC evidence level D	CIViC 1-star trust rating	https://www.ncbi.nlm.nih.gov/pubmed/?term=26943774

(b) Résultat de la requête du dessus, avec les données biomédicales dans Wikidata [164] (lien Web).

Figure 4.1 – Une requête SPARQL est construite autour de patrons basiques de graphes (BGP) dont chacun est constitué par un ensemble de patrons de triplets.

Dans ce chapitre, nous décrirons trois contributions nécessaires à l'implémentation de notre approche d'autocomplétion par snippets. La première contribution est une méthode pour mettre en évidence les principales caractéristiques des BGP (section 4.3.2) et ainsi pouvoir calculer une distance de similarité utilisée pour comparer les BGP. La seconde contribution est un algorithme de Classification Hiérarchique des BGP, qu'on nommera BGPHC (Section 4.3.3), que nous utilisons pour organiser les patrons communs de triplets trouvés au sein des BGP des requêtes contenues dans notre base de connaissances. Cette hiérarchie s'avère indispensable pour déterminer les snippets les plus adaptés au cours de la rédaction d'un BGP par un utilisateur. Nous pensons que cette méthode pourrait également être adaptée dans d'autres contextes, comme au sein d'un moteur de recherche capable de proposer des requêtes.

Ces deux premières contributions sont indispensables à l'implémentation de notre approche au travers d'un workflow (section 4.3.4), qui constitue notre troisième contribution.

Dans la partie suivante, nous commençons par expliquer la manière dont nous calculons une distance de similarité entre les BGP contenus dans les requêtes connues.

4.3.2 Structure commune des BGP et mesure de distance

Parmi toutes les approches pour comparer les modèles de requêtes [38], la méthode [89] est la plus intéressante dans notre contexte car elle calcule la similarité en fonction du modèle structurel du BGP tout en restant agnostique au contexte des données (c.-à-d., agnostique aux schémas RDF). En conséquence, cette méthode peut s'appliquer à toutes les requêtes quels que soient leurs services SPARQL. Nous avons ainsi décidé de suivre leur méthode permettant d'extraire les caractéristiques principales des BGP afin de trouver les structures communes entre des requêtes déjà connues et donc probablement aussi communes à de futurs requêtes.

Les premières expérimentations avec cette technique ont montré qu'elle n'était pas complètement satisfaisante, en raison de certains prédicats dans les patrons de triplets qui sont réutilisés en permanence. Les prédicats de type « *est un* » (par exemple, `rdf:type` ou `wdt:P31`) utilisés dans la majorité des requêtes créent des similitudes inutiles qui impactent la qualité du résultat. Nous avons donc adapté la méthode comme suit.

Linegraphs. À partir d'un BGP, nous construisons un graphe, un *Linegraph*, qui est une représentation raffinée du BGP et qui permet de capturer des similarités significatives. Plus précisément, le linegraph d'un BGP G, noté $\mathcal{L}(G)$, est un graphe tel que chaque sommet de $\mathcal{L}(G)$ représente un arc du BGP ; et deux sommets de $\mathcal{L}(G)$ sont adjacents si et seulement si leurs arcs correspondants partagent un sommet commun de type variable dans le BGP G. Chaque sommet n du graphe est libellé comme suit :

— 1er cas : Si, dans le BGP G, n représente un prédicat de type « *est un* » qui est associé à un objet constant, alors l'étiquette de n est la concaténation du label du prédicat et du label de l'objet constant.

— 2e cas : Si n représente un autre prédicat p dans le BGP G, alors l'étiquette de n reprend l'étiquette de p.

En ce qui concerne les arcs, quatre types d'arcs sont considérés dans un linegraph pour exprimer les quatre types de jointures possibles entre deux triplets dans un BGP partageant un sommet commun (objet ou sujet). Nous nommons ces jointures : *sujet-sujet*, *sujet-objet*, *objet-sujet* et *objet-objet*. Une jointure x-y (pour x, y étant un sujet, un objet, ou les deux) est utilisée lorsque le sommet commun entre les deux triplets est un « x » pour le premier patron de triplets, et un « y » pour le second. Les arcs sont ensuite étiquetés comme suit : l_0 pour *sujet-sujet*, l_1 pour *sujet-objet*, l_2 pour *objet-sujet* et l_3 pour *objet-objet*.

Nous illustrons le concept de BGP associé à un ensemble de patrons de triplets et à son linegraph dans la figure 4.2.

Mesure de la distance entre les linegraphs de BGP

Nous décrivons maintenant une nouvelle mesure de distance, la LBGPD (pour *Linegraph BGP Distance*), conçue pour identifier les similarités structurelles entre BGP à l'aide de leurs linegraphs.

Comme pour la méthode de similarité [89], nous avons choisi d'utiliser la *distance de Jaccard*, noté J_δ, qui mesure la dissimilarité entre deux ensembles :

$$J_\delta(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.1}$$

En indexant au préalable l'ensemble des arcs des linegraphs des BGP, l'algorithme 4.1 calcule la LBGPD entre deux BGP en appliquant la distance de Jaccard en fonction des arcs en commun dans les linegraphs de ces deux BGP.

$$LBGPD(BGP_x, BGP_y) = J_\delta(\mathcal{L}_x, \mathcal{L}_y) = 1 - \frac{|\mathcal{L}_x \cap \mathcal{L}_y|}{|\mathcal{L}_x \cup \mathcal{L}_y|} \tag{4.2}$$

Algorithme 4.1 Calcul de LBGPD en utilisant $\text{Jaccard}_\delta(\mathcal{L}_x, \mathcal{L}_y) : \delta \in [0..1]$

```
1 : nombreDesArcsPrésentsDansCesLinegraphs ←| Lx ∪ Ly |
2 : nombreDesArcsEnCommun ←| Lx ∩ Ly |
3 : retourne 1 - ( nombreDesArcsEnCommun / nombreDesArcsPrésentsDansCesLinegraphs )
```

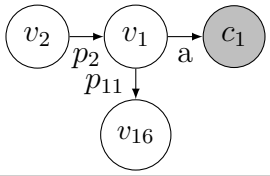
Pour illustrer le concept de LBGPD, nous calculons la distance entre BGP_2 et BGP_3 de la figure 4.3 en utilisant leurs linegraphs qui ont été calculés préalablement :

$$LBGPD(BGP_2, BGP_3) = J_\delta(\mathcal{L}_2, \mathcal{L}_3) = 1 - \frac{|\mathcal{L}_2 \cap \mathcal{L}_3|}{|\mathcal{L}_2 \cup \mathcal{L}_3|} = 1 - \frac{8}{26} \simeq 0.69 \tag{4.3}$$

À l'aide de cette distance de similarité, nous pouvons maintenant décrire la classification hiérarchique des BGP, qui nous fournira une représentation organisée de leurs linegraphs communs.

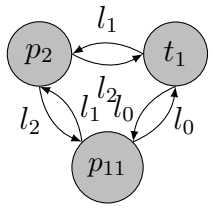
Processus **A.1.** : extraction de BGP_1 d'une requête connue

(a) Exemple de BGP extrait	**(b)** BGP_1	**(c)** BGP_1 graphiquement

`?disease wdt:P31 wd:Q12136 .`
`?drug wdt:P2175 ?disease .`
`?disease wdt:P18 ?image .`
	v_1 a c_1
	v_2 p_2 v_1
	v_1 p_{11} v_{16}
		

(c) est un exemple graphique du patron de graphes BGP_1 en (b) extrait d'une requête (a). Pour passer du BGP (a) à (b), la variable `?Disease` a été associée à v_1, le prédicat `wdt:P31` à a, l'objet constant `wd:Q12136` à c_1. v_2, v_{16}, p_2 et p_{11} sont construits de la même manière.

Processus **A.2.** : construction du linegraph $\mathcal{L}_1$ à partir de BGP_1



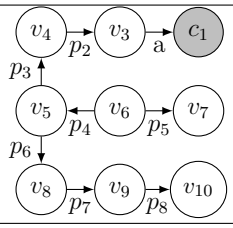
Le linegraph de BGP_1 (à gauche) est construit en fonction des arcs adjacents par l'intermédiaire des variables et étiquetés comme suit : l_0 pour *sujet-sujet*, l_1 pour *sujet-objet*, l_2 pour *objet-sujet* et l_3 pour *objet-objet*. Chaque prédicat est transformé en un sommet (par exemple, le prédicat `wdt:P2175` est transformé en p_2). Pour traiter le cas spécifique des prédicats de type "*est un*" `wdt:P31` et sa constante `wd:Q12136`, nous créons un nouveau sommet t_1 qui représente l'association de `wdt:P31` avec `wd:Q12136`. Pour illustrer l'étiquetage des arcs, on peut prendre l'exemple de l'arc $p_2 \rightarrow t_1$ qui a l'étiquette l_2 parce qu'il remplace le sommet $v1$ qui est une variable à la position de l'objet dans le patron du triplet de p_2 et à la position du sujet dans le patron du triplet de c_1.

Figure 4.2 – Illustration du processus **A.1** qui extrait les BGP des requêtes SPARQL et du processus **A.2** qui construit le linegraph de chacun des BGP connus.

Exemples de deux BGP après leur extraction de requêtes connues : BGP_2 et BGP_3.

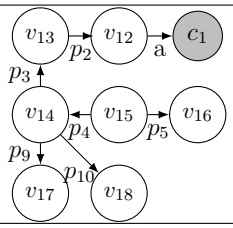
```
?disease wdt:P31 wd:Q12136 .
?drug wdt:P2175 ?disease .
?predictor ps:P3354 ?drug .
?variant p:P3354 ?predictor .
?variant wdt:P3433 ?gene .
?predictor prov:wasDerivedFrom ?derivedFrom.
?derivedFrom pr:P248 ?statedIn .
?statedIn wdt:P698 ?pubMedID .
```

v_3 a c_1
v_4 p_2 v_3
v_5 p_3 v_4
v_6 p_4 v_5
v_6 p_5 v_7
v_5 p_6 v_8
v_8 p_7 v_9
v_9 p_8 v_{10}

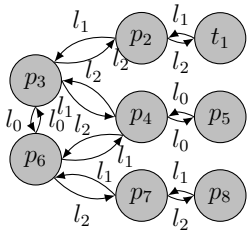


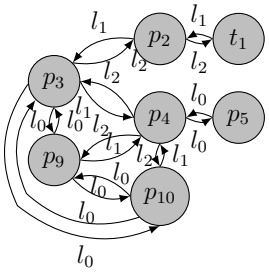

```
?disease wdt:P31 wd:Q12136 .
?drug wdt:P2175 ?disease .
?predictor ps:P3354 ?drug .
?variant p:P3354 ?predictor .
?variant wdt:P3433 ?gene .
?predictor pq:P4271 ?rating .
?predictor pq:P459 ?determinationMethod .
```

v_{12} a c_1
v_{13} p_2 v_{12}
v_{14} p_3 v_{13}
v_{15} p_4 v_{14}
v_{15} p_5 v_{16}
v_{14} p_9 v_{17}
v_{14} p_{10} v_{18}



On peut ensuite calculer leurs linegraphs : $\mathcal{L}_2$ et $\mathcal{L}_3$





On indexe tous les arcs de tous les linegraphs : $\mathcal{L}_1$, $\mathcal{L}_2$ et $\mathcal{L}_3$

$Arc_0 \leftarrow Tuple3(p_2, l_2, t_1)$
$Arc_1 \leftarrow Tuple3(t_1, l_1, p_2)$
$Arc_2 \leftarrow Tuple3(t_1, l_0, p_{11})$
$Arc_3 \leftarrow Tuple3(p_{11}, l_0, t_1)$
$Arc_4 \leftarrow Tuple3(p_2, l_2, p_{11})$
$Arc_5 \leftarrow Tuple3(p_{11}, l_1, p_2)$
$Arc_6 \leftarrow Tuple3(p_2, l_1, p_3)$
$Arc_7 \leftarrow Tuple3(p_3, l_2, p_2)$
$Arc_8 \leftarrow Tuple3(p_3, l_1, p_4)$
$Arc_9 \leftarrow Tuple3(p_4, l_2, p_3)$
$Arc_{10} \leftarrow Tuple3(p_4, l_0, p_5)$
$Arc_{11} \leftarrow Tuple3(p_5, l_0, p_4)$
$Arc_{12} \leftarrow Tuple3(p_3, l_0, p_6)$
$Arc_{13} \leftarrow Tuple3(p_6, l_0, p_3)$
$Arc_{14} \leftarrow Tuple3(p_6, l_1, p_4)$
$Arc_{15} \leftarrow Tuple3(p_4, l_2, p_6)$
$Arc_{16} \leftarrow Tuple3(p_6, l_2, p_7)$
$Arc_{17} \leftarrow Tuple3(p_7, l_1, p_6)$
$Arc_{18} \leftarrow Tuple3(p_7, l_2, p_8)$
$Arc_{19} \leftarrow Tuple3(p_8, l_1, p_7)$
$Arc_{20} \leftarrow Tuple3(p_3, l_0, p_9)$
$Arc_{21} \leftarrow Tuple3(p_9, l_0, p_3)$
$Arc_{22} \leftarrow Tuple3(p_9, l_1, p_4)$
$Arc_{23} \leftarrow Tuple3(p_4, l_2, p_9)$
$Arc_{24} \leftarrow Tuple3(p_4, l_2, p_{10})$
$Arc_{25} \leftarrow Tuple3(p_{10}, l_1, p_4)$
$Arc_{26} \leftarrow Tuple3(p_9, l_0, p_{10})$
$Arc_{27} \leftarrow Tuple3(p_{10}, l_0, p_9)$
$Arc_{28} \leftarrow Tuple3(p_3, l_0, p_{10})$
$Arc_{29} \leftarrow Tuple3(p_{10}, l_0, p_3)$

En sortie du processus A.2, chaque linegraph est une liste des index de ses arcs :

$\mathcal{L}_1 \leftarrow \{0, 1, 2, 3, 4, 5\}$
$\mathcal{L}_2 \leftarrow \{0, 1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$
$\mathcal{L}_3 \leftarrow \{0, 1, 6, 7, 8, 9, 10, 11, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29\}$

Figure 4.3 – Exemples de transformation des BGP BGP_2 et BGP_3 en linegraphs $\mathcal{L}_2$ et $\mathcal{L}_3$ au travers du processus **A.1** et **A.2** (BGP_1 et $\mathcal{L}_1$ se trouvent dans la figure 4.2).

4.3.3 BGPHC : classification hiérarchique de BGP

Une classification hiérarchique et descendante. Pour obtenir des snippets, nous utilisons une classification hiérarchique, qu'on abrégera par BGPHC, inspirée par la méthode de classification hiérarchique descendante [8]. Dans cette classification, chaque cluster calculé est un ensemble de linegraphs qui partagent des arcs.

Chaque cluster est caractérisé par un ensemble d'arcs communs. Ceci définit la hiérarchie. La classification est *floue* car un linegraph donné peut partager des arcs avec d'autres linegraphs et donc ce linegraph peut appartenir à plusieurs clusters simultanément. Puisque le nombre de clusters dans le résultat d'un BGPHC est inconnu a priori, nous avons dû implémenter une *classification non supervisée* en utilisant l'approche *descendante* de classification hiérarchique. Ainsi, nous commençons notre classification avec un cluster initial contenant tous les linegraphs connus, puis nous divisons récursivement les clusters jusqu'à l'arrêt de l'algorithme. Notre méthode de division permet de construire une classification hiérarchique des BGP et, simultanément, une classification des linegraphs communs entre les BGP, des plus partagés aux moins partagés.

L'algorithme de classification hiérarchique récursif. L'algorithme 4.2 est un algorithme de classification récursif de linegraphs qui détermine leurs sous-graphes communs de taille minimum (non vides) afin d'englober un maximum de ces linegraphs dans chaque cluster calculé. Dans chaque cluster v_i (chaque sommet de la structure hiérarchique), nous notons $\mathcal{L}_c(v_i)$ le sous-linegraph commun calculé, c'est-à-dire le sous-graphe des linegraphs partageant au moins un arc. Par la suite, on nommera ce sous-graphe commun le *linegraph commun*.

Algorithme 4.2 CalculerBGPHC($\mathbb{Q}$) : Φ

Entrée : $\mathbb{Q}$ est l'ensemble des requêtes SPARQL connues
Sortie : Φ est le BGPHC de $\mathbb{Q}$

```
1 : BGP ← ExtraireBGP(Q), L ← CalculerLinegraphs(BGP)
2 : L_c0 ← ∅, L_0 ← L, L_0^C ← ∅
    ▷ Créez l'arbre avec le sommet qui servira de racine. Chaque sommet est un 3-tuples.
3 : Φ ← nouveau Arbre(Tuple3(L_c0, L_0, L_0^C))
4 : v_0 ← Racine(Φ)
5 : C ← ClassificationLinegraph(L_0, v_0)                    ▷ ensemble des clusters
6 : AjouterEnfants(v_0, C)
7 : RecCalculerBGPHC(Φ, v_0)
8 : retourne Φ
9 : procédure RecCalculerBGPHC(Φ, v_p)
    ▷ v_p ∈ V position actuelle dans Φ
10 :    si Enfants(v_p) ≠ ∅ alors
11 :        pour tout v_c ∈ Enfants(v_p) faire
12 :            si |L(v_c)| > 1 alors
13 :                C ← ClassificationLinegraph(L(v_c), v_c)
14 :                AjouterEnfants(v_c, C)
15 :                RecCalculerBGPHC(Φ, v_c)
16 :            fin si
17 :        fin pour
18 :    fin si
19 : fin procédure
```

L'algorithme de classification récursif. L'algorithme 4.3 calcule les clusters. Chaque cluster est calculé à partir du cluster calculé au niveau précédent, noté v_p. Au cours du calcul de chaque cluster, trois structures de données sont construites (pour optimiser le calcul des linegraphs communs). Ils stockent respectivement dans chaque cluster v_i :

- $\mathcal{L}_c(v_i)$, le linegraph commun du cluster,
- $L^{\complement}(v_i)$ le complément de $L(v_i)$ défini par la formule suivante, où $\mathbb{L}$ est l'ensemble des linegraphs indexés dans notre base de connaissances à partir de tous les BGP des requêtes SPARQL connues :

$$L^{\complement}(v_i) = \{\mathcal{L} \in L(v_p) \subseteq \mathbb{L} \mid \mathcal{L} \notin L(v_i)\} \tag{4.4}$$

- $L(v_i)$, l'ensemble des linegraphs partageant au moins un arc avec le linegraph commun $\mathcal{L}_c(v_i)$.

L'algorithme prend $L(v_p)$ en entrée, l'ensemble des linegraphs du cluster calculé au précédent niveau. L'algorithme évalue d'abord le nombre de clusters nécessaires pour obtenir un sous-graphe commun (non vide) pour tous les linegraphs en entrée. Ensuite, il calcule les sous-graphes des arcs communs minimums (non vides) nécessaires afin de maximiser le nombre de linegraphs dans chaque cluster.

Dans l'algorithme de classification, nous utilisons la syntaxe $\mathcal{L}_c(v_0 \to v_p)$ pour représenter l'union des linegraphs communs calculés aux précédents niveaux, du cluster précédent calculé v_p jusqu'à la racine v_0 :

$$\mathcal{L}_c(v_0 \to v_x) = \bigcup_{y=v_0}^{v_x} \mathcal{L}_c(v_y) \tag{4.5}$$

$\mathcal{L}_c(v_0 \to v_p)$ sert à ne plus tenir compte des linegraphs communs déjà détectés dans les niveaux précédents et cette union nous servira également par la suite à calculer les snippets.

Algorithme 4.3 ClassificationLinegraph$(L, v_p) : C$

$C \leftarrow \varnothing$
▷ Calcul du nombre de clusters
pour tout $\mathcal{L} \in L$ **faire**
  $ajoutDansUnCluster \leftarrow FALSE$
  $\Delta\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L}_c(v_0 \rightarrow v_p)$ ▷ Retire les $\mathcal{L}_c$ calculés dans les précédents niveaux
  si $(\Delta\mathcal{L} \neq \varnothing)$ **alors**
    pour tout $\mathcal{C} \in C$ **faire**
      $d_J \leftarrow \mathrm{J}_\delta(\mathcal{L}_c(\mathcal{C}), \Delta\mathcal{L})$
      si $d_J < 1$ **alors**
        $\mathcal{L}'_c \leftarrow \mathcal{L}_c(\mathcal{C}) \cap \Delta\mathcal{L}$
        $L' \leftarrow L(\mathcal{C}) \cup \{\mathcal{L}\}$
        Modifier$(\mathcal{C}, Tuple3(\mathcal{L}'_c, L', L^{\complement}(\mathcal{C})))$
        $ajoutDansUnCluster \leftarrow true$
      sinon
        $L'^{\complement} \leftarrow L^{\complement}(\mathcal{C}) \cup \{\mathcal{L}\}$
        Modifier$(\mathcal{C}, Tuple3(\mathcal{L}_c(\mathcal{C}), L(\mathcal{C}), L'^{\complement}))$
      fin si
    fin pour
    si non $ajoutDansUnCluster$ **alors**
      AjouteNouveauCluster$(C, Tuple3(\Delta\mathcal{L}, \{\mathcal{L}\}, \varnothing))$
    fin si
  fin si
fin pour
▷ Calcul les clusters définitifs
pour tout $\mathcal{L} \in L$ **faire**
  $\Delta\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L}_c(v_0 \rightarrow v_p)$
  si $(\Delta\mathcal{L} \neq \varnothing)$ **alors**
    pour tout $\mathcal{C} \in C$ **faire**
      si $\mathcal{L} \notin (L^{\complement}(\mathcal{C}) \cup L(\mathcal{C}))$ **alors**
        $d_J \leftarrow \mathrm{J}_\delta(\mathcal{L}_c(C), \Delta\mathcal{L})$
        si $d_J < 1$ **alors**
          $\mathcal{L}'_c \leftarrow \mathcal{L}_c(\mathcal{C}) \cap \Delta\mathcal{L}$
          $L' \leftarrow L(\mathcal{C}) \cup \{\mathcal{L}\}$
          Modifier$(\mathcal{C}, Tuple3(\mathcal{L}'_c, L', L^{\complement}(\mathcal{C})))$
        sinon
          $L'^{\complement} \leftarrow L^{\complement}(\mathcal{C}) \cup \{\mathcal{L}\}$
          Modifier$(\mathcal{C}, Tuple3(\mathcal{L}'_c(\mathcal{C}), L'(\mathcal{C}), L'^{\complement}))$
        fin si
      fin si
    fin pour
  fin si
fin pour
retourne C

Illustration. La figure 4.4 présente un exemple de BGPHC. Dans cet exemple, l'ensemble des linegraphs à classer, noté $\mathbb{L}$, est au nombre de trois. On peut voir en (a) le point de départ avec un cluster initial unique $L(v_0)$ qui contient tous les linegraphs connus. En (b), les sous-graphes communs dans un cluster $\mathcal{L}_c(v_i)$ ont été calculés et sont représentés. Dans cet exemple en (c), les derniers clusters ne contiennent qu'un seul linegraph, mais, quand il existe des linegraphs identiques, il est possible qu'il y ait plusieurs linegraphs dans les derniers clusters.

Nous illustrons maintenant l'utilisation du BGPHC généré et la mesure LBGPD pour suggérer des snippets de requêtes que nous appellerons *SPARQLets*.

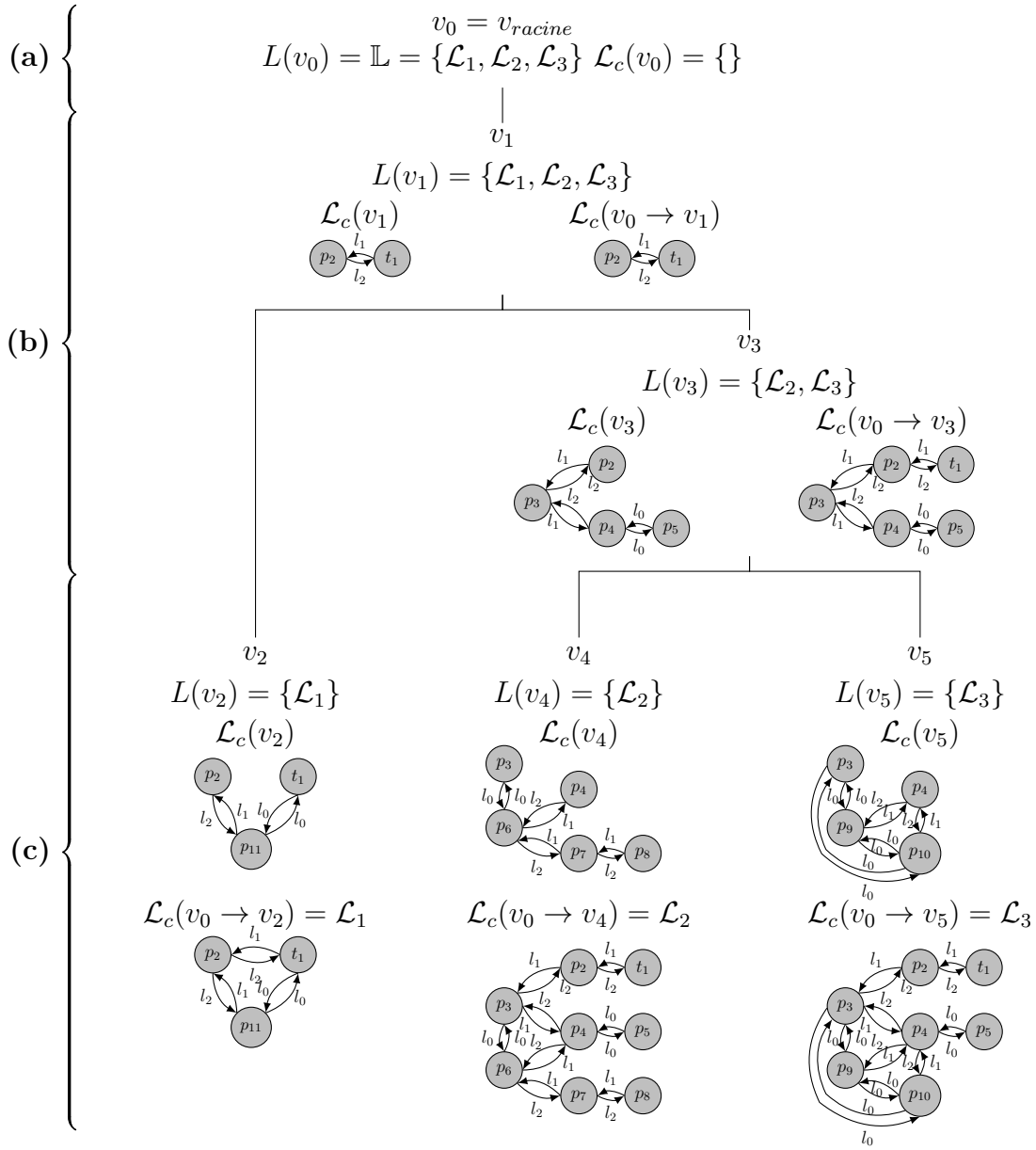


Figure 4.4 – Exemple de classification hiérarchique des linegraphs de BGP (BGPHC) à l'issue du processus **A.3** à partir des BGP BGP_1, BGP_2 et $BGP3$ des figures 4.2 et 4.3.

4.3.4 Le workflow de suggestion de snippets

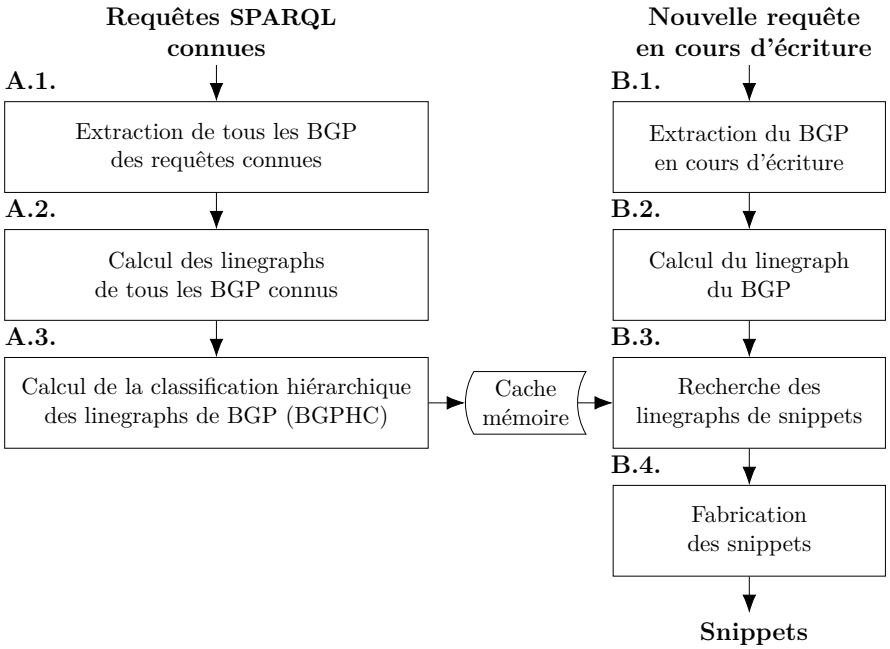


Figure 4.5 – Schéma du workflow de génération des snippets de SPARQLets-Finder où **A.1-3** construit le BGPHC nécessaire à **B.1-4** afin de générer des propositions de snippets pour une requête en cours d'écriture.

Le programme SPARQLets-Finder orchestre une succession d'opérations au sein d'un workflow, décrit dans la figure 4.5, afin d'obtenir des snippets. Nous commencerons par illustrer le processus de construction des snippets, puis nous expliquerons comment ils sont calculés au sein d'un BGPHC.

Dans le processus B.1. de la figure 4.6, la première étape, BGP_0, est extraite de la requête en cours d'écriture. Le BGP est sélectionné en fonction de la position du curseur dans l'éditeur de requêtes. Nous utilisons un parseur syntaxique ANTLR4 pour extraire ce BGP en B.1., mais également à l'étape A.1. pour extraire tous les BGP des requêtes connues de notre base de connaissances. Nous avons dû développer notre propre parseur SPARQL [141] pour pouvoir les extraire même en présence d'erreurs dans la requête soumise (partiellement écrite).

Dans B.2., le linegraph du BGP en entrée est calculé : $\mathcal{L}_{BGP_0} = \mathcal{L}(BGP_0)$.

Dans B.3., l'algorithme 4.4 utilise l'arbre orienté BGPHC pour trouver les linegraphs des futurs snippets. Chaque chemin entre un sommet et la racine constitue un potentiel linegraph afin de fabriquer un nouveau snippet.

Pour sélectionner les chemins dans l'arbre que constitue notre classification hiérarchique de linegraphs, nous utilisons la distance de Jaccard entre $\mathcal{L}_{BGP_0}$ et $\mathcal{L}_c(v_0 \rightarrow v_i)$ comme décrite dans la section précédente. Avec les valeurs des distances entre la racine v_0 de l'arbre et ses sommets v_i, nous pouvons parcourir cet arbre tant que la similarité augmente entre $\mathcal{L}_{BGP_0}$ et $\mathcal{L}_c(v_0 \rightarrow v_i)$. Lorsque la similarité diminue, le parcours dans l'arbre est stoppé et les sommets où ce parcours

s'est arrêté sont collectés (Nous justifierons cette méthode d'arrêt dans la section suivante). À partir de l'ensemble des sommets collectés V_s, l'ensemble des linegraphs L_s est calculé (ligne 17-18 de l'algorithme 4.4).

Nous sélectionnons les cinq linegraphs de snippets qui se rapprochent le plus de $\mathcal{L}_{BGP_0}$ (Ligne 6 de l'algorithme 4.4). Nous limitons arbitrairement leur nombre à cinq afin de réduire le temps de calcul nécessaire pour convertir les linegraphs en snippets dans le processus B.4.

Le processus B.4., illustré dans la figure 4.7, est le dernier processus de notre workflow avant la génération des snippets à l'éditeur de requêtes qui les proposera à l'utilisateur. Pour obtenir ces snippets, nous convertissons les linegraphs des snippets en une liste de BGP qui pourront s'insérer dans BGP_0, c'est-à-dire avec les mêmes noms de variables, sans doublons, etc. A ce stade de l'implémentation, nous obtenons déjà des snippets lisibles par les utilisateurs. Certains post-traitements pourront encore être ajoutés pour améliorer leur lisibilité.

Algorithme 4.4 RechercherSnippets($BGP_0, \Phi, \mathcal{K}$) : S

Entrée : BGP_0 d'une requête, Φ BGPHC calculé par l'algorithme 4.2 et $\mathcal{K}$ la connaissance sur les requêtes SPARQL qui nous servira à construire les noms de variables dans les snippets.
Sortie : S ensemble des snippets suggérés.

▷ $\mathrm{J}_\delta(\mathcal{L}_a, \mathcal{L}_b)$ est la distance de Jaccard entre $\mathcal{L}_a$ et $\mathcal{L}_b$
▷ $d_{Ji} = \mathrm{J}_\delta(\mathcal{L}_c(v_0 \to v_i), \mathcal{L}_{BGP_0})$ et $d_{Ji} \in [0..1]$

1 : $\mathcal{L}_{BGP_0} \leftarrow \mathcal{L}(BGP_0)$
2 : $v_0 \leftarrow$ Racine(Φ) ▷ Nous démarrons à la racine
3 : $L_s \leftarrow \varnothing$ ▷ Pour commencer, L_s est vide
4 : $d_{J0} \leftarrow 1$ ▷ $\mathrm{J}_\delta(\mathcal{L}_c(v_0 \to v_0), \mathcal{L}_{BGP_0}) = \mathrm{J}_\delta(\varnothing, \mathcal{L}_{BGP_0}) = 1$
5 : RecSSL($\mathcal{L}_{BGP_0}, L_s, \Phi, v_0, d_{J0}$)
6 : $\widehat{L_s} \leftarrow \underset{\mathcal{L}_s \in L'_s \subseteq L_s, |L'_s| \leq 5}{\operatorname{argmin}} \operatorname{sort}(J_\delta(\mathcal{L}_s, \mathcal{L}_{BGP_0}))$
7 : $S \leftarrow FabriquerSnippet(\widehat{L_s}, BGP_0, \mathcal{K})$ ▷ Conversion de $\mathcal{L}$ en BGP
8 : **retourne** S
9 : **procédure** RecSSL($\mathcal{L}_{BGP_0}, L_s, \Phi, v_p, d_J$)
10 : $V_{enfants} \leftarrow$ Enfants(Φ, v_p)
11 : **si** $V_{enfants} \neq \varnothing$ **alors**
12 : **pour tout** $v_i \in V_{enfants}$ **faire**
13 : $d_{Ji} \leftarrow \mathrm{J}_\delta(\mathcal{L}_c(v_0 \to v_i), \mathcal{L}_{BGP_0})$
14 : **si** $d_{Ji} \leq d_J$ **alors**
15 : RecSSL($\mathcal{L}_{BGP_0}, L_s, \Phi, v_i, d_{Ji}$)
16 : **sinon si** $d_{Ji} > d_J$ **alors**
17 : $\mathcal{L}_{snippet} \leftarrow \mathcal{L}_c(v_0 \to v_i) \setminus \mathcal{L}_{BGP_0}$
18 : $L_s \leftarrow L_s \cup \{Tuple2(d_{Ji}, \mathcal{L}_{snippet})\}$
19 : **fin si**
20 : **fin pour**
21 : **sinon**
22 : **si** $d_J < 1$ **et** $d_J \neq 0$ **alors**
23 : $\mathcal{L}_{snippet} \leftarrow \mathcal{L}_c(v_0 \to v_p) \setminus \mathcal{L}_{BGP_0}$
24 : $L_s \leftarrow L_s \cup \{Tuple2(d_J, \mathcal{L}_{snippet})\}$
25 : **fin si**
26 : **fin si**
27 : **fin procédure**

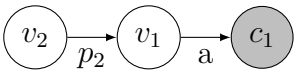
En entrée, une requête SPARQL, avec en rouge la position du curseur à proximité du BGP en cours d'écriture par un utilisateur.

```
SELECT *
WHERE {
   ?disease wdt:P31 wd:Q12136 .
   ?drug wdt:P2175 ?disease .
}
```

B.1. Extraction du BGP en cours d'écriture : BGP_0

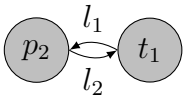
```
?disease wdt:P31 wd:Q12136 .
?drug wdt:P2175 ?disease .
```

v_1 a c_1
v_2 p_2 v_{v3}



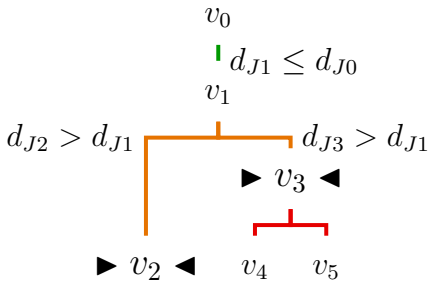
B.2. Calcul du linegraph

$\mathcal{L}_{BGP_0} = \mathcal{L}(BGP_0)$



B.3. Recherche des linegraphs des snippets adaptés à BGP_0 à l'aide du BGPHC de la figure 4.4.

$d_{Ji} = Jaccard_\delta(\mathcal{L}_c(v_0 \to v_i), \mathcal{L}_{BGP_0})$



Dès que la distance augmente, on ne va pas plus loin pour collecter les sommets. Ici, $v2$ et $v3$.

En sortie, les linegraphs de snippets trouvés BGP_0

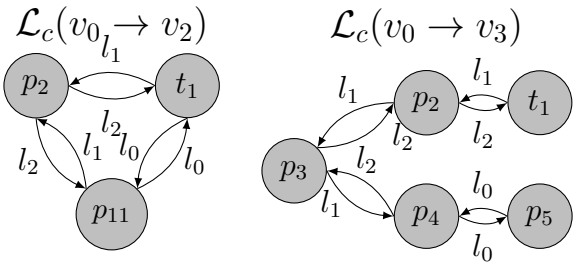
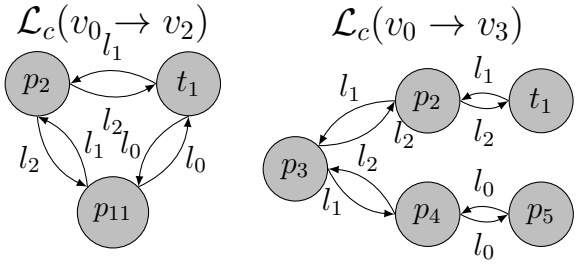


Figure 4.6 – Illustration des processus **B.1-3.** qui servent à calculer les linegraphs des snippets en fonction du BGPHC dans la figure 4.4 et le BGP extrait d'une nouvelle requête.

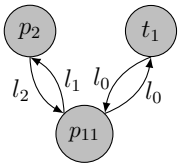
En entrée, les linegraphs des snippets obtenus au travers du processus **B.1-3.** avec l'exemple de requête dans la figure 4.6



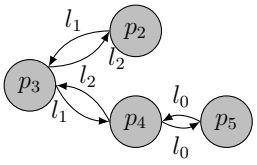
B.4. Fabrication des snippets

1re étape
Soustraction
du linegraph $\mathcal{L}_{BGP_0}$

$$\mathcal{L}_{snippet1} = \mathcal{L}_c(v_0 \rightarrow v_2) \setminus \mathcal{L}_{BGP_0}$$



$$\mathcal{L}_{snippet2} = \mathcal{L}_c(v_0 \rightarrow v_3) \setminus \mathcal{L}_{BGP_0}$$



2e étape
Conversion en BGP

$BGP_{snippet1}$	$BGP_{snippet2}$
v_1 a c_1	v_1 p_2 v_2
v_1 p_{11} v_2	v_3 p_3 v_1
v_3 p_2 v_1	v_4 p_4 v_3
	v_4 p_5 v_5

3e étape
Mise en forme

Liste des techniques utilisées : retrait des propriétés en doublon, changement des noms de variables, réorganisation des patrons de triplets, etc. (toujours en cours d'amélioration)

$Snippet_1$
```
?disease wdt:P18 ?image .
```
$Snippet_2$
```
?drug wdt:P2175 ?medicalConditionTreated .
?var4 p:P3354 ?positiveTherapeuticPredictorv1 .
?var4 wdt:P3433 ?biologicalVariantOf .
?positiveTherapeuticPredictorv1 ps:P3354 ?drug .
```

En sortie, si l'utilisateur sélectionne le $Snippet_2$, il est inséré après BGP_0 dans la requête, aboutissant à BGP_{res}. L'utilisateur n'a plus qu'à finaliser sa requête :

```
?disease wdt:P31 wd:Q12136 .
?drug wdt:P2175 ?disease .
~~?drug wdt:P2175 ?medicalConditionTreated.~~
~~?var4~~ ?variant p:P3354 ?positiveTherapeuticPredictorv1.
~~?var4~~ ?variant wdt:P3433 ?biologicalVariantOf .
?positiveTherapeuticPredictorv1 ps:P3354 ?drug .
```

Figure 4.7 – Illustration du processus **B.4.** qui fabrique les snippets à partir de leurs linegraphs calculés dans la figure 4.6.

Pour terminer l'explication de notre approche dans la section suivante, nous discuterons de la méthode utilisée pour parcourir l'arbre de classification afin de sélectionner des snippets.

4.3.5 Discussion à propos du parcours dans BGPHC

Dans l'algorithme 4.4, le choix d'arrêter de parcourir l'arbre BGPHC à partir du moment où la distance avec $\mathcal{L}_{BGP_0}$ commence à diverger est le fruit du raisonnement suivant.

Objectif d'un snippet. L'arbre étant construit de manière non supervisée, chaque niveau de profondeur sur un parcours est déterminé par des triplets partagés entre plusieurs BGP. Chaque grappe de ces triplets correspond donc au résultat d'un besoin précis reproduit plusieurs fois par un ou plusieurs utilisateurs. Il n'est pas certain qu'il soit possible de diviser une grappe de triplets sans altérer le besoin initial qui a poussé des utilisateurs à l'écrire. Un snippet est idéalement constitué d'une seule grappe à la fois, afin d'offrir à l'utilisateur un snippet correspondant à un seul besoin précis, pour en simplifier la réutilisation.

Détecter une nouvelle grappe de triplets. La difficulté est ainsi de déterminer à quelle profondeur de l'arbre nous devons chercher cette grappe de triplets, en sachant que la meilleure profondeur peut différer d'une branche à une autre. Le fait de fixer des seuils dans un parcours ne semble pas une solution souhaitable, car arbitraire. Par contre, l'évolution de la distance de similarité durant le parcours est différente d'une branche à une autre et donne un indicateur fiable concernant la détection d'une grappe de triplets qui ne figure pas encore dans BGP_0.

Sélectionner les grappes de triplets les plus intéressantes. Nous souhaitons voir émerger des snippets « inattendus », c'est-à-dire des grappes de triplets que l'utilisateur n'a pas encore commencé à écrire, bien qu'il ait de fortes chances de devoir le faire à un moment ou à un autre. Cela signifie que nous recherchons les grappes de triplets non présentes dans BGP_0 qui se trouvent à proximité immédiate des linegraphs les plus similaires à $\mathcal{L}_{BGP_0}$ dans BGPHC. Nous utilisons donc l'évolution de la distance de similarité dans un parcours pour obtenir les sommets des linegraphs les plus proches de $\mathcal{L}_{BGP_0}$ afin de collecter leurs fils, à condition qu'ils soient plus différents que leur père de BGP_0. A l'aide de ces grappes de triplets inconnues de l'utilisateur, nous construisons ensuite les snippets adaptés à BGP_0.

Optimisation de la méthode. L'algorithme récursif 4.4 optimise cette méthode en collectant les données nécessaires au calcul des snippets en détectant simplement le moment où la distance de similarité de $\mathcal{L}_{BGP_0}$ diverge au sein de BGPHC.

Nous ne savons pas si ce parcours dans BGPHC est la solution optimale, mais cette première implémentation était suffisamment prometteuse pour effectuer une évaluation globale de cette approche avec de vrais utilisateurs au sein de la plateforme LinkedWiki.

Les perspectives d'amélioration de cette autocomplétion devront inclure l'étude d'autres algorithmes pour exploiter au mieux les possibilités qu'offre BGPHC.

La section suivante présente l'évaluation globale de cette première approche.

4.4 Évaluation de la pertinence des snippets

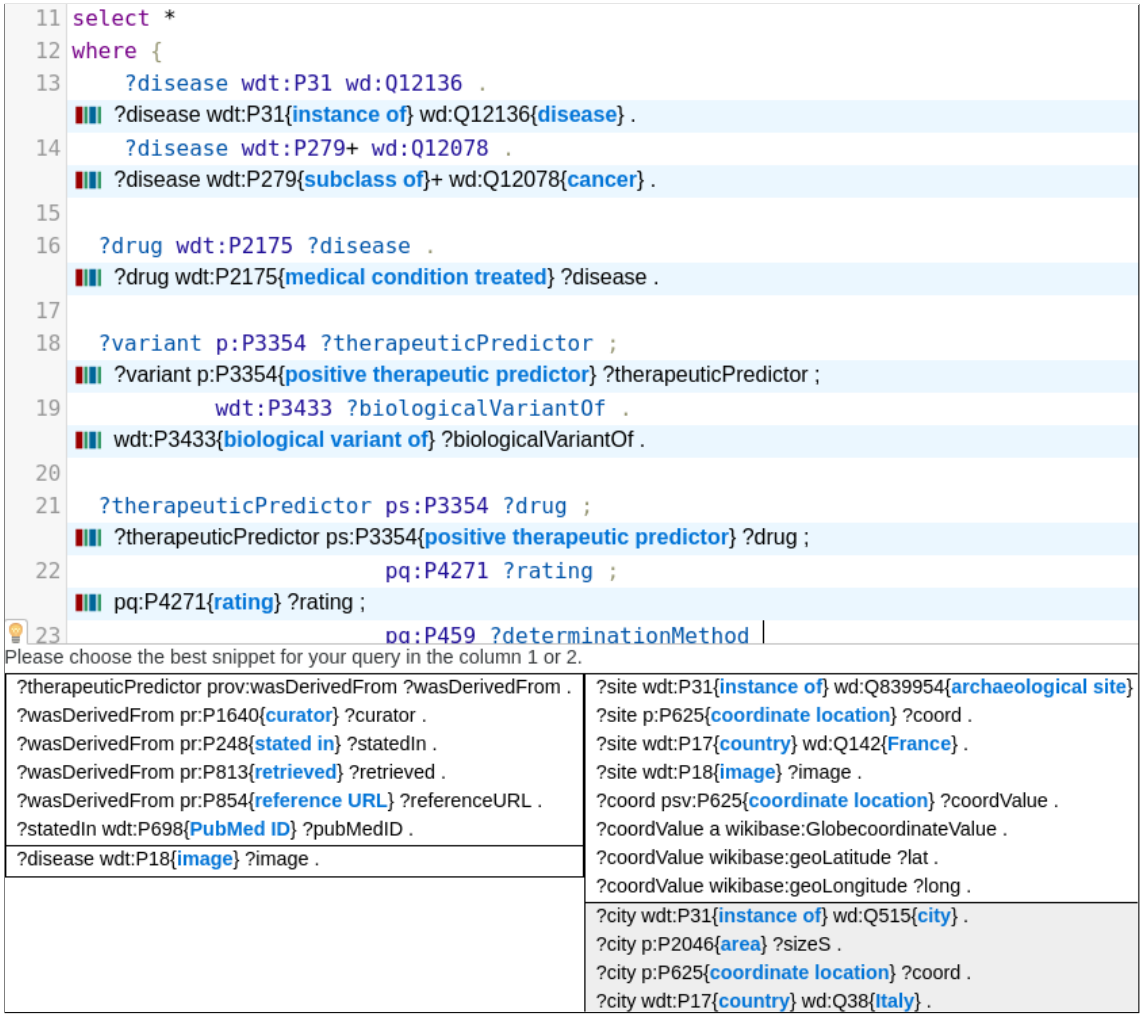


Figure 4.8 – On évalue quantitativement SPARQLets-Finder à l'aide d'utilisateurs écrivant des requêtes qui peuvent choisir entre des snippets de deux approches différentes positionnées aléatoirement à gauche ou à droite : à gauche dans cette figure, SPARQLets-Finder, et à droite, une « Recherche en texte intégral » (lien Web).

Dans cette section, nous évaluons notre approche quantitativement, puis qualitativement.

Au moment de l'évaluation (juin 2018), la base de connaissances comprenait 1 400 BGP extraits de 580 requêtes SPARQL écrites par des utilisateurs réels, dont environ 100 qui traitaient de questions sur des données biologiques ou de biomédecine. L'évaluation quantitative fournira une mesure approximative de l'efficacité de l'approche. L'évaluation qualitative démontrera les bénéfices de notre approche pour les utilisateurs.

4.4.1 Évaluation quantitative

Cette section présente une évaluation quantitative. Nous allons comparer notre approche, à savoir SPARQLets-Finder, qui utilise un BGPHC, à une approche alternative qui utilise des techniques déjà fournies par un certain nombre de systèmes de base de données, notamment dans les bases de données relationnelles (SQL). Nous nommons cette approche alternative « recherche en texte intégral » (en anglais, *full-text search*).

Approche par recherche en texte intégral. Pour implémenter cette approche, nous avons dupliqué le workflow d'origine dans la figure 4.5, puis remplacé les étapes B.2. et B.3. par l'algorithme 4.5.

Au sein de cet algorithme, la requête SQL avec la fonction `MATCH AGAINST` cherche les IRI absolus présents dans BGP_0 au sein des BGP des requêtes SPARQL connues comme des mot-clés au sein de textes. Quand cette première requête SQL ne trouve pas de réponses, on utilise alors la fonction `LIKE`.

La requête avec `MATCH AGAINST` exécutée au sein de MariaDB [81] n'a pas de réponse quand les IRI recherchés sont trop présents dans les textes des BGP indexés par la base de données. Cette optimisation compréhensible pour les textes en langage naturel est une limitation pour les requêtes SPARQL, que nous contournons avec une seconde requête SQL et la fonction `LIKE`.

Algorithme 4.5 Recherche en texte intégral$(BGP_0, \mathcal{K}) : S$

Entrée : BGP_0 d'une requête et $\mathcal{K}$ la connaissance sur les requêtes SPARQL
Sortie : S ensemble des snippets suggérés.

1 : $\mathbb{BGP} \leftarrow$ ExtraireBGP$(\mathcal{K})$
2 : $IRI \leftarrow$ ExtraireIRI(BGP_0)
3 : $\widehat{BGP_s} \leftarrow$ $\mathbb{BGP}$.executeSQLQueryWithMariaDb(

```
SELECT id FROM BGP WHERE
MATCH(BGP_text)
    AGAINST(
        " IRI0 IRI1 ...IRIn "
        WITH QUERY EXPANSION
    )
LIMIT 10
)
```

4 : **si** $|\widehat{BGP_s}| = 0$ **alors**
5 : $\widehat{BGP_s} \leftarrow$ $\mathbb{BGP}$.executeSQLQueryWithMariaDb(

```
SELECT id FROM BGP WHERE
   BGP_text LIKE "%IRI0%" AND
   BGP_text LIKE "%IRI1%" AND
   ...
   BGP_text LIKE "%IRIn%"
LIMIT 10
)
```

6 : **fin si**
7 : $S \leftarrow FabriquerSnippet'(\widehat{BGP_s}, BGP_0, \mathcal{K})$
8 : **retourne** S

Les deux approches sont déployées dans l'éditeur SPARQL pour faire comparer leurs snippets respectifs directement auprès des utilisateurs. La figure 4.8 illustre la suggestion d'extraits qui apparaît lorsqu'un utilisateur clique sur l'icône « ampoule » dans la marge gauche de l'éditeur. Cette icône apparaît durant l'écriture d'un BGP par l'utilisateur quand l'une des deux approches a des snippets à suggérer. Chaque colonne de la fenêtre contextuelle contient des extraits fournis par l'une des deux approches. Pour éviter les biais, la colonne d'affichage des snippets de SPARQLets-Finder est choisie aléatoirement.

Tests avec de vrais utilisateurs. Au moment de l'évaluation (12/09/2018) avec de vrais utilisateurs (environ 134 sélections de snippets), nous avons pu mesurer que les utilisateurs sélectionnaient beaucoup plus souvent SPARQLets-Finder que la « recherche en texte intégral » (voir Figure 4.9).

Tests automatiques. Pour confirmer cette tendance, nous avons généré des tests automatiques à partir des BGP contenus dans la base de connaissances. Chaque test :

— attend en sortie comme résultat BGP_{res}, qui est le BGP d'origine dans la base de connaissance,

— prend en entrée BGP_0 qui est obtenu en supprimant un ou plusieurs patrons de triplets à la fin du BGP_{res}.

Pour comparer les snippets proposés par les deux approches, nous recherchons le snippet le plus proche de BGP_{res}. Pour ce faire, nous utilisons la mesure de similarité de Levenshtein, parce qu'elle semble le mieux capturer le résultat attendu par les utilisateurs.

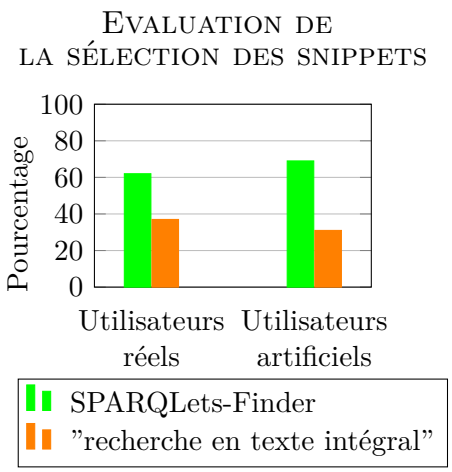


Figure 4.9 – Comparaison entre SPARQLets-Finder et une "recherche en texte intégral".

A l'issue de ces tests, SPARQLets-Finder s'est révélé être meilleur dans la grande majorité des cas. Il y a cependant une exception : lorsqu'en entrée BGP_0 ne contient qu'un seul patron de triplets les deux approches réagissent mal. Nous avons l'intention d'améliorer SPARQLets-Finder pour surmonter cette difficulté.

Les résultats sont présentés dans la figure 4.9 pour environ 1700 tests prenant en entrée des BGP d'au moins deux patrons de triplets. Les résultats de nos tests automatiques confirment les résultats obtenus à plus petite échelle avec des utilisateurs réels. Cette évaluation quantitative inclut la prise en compte simultanée de plusieurs utilisateurs et valide le passage à l'échelle de notre approche.

Dans la section suivante, nous démontrerons la capacité de SPARQLets-Finder à aider réellement les scientifiques à concevoir leurs propres requêtes.

4.4.2 Évaluation qualitative auprès de scientifiques

Pour évaluer qualitativement SPARQLets-Finder, nous revenons à l'exemple utilisé au début de ce chapitre, que nous rappelons dans la figure 4.10, et nous restituons les réactions de cliniciens utilisant notre système.

Avec seulement une connaissance limitée de l'organisation des données, des cliniciens sont maintenant en mesure d'interroger Wikidata car la difficulté d'écrire une requête est réduite grâce aux différentes autocomplétions implémentées.

Ils apprécient le support dans la gestion des préfixes (lignes 1 à 9 sur la figure 4.10) et l'accès à des modèles de sous-requêtes pour faire appel au service d'étiquettes Wikidata (lignes 40 à 42).

La possibilité de rechercher les IRI des propriétés et des instances en utilisant des mots-clés dans sa langue naturelle simplifie également leurs tâches dans la construction de requêtes : par exemple, pour l'IRI qui identifie le concept de cancer (lignes 19), ainsi que l'IRI de la propriété *maladie traitée* (ligne 22).

Enfin, les snippets s'avèrent très utiles dans la construction des deux parties les plus complexes de la requête : la recherche de prédicateurs thérapeutiques positifs pour la réponse (lignes 25 à 33) avec leurs références dans la littérature biomédicale (lignes 36 à 38, la figure 4.8 contient dans la colonne de gauche le snippet qui a permis à l'utilisateur d'écrire cette partie).
Dans les deux cas, les snippets suggèrent des structures déjà construites et aident ainsi les cliniciens à construire leurs propres requêtes, même si la proposition ne répond pas immédiatement à leurs besoins.

La figure 4.8 fournit une capture de l'interface que voit l'utilisateur lorsqu'il reçoit deux snippets de SPARQLets-Finder (à gauche) et d'autres snippets produits par la « recherche en texte intégral ». Les snippets fournis par SPARQLets-Finder sont particulièrement pertinents pour l'utilisateur.

Pour obtenir des snippets avec un même niveau de pertinence dans d'autres domaines, les scientifiques n'ont besoin que de commencer à partager leurs requêtes SPARQL et les snippets s'amélioreront automatiquement. Dans l'expérience présentée ici, seulement 100 requêtes ont suffi pour entraîner SPARQLets-Finder. L'effort reste donc raisonnable.

```
PREFIX bd: <http://www.bigdata.com/rdf#>                                    Français
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
PREFIX pr: <http://www.wikidata.org/prop/reference/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX ps: <http://www.wikidata.org/prop/statement/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>

SELECT DISTINCT
?diseaseLabel ?drugLabel ?variantLabel  ?geneLabel
?determinationMethodLabel ?ratingLabel
(CONCAT("https://www.ncbi.nlm.nih.gov/pubmed/?term=",?pubMedID)
 as ?ArticlePubMed)
WHERE {
  # Find all the disease that are cancers
  ?disease wdt:P31 wd:Q12136 ;
 ?disease wdt:P31{nature de l'élément} wd:Q12136{maladie} ;
           wdt:P279+ wd:Q12078 .
 wdt:P279{sous-classe de}+ wd:Q12078{cancer} .

  # Find the drugs treating cancer
  ?drug wdt:P2175 ?disease .
 ?drug wdt:P2175{maladie traitée} ?disease .

  # Find the genetic variants associated with a better response to the treatment
  ?therapeuticPredictor ps:P3354 ?drug ;
 ?therapeuticPredictor ps:P3354{prédicteur thérapeutique positif} ?drug ;
         # Find metadata regarding the positive prediction response
                       pq:P4271 ?rating ;
 pq:P4271{classification} ?rating ;
                       pq:P459 ?determinationMethod .
 pq:P459{méthode de détermination} ?determinationMethod .

  # Find the variant associated with the response predictor
  # Find the associated gene
  ?variant p:P3354 ?therapeuticPredictor ;
 ?variant p:P3354{prédicteur thérapeutique positif} ?therapeuticPredictor ;
           wdt:P3433 ?gene .
 wdt:P3433{variante biologique de} ?gene .

  # Find the provenance of the predictor
  ?therapeuticPredictor prov:wasDerivedFrom ?wasDerivedFrom .
  ?wasDerivedFrom pr:P248 ?statedIn .
 ?wasDerivedFrom pr:P248{affirmé dans} ?statedIn .
  ?statedIn wdt:P698 ?pubMedID  .
 ?statedIn wdt:P698{identifiant PubMed} ?pubMedID .

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en,fr" .
  }
}
limit 10
```

Figure 4.10 – La requête de la figure 4.1 avec les commentaires du scientifique, plus les sucres syntaxiques de SPARQL pour alléger son écriture, ainsi que les annotations automatiques de l'éditeur pour traduire les identifiants de Wikidata (lien Web).

4.5 Conclusion

Dans ce chapitre, nous avons décrit le problème consistant à aider les scientifiques dans la conception de leurs requêtes SPARQL afin de leur permettre d'exploiter la richesse des données au sein du Linked Open Data. En particulier, nous nous sommes concentrés sur l'identification des besoins des utilisateurs en matière de fonctionnalités d'autocomplétion et nous avons proposé des solutions à ces besoins.

Plus précisément, les contributions présentées dans ce chapitre sont doubles. Tout d'abord, pour répondre aux besoins exprimés par les utilisateurs dans l'étude du chapitre 3 (section 3.2), nous avons introduit la première approche d'autocomplétion (à notre connaissance) basée sur des *snippets*, capable de fournir aux utilisateurs des complétions de leurs requêtes basées sur des requêtes précédentes similaires. Deuxièmement, nous avons évalué notre approche à la fois quantitativement, en fournissant des mesures obtenues sur un grand nombre de cas d'utilisation, et qualitativement, sur la base de requêtes biomédicales. Nous avons démontré que les snippets fournis étaient particulièrement pertinents pour l'utilisateur pour concevoir une nouvelle requête et que l'effort requis pour apprendre à utiliser notre solution restait raisonnable.

De futurs travaux sont planifiés dans plusieurs directions.

Nous voulons d'abord améliorer notre algorithme : améliorer la qualité des résultats, et rendre l'algorithme plus performant (en temps d'exécution). Pour ce faire, nous comptons créer une plateforme permettant d'évaluer la qualité des résultats de SPARQLets-Finder dans différentes versions de son code, ainsi que les temps d'exécution.

Ensuite, nous voulons permettre à SPARQLets-Finder de traiter de très petites requêtes, c'est-à-dire avec des BGP contenant qu'un seul patron de triplets. Enfin, nous avons l'intention d'étendre ce travail pour mieux gérer les requêtes fédérées. Cela implique de considérer en même temps plusieurs BGP de différents services SPARQL et, par conséquent, les relations entre les patrons de graphes de différents services au sein d'une même requête. Une fois cette étape réalisée, nous serons en mesure d'affiner la conception des snippets en nous focalisant sur les schémas RDF utilisés par les différents services.

Résumé du chapitre 4

La conception d'une requête SPARQL peut être une tâche fastidieuse, même pour les utilisateurs expérimentés. Ceci est souvent dû à une connaissance imparfaite des schémas RDF impliqués dans la requête. Pour résoudre ce problème, un nombre croissant d'éditeurs de requêtes offrent des fonctionnalités d'autocomplétion. Malheureusement, ces fonctionnalités sont limitées et principalement axées sur la vérification syntaxique.
Pour surmonter ce problème, notre contribution dans ce chapitre est double. Dans un premier temps, nous introduisons le premier algorithme d'autocomplétion, à notre connaissance, capable de suggérer des snippets (fragments de la requête SPARQL) en fonction des requêtes précédentes écrites par des utilisateurs, enrichissant ainsi l'expérience de l'utilisateur.
Enfin et surtout, nous démontrons l'intérêt de notre algorithme pour de véritables requêtes biomédicales avec le service SPARQL de Wikidata, une base de connaissances collaborative.

Résultats publiés du chapitre 4

[117] Article publié : « Designing scientific SPARQL queries using autocompletion by snippets ». Karima Rafes, Serge Abiteboul, Sarah Cohen-Boulakia et Bastien Rance. Dans : eScience, 2018 IEEE 14th International Conference on. IEEE. 2018.

Chapitre 5

SPARQL Score : indicateur d'interopérabilité

Sommaire

5.1 Introduction

L'interopérabilité dans le Linked Data est la capacité des machines compatibles avec le Linked Data à partager n'importe quelle donnée RDF.

La recommandation SPARQL 1.1 du W3C a défini des tests pour vérifier la conformité des services SPARQL et de leurs clients. La plupart des éditeurs de ces services déclarent proposer des services conformes à l'ensemble des dernières recommandations, mais le rapport d'implémentation officiel de SPARQL 1.1 [157] montre qu'aucun d'entre eux ne réussit l'ensemble des tests officiels.

Il existe des références de bancs de tests concernant la *performance*, par exemple le banc de tests de Berlin SPARQL [21], et le *support aux ontologies*, par exemple [92] (même si on peut argumenter que ce banc n'est pas représentatif des applications du

monde réel [161]). De manière assez surprenante, il semble qu'il n'existe aucune évaluation indépendante, comparative et exhaustive des tests du W3C pour déterminer l'*interopérabilité* des services SPARQL des bases de données RDF.

De ce fait, il est impossible de prévoir si une fonctionnalité SPARQL particulière dans une base de données RDF est supportée ou non avant son déploiement. Ceci nuit à l'adoption du Linked Data par les écosystèmes de la recherche scientifique.

Pour la plateforme LinkedWiki, décrite dans le chapitre 3, nous avons développé un indicateur transparent et reproductible des services SPARQL au sein des bases de données RDF afin d'aider les chercheurs à choisir la meilleure solution pour partager leurs données. Notre panel d'utilisateurs est large : des communautés vastes et bien organisées comme dans la physique des hautes énergies (CERN), ainsi que des communautés locales qui découvrent le besoin de partager au-delà des expériences de courte durée, et bien d'autres encore ; cela inclut des communautés de sciences dures et de sciences humaines et sociales.

Dans les travaux décrits dans ce chapitre, nous introduisons un cadre de tests, TFT (Tests For Triplestores) ainsi qu'un indicateur d'interopérabilité associé pour s'intégrer au sein du processus d'intégration continue des éditeurs de logiciels. Cela signifie que les éditeurs ou des utilisateurs peuvent utiliser TFT afin de vérifier l'interopérabilité de leur protocole et du langage de requête après chaque modification de leurs logiciels pour éviter des régressions.

Lorsque les résultats de TFT sont publics, ils sont accessibles au travers d'un service SPARQL et de sites Web.

La plateforme LinkedWiki utilise les résultats publics de ces tests comme un indicateur d'interopérabilité pour décrire la conformité des images virtuelles, contenant une base de données RDF, avant leur déploiement dans le cloud de l'Université Paris-Saclay. Cet indicateur est visible dans la figure 3.4 et il est associé à un lien permettant d'accéder aux détails de ce résultat. Les chercheurs disposent ainsi de plus d'informations pour choisir une base de données RDF en fonction de leurs besoins (par exemple le support : à l'ontologie RDFS, aux requêtes fédérées, aux parcours XPath dans le graphe RDF, etc.). Si les tests actuels sont insuffisants pour recouvrir les besoins d'un chercheur, il suffit d'en créer de nouveaux et de les partager.

La suite du chapitre décrit plus précisément le problème de l'interopérabilité des services SPARQL (section 5.2), nos contributions (en 5.3), les conditions à respecter pour passer ces tests (en 5.4) et les fonctionnalités disponibles (en 5.5). Nous décrivons comment quiconque peut utiliser TFT pour concevoir son banc de tests sans avoir besoin d'aucune ressource matérielle (en 5.6), puis son implémentation (en 5.7) et son évaluation par les éditeurs de service SPARQL (en 5.8). Nous concluons ce chapitre sur le potentiel impact de nos contributions dans la mise en œuvre du Linked Data au sein des universités (en 5.9).

5.2 L'interopérabilité

Nos travaux répondent au problème du manque d'interopérabilité. Après avoir décrit ce que l'on entend par le terme d'interopérabilité, nous décrivons les causes

de ce problème « technique » qui freine la recherche informatique.

Le problème : comment utiliser toutes les informations du Web ? Le Web sémantique, dont le Linked Data fait partie, vise entre autres à partager des informations lisibles entre humains et machines. L'énorme quantité d'information présente est déjà inutilisable par les humains sans l'aide de machines. Et la majorité des machines d'aujourd'hui sont incapables de réutiliser cette masse d'information.

Simplifier la conception des API n'est pas la solution. Les machines sur le Web se spécialisent pour être plus efficaces : robot d'indexation (web crawler), calculateur, analyseur sémantique, bases de données, etc. Certaines machines ne font que stocker et d'autres ne font que consommer.

La communication entre ces machines devient vitale. Les premières réponses à ce besoin d'*API* (*Application Programming Interface*) se nomment CORBA (en anglais, *Common Object Request Broker Architecture*) [135], *WebService* avec SOAP (en anglais, *Simple Object Access Protocol*) [47], REST (en anglais, *Representational State Transfer*) [57], etc. Ces technologies facilitent la création d'API avec un protocole unique.

Malheureusement, la conséquence est la multiplication des API. Les API sont simples pour les développeurs, mais elles rendent impossible la mise en place d'agents autonomes capables de découvrir et de consommer automatiquement les données du Web.

Le Linked Data, une nouvelle API ? La mise en œuvre d'agents capables d'utiliser toutes les données du Web, sans qu'ils aient besoin de télécharger et transformer les données préalablement, est l'objectif du Linked Data avec le protocole SPARQL. C'est l'un des problèmes majeurs pour le *Web of Things*, où chaque objet devient un agent Web potentiel. L'API qu'offre SPARQL diffère des autres API car elle reste fixe quelles que soient les données et elle n'attend au minimum qu'un paramètre : la requête avec le langage SPARQL où l'agent a décrit la sélection des données dont il a besoin.

Le protocole et le langage du Linked Data avec le protocole SPARQL sont connus par avance par les agents, c'est-à-dire que les interfaces sont intégralement connues pour permettre leurs conceptions. On dit ainsi que les machines au sein du Linked Data sont interopérables dès lors qu'elles respectent le protocole et langage SPARQL.

Ce devoir d'interopérabilité est au cœur du Linked Data, car le problème à résoudre est bien de permettre aux machines qui hébergent les données de les partager avec les machines qui vont se spécialiser pour aider les humains à utiliser toutes les informations qui s'accumulent au sein du Web.

Avec le Linked Data, le problème de l'interopérabilité entre les machines est-il résolu ? Sur le papier, la recommandation SPARQL 1.1 du W3C offre une description précise pour implémenter le Linked Data. Dans les faits, même les éditeurs qui déclarent supporter cette recommandation ne réussissent pas tous les tests officiels [157] et donc ne supportent que partiellement cette API. L'interopérabilité au sein du Linked Data n'est donc que partiellement atteinte. Tant que ce problème persistera, l'API proposée par le Linked Data ne pourra pas remplacer les précédentes API car elle n'offre pour le moment qu'une autre solution, et non pas une

solution globale et définitive à l'échelle du Web.

Complications pour l'adoption du Linked Data dans la recherche. Ce manque d'interopérabilité au sein du Linked Data entraîne deux complications critiques pour son adoption généralisée par de grandes communautés scientifiques organisées, telles que la communauté de la physique des hautes énergies (HEP) : (1) la main-d'œuvre expérimentée concernant le Linked Data est rare parce que le Linked Data est encore peu utilisé, et (2) à cause des données de la recherche qui s'accumulent avec le temps, la migration des bases de données actuelles dans le Linked Data devient de plus en plus complexe.

Le Linked Data a encore un long chemin à parcourir pour résoudre le problème de l'interopérabilité des données sur le Web. Ce problème pénalise les infrastructures scientifiques qui ne sont pas en mesure de déployer ces technologies et, donc, les scientifiques qui ne peuvent toujours pas utiliser des agents Web pour les aider à naviguer dans ce déluge d'informations.

Dans la section suivante, nous évoquerons nos contributions pour la résolution de ce problème.

5.3 Nos contributions

Dans cette section, nous parlerons de nos trois contributions : le moyen de détecter les problèmes d'interopérabilité, le moyen de diffuser ces informations auprès des utilisateurs, et la manière de contourner les problèmes de protocole en réalignant les API des services SPARQL.

5.3.1 Une méthode pour tester les services SPARQL

Le framework TFT que nous avons implémenté peut s'intégrer dans le processus d'intégration continue des développeurs de bases de données RDF afin de les aider à améliorer leurs logiciels.

De plus, nous proposons également une méthode pour créer gratuitement un banc de tests en ligne sur n'importe quelle base de données. Les chercheurs et les directions de systèmes d'information (DSI) peuvent ainsi tester gratuitement l'interopérabilité de leurs bases de données avant de les utiliser.

Les résultats de ces bancs de tests en ligne sont fiables car complètement transparents et reproductibles. Certains de ces résultats (avec l'accord des éditeurs) peuvent être rendus publics et permettent de mettre en lumière les problèmes qui existent encore dans de nombreux services SPARQL, ce qui peut encourager les éditeurs à améliorer leurs produits.

Dans la section suivante, nous expliquons comment influencer également les utilisateurs avec les résultats de ces tests.

5.3.2 Un indicateur d'interopérabilité des services SPARQL

En synthétisant les résultats aux tests en un seul nombre, nous offrons un indicateur lisible pour les utilisateurs, que nous allons expliquer dans cette section.

Les communautés les plus avancées veulent utiliser la dernière technologie au sein des bases de données (inférence, vélocité, clustering, etc.). Ces innovations sont rarement disponibles dans les versions stables des bases de données avant plusieurs mois, voire plusieurs années. Les versions instables sont souvent disponibles en téléchargement gratuit et les chercheurs peuvent installer ces dernières versions très rapidement. De plus, pour les petites communautés, le compromis entre respect des normes et performances de pointe est souvent arbitré plus ou moins aveuglément en faveur de ces dernières. Un indicateur d'interopérabilité fournit une solution simple pour prendre une décision éclairée concernant l'interopérabilité tout en sensibilisant le chercheur à ce problème.

La plateforme LinkedWiki qui simplifie le déploiement d'environnements virtuels de recherche (VRE) informe les scientifiques de la compatibilité avec le Linked Data de ces VRE à l'aide de notre indicateur d'interopérabilité.

Tout ceci est mis en œuvre en gardant à l'esprit qu'il ne faut pas sacrifier l'agilité du chercheur dans ses choix technologiques, ce qui est essentiel pour mener des recherches de pointe.

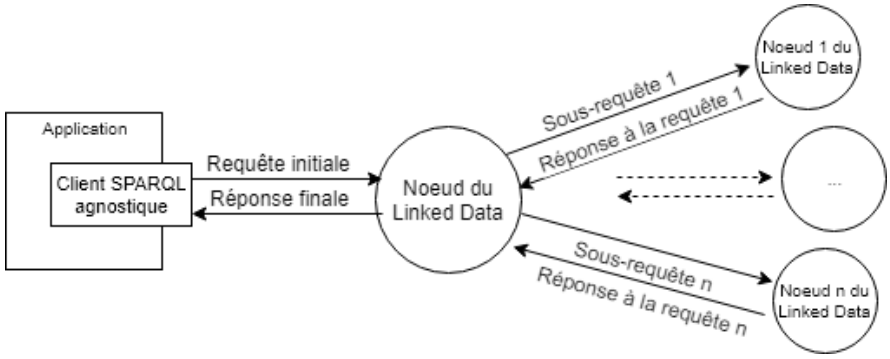
Dans la partie suivante, nous considérons les conditions nécessaires pour calculer cet indicateur.

5.3.3 Une API minimale pour les services SPARQL

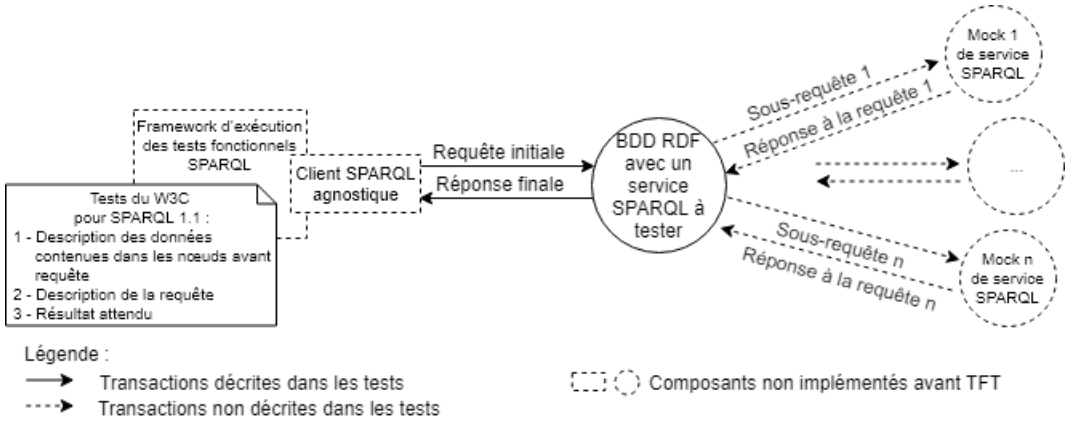
Dans cette section, nous expliquons le fonctionnement d'un service SPARQL au sein du Linked Data et les différences entre éditeurs qui subsistent encore cinq années après la recommandation du protocole SPARQL 1.1 par le W3C.

Fonctionnement d'un service SPARQL. Comme nous l'avons vu dans la section 2.3, un service SPARQL sert de point d'accès entre une application et une base de données RDF. Comme pour les bases de données relationnelles et SQL, SPARQL est un langage de requête standard que les éditeurs de bases de données RDF avec un service SPARQL s'engagent à respecter. Par contre, les différences sont nombreuses entre SQL et SPARQL et elles visent à transformer un silo hermétique de données en un nœud de stockage et de transmission d'informations auprès d'applications, mais aussi d'autres nœuds du réseau. La figure 5.1 (a) représente la communication entre les nœuds du Linked Data et une application à l'aide d'un client SPARQL. Ce type de client peut interroger plusieurs nœuds de ce réseau au travers d'une requête, mais pour chaque requête, le client ne passera que par un seul service SPARQL pour interroger ce réseau. Chaque nœud du réseau est potentiellement un intermédiaire. Une requête interrogeant plusieurs nœuds du réseau (toujours au travers d'un seul service SPARQL) se nomme une requête fédérée.

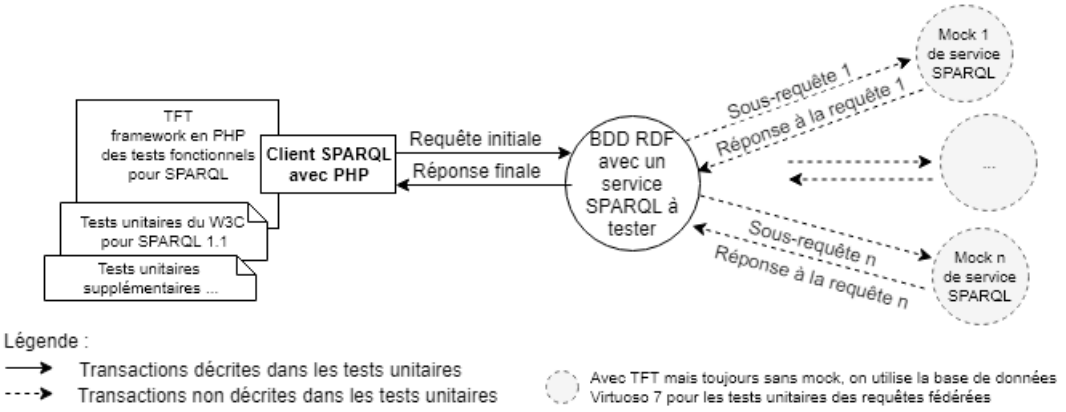
Différences entre les API des services SPARQL. La figure 5.1 (b) représente le niveau de détail des tests fonctionnels datant de 2013, qui ont peu évolué jusqu'à aujourd'hui.



(a) Une application utilise un client SPARQL qui communique avec un seul nœud du Linked data (au travers de son service SPARQL) pour consulter les données se trouvant dans un ou plusieurs des nœuds du Linked Data.



(b) Le W3C ne fournit que des tests décrivant les transactions entre un client et un nœud du Linked Data. Il n'y a aucun outil permettant d'appliquer ces tests sans TFT.



(c) TFT permet d'exécuter les tests du W3C. Il n'existe pour le moment aucun *mock* SPARQL (simulacre de service SPARQL) permettant de vérifier l'interopérabilité entre les nœuds.

Figure 5.1 – (a) décrit les interactions qui existent entre une application et le Linked Data, (b) décrit les interactions qui existent au travers des tests définis dans la spécification SPARQL 1.1., (c) reprend (b) et montre l'importance de la contribution TFT dans le processus d'exécution des tests.

Il faut comprendre qu'un test fonctionnel est un test particulier qui vérifie de bout en bout une application, comme le ferait un utilisateur ou une machine en conditions réelles. Ces tests sont généralement complexes à implémenter et les éditeurs renoncent souvent à le faire. Les résultats des tests fonctionnels en 2013 pour valider la recommandation SPARQL 1.1 étaient donc généralement déclaratifs. Les résultats aux tests fournis par chaque éditeur [151] en 2013 au format EARL [2] ont permis de générer la synthèse officielle *SPARQL 1.1 Test Results* [156]) qui indiquait que 100% de la spécification concernant le protocole était implémentée.

Dans les faits, si un développeur développe un client SPARQL, les chances pour que ce même client fonctionne en lecture et écriture sur des services SPARQL d'éditeurs de logiciels différents sont faibles. En effet, le protocole SPARQL est rarement implémenté de la même façon d'un éditeur à l'autre.

N'ayant pas trouvé de client SPARQL compatible avec tous les éditeurs pour développer la plateforme LinkedWiki, nous avons implémenté notre propre client SPARQL [79].

En théorie, dans les spécifications du W3C et dans les ontologies décrivant les services SPARQL, comme DCAT (*Data Catalog Vocabulary*) [93] ou SPARQL 1.1 Service [142], il est convenu que seul l'URI du point d'accès est nécessaire pour initialiser une transaction SPARQL en lecture ou en écriture [142, 93]. Les autres paramètres sont donc logiquement optionnels.

En réalité, les paramètres nécessaires varient d'un éditeur à un autre. Les tests du W3C en 2013 se sont concentrés sur le langage de requêtes et les formats des données en sortie. Les tests concernant le protocole réseau étaient uniquement décrits en langage naturel et ils étaient donc soumis à l'interprétation d'un être humain.

Si on étudie les API de différents éditeurs, on découvre qu'il faut au minimum quatre paramètres en lecture pour être compatible avec la majorité d'entre eux. Avec l'API en écriture, on monte à huit paramètres, dont voici la liste qui est probablement incomplète car nous n'avons testé que quelques éditeurs :

1. Il faut connaître le point d'accès en lecture (URI) du service SPARQL.
2. Le point d'accès en écriture est aussi nécessaire, car il peut être différent de celui de lecture.
3. En lecture, la méthode HTTP (*Get* ou *Post*) peut être nécessaire car certains services supportent l'un ou l'autre mais pas les deux.
4. En lecture, il faut connaître la méthode de transmission de la requête SPARQL. Le paquet HTTP peut utiliser le paramètre « query » (en Get ou Post) ou passer la requête sans paramètre (dans la partie `body` du paquet HTTP).
5. En lecture, pour définir le format de la réponse du service, le paramètre *Accept* de la requête HTTP peut être nécessaire ou non et le type indiqué doit être supporté par le service SPARQL.
6. En écriture, le paramètre *Accept* est nécessaire pour les mêmes raisons qu'en lecture.
7. En écriture, la méthode HTTP supportée (*Get* ou *Post*) peut être différente de la méthode en lecture.

8. En écriture, la méthode de transmission de la requête SPARQL dans le paquet HTTP peut utiliser le paramètre « query », « update » ou bien aucun (dans la partie `body` du paquet HTTP avec le `Content-Type: application/sparql-update`).

Pour le format de sortie, là encore des différences existent dans l'application des spécifications du protocole. Il faut aussi prendre en compte au minimum deux paramètres pour le contrôle d'accès (*login/mot de passe*), ce qui n'est pas évoqué dans la spécification.

La raison la plus probable de cette multitude de paramètres est la suivante : les spécifications qui rassemblent toutes les API implémentées par les éditeurs en 2013, donnent un éventail trop large de fonctionnalités et les éditeurs n'implémentent en général que les parties de l'API qui leur demandent un effort raisonnable et qui répondent aux besoins de leurs clients.

Ce problème d'implémentation sélective de la spécification concernant le protocole n'est pas une exception. Des requêtes fonctionnant pour certains services SPARQL peuvent ne pas fonctionner pour d'autres cinq ans après la recommandation de SPARQL 1.1. La spécification ne semble donc pas suffisante pour mettre en place une totale interopérabilité avec le langage et le protocole SPARQL.

Même si le non-respect de la spécification du langage SPARQL est un problème, il ne constitue pas un problème bloquant pour les développeurs, qui peuvent éviter d'utiliser certaines parties de la spécification.

Par contre, les différences dans l'implémentation du protocole constituent un problème bloquant, qui peut empêcher les communications entre un client SPARQL et un service. C'est un problème récurrent qui empêche le calcul de notre indicateur d'interopérabilité quand il survient.

Face à ces différences dans l'implémentation du protocole, nous avons essayé deux approches.

Approche commune : s'adapter au protocole de l'éditeur. Généralement, les clients SPARQL s'adaptent aux implémentations des éditeurs en insérant à chaque fois les paramètres nécessaires à leurs API. Le fait de mettre au cœur de TFT notre client SPARQL, comme l'illustre la figure 5.1 (c), nous a permis de construire un client SPARQL plus robuste.

La conséquence de cette adaptation aux différences entre éditeurs est que l'utilisation de TFT est devenue très complexe car nous avons dû ajouter jusqu'à sept paramètres supplémentaires. De plus, nous avons même utilisé la technologie Docker pour décrire l'installation et la configuration nécessaires des bases de données RDF, afin de faciliter la reproductibilité de nos tests.

A partir de là, les éditeurs auxquels nous nous sommes adaptés ont corrigé plusieurs de leurs problèmes concernant les requêtes en lecture ou en écriture.

Malheureusement, pour les autres éditeurs, TFT ne peut fonctionner correctement avec leurs API.

Après trois ans d'existence de TFT, ces problèmes de protocoles persistent et l'implémentation de TFT est presque aboutie.

Notre approche : définir une API SPARQL minimale et réaligner les API qui divergent. Pour chercher une solution, nous avons implémenté les 34 tests

de protocoles du W3C (qui étaient décrits uniquement en langage naturel) avec la solution JMeter [74].

Ces nouveaux tests ont permis de mettre clairement en évidence l'inconsistance de l'implémentation des API entre éditeurs. Sachant cela, nous avons choisi trois de ces tests, définissant ainsi une API minimale. On considère cette API comme indispensable mais également la plus simple pour lire et écrire au travers d'un service SPARQL (noms des trois tests : `query via GET`, `query via URL-encoded POST` et `update via URL-encoded POST`).

Voici les conditions minimales que nous avons choisies :

— En lecture ou en écriture, l'URI du point d'accès du service SPARQL est identique et se termine par : `/sparql`
(par exemple : `http://exemple.org/dataset/sparql`)

— En lecture, les méthodes HTTP GET ou POST sont supportées et elles utiliseront le même paramètre nommé `query` pour transmettre une requête.

— En écriture, seule la méthode POST est supportée et elle utilisera le paramètre nommé `update` pour transmettre une requête.

— En lecture ou écriture, la réponse par défaut (c'est-à-dire sans paramètre) du service doit être au format *SPARQL Query Results XML* [60].

Nous estimons que ces quatre conditions constituent la solution la plus simple pour les utilisateurs et les éditeurs. De plus, cette API minimale est conforme à la recommandation du W3C, nous espérons que les éditeurs feront l'effort d'implémenter au minimum cette API à l'avenir s'ils souhaitent connaître leur score d'interopérabilité.

Cette API minimale fait partie des conditions nécessaires pour permettre à TFT d'évaluer l'interopérabilité d'un service SPARQL. Dans la section suivante, nous allons décrire toutes ces conditions.

5.4 Conditions pour évaluer l'interopérabilité

Dans cette section, nous présentons les conditions pour le bon fonctionnement de TFT et les moyens de contourner des problèmes que l'on peut rencontrer.

Une condition pour que TFT puisse fonctionner. TFT a besoin d'une API SPARQL minimale pour pouvoir exécuter les requêtes nécessaires aux tests. Par conséquent, les bases de données doivent supporter les requêtes qui :

— suppriment toutes les données dans toutes les bases de données (la base de données du service SPARQL à tester ainsi que les bases de données impliquées dans les tests des requêtes fédérées) ;

— chargent les données pour initialiser les bases de données avant le test ;

— et contrôlent la réponse du test et l'état final obtenu dans les bases de données.

Pour que nous soyons dans ce cadre, il faudrait que le W3C recommande une API minimale et que les éditeurs la respectent.

On ne sait pas combien de temps, il faudra attendre pour que les conditions ci-dessus soient réalisées. On devra se contenter d'une solution de contournement tant que ces conditions ne sont pas remplies.

Une solution de contournement pour obtenir une API SPARQL minimale. En attendant l'implémentation d'une API minimale, il existe une solution technique qui consiste à intercepter et modifier les paquets HTTP avant que le service SPARQL ne les reçoive.

Ainsi, il est possible de substituer une API non conforme par une API minimale, par exemple avec le logiciel Varnish [149]. On utilise cette méthode, si nécessaire, au sein des images Docker de nos bancs de tests pour obtenir cette API minimale des services SPARQL que nous testons.

Malheureusement, respecter cette API minimale ne suffit pas pour exécuter les tests. Il faut que le service puisse aussi supporter des requêtes SPARQL précises.

Supporter la suppression des données avec une requête `CLEAR ALL`. Pour réinitialiser la base de données avant chaque test, on utilise la requête `CLEAR ALL` (ou `CLEAR GRAPH`) pour vider la base de données. Puis, on vérifie que le nombre de triplets dans la base de données est bien nul.

Si compter le nombre de triplets ne pose pas de problèmes la requête d'écrasement des données est rarement autorisée par défaut. Pire, cette requête peut être bridée ou bien ne pas être implémentée.

Si une base de données ne supporte pas la requête `CLEAR ALL`, il faut adapter TFT ou demander à l'éditeur de supporter cette requête.

Après cette phase de suppression de toutes les données, TFT utilise des requêtes pour définir l'état initial de la base de données avant chaque test.

Supporter le chargement des données avec une requête `LOAD`. L'initialisation des bases de données avant chaque test utilise la requête `LOAD INTO GRAPH`.

Si cette requête n'est pas supportée, il faut adapter TFT, mais la requête `LOAD` est généralement disponible.

Les problèmes apparaissent avec la requête `LOAD` au chargement du fichier d'initialisation des données au format RDF/Turtle 1.0 (le format Turtle 1.1 date de 2014 et les tests SPARQL datent de 2013). La base de données doit supporter intégralement le format RDF/Turtle 1.0, mais ce n'est pas toujours le cas.

Depuis que les services SPARQL s'exécutent via Docker, on ne peut plus contourner ce problème (en transformant les fichiers avant une injection par ligne de commande). Il faut que les bases de données supportent le format RDF/Turtle 1.0 pour charger les données d'initialisation sinon le test échoue.

Dans la partie suivante, nous parlerons des fonctionnalités qu'offre TFT.

5.5 Fonctionnalités

Dans cette section, nous décrivons les fonctionnalités de TFT. Puis, nous évoquons la manière de consulter les résultats obtenus.

Les trois fonctionnalités principales de TFT. Elles sont à destination des utilisateurs avancés en charge d'utiliser, déployer ou développer des services SPARQL.

Le framework TFT contient trois scripts qui ont chacun une fonction précise. Voici ces trois fonctionnalités :

1. charger dans une base de données RDF les suites de tests officiels du W3C et d'autres tests ajoutés (par des utilisateurs) ;
2. exécuter les 525 tests (le 20/08/2018) ;
3. calculer un score qui sert d'indicateur d'interopérabilité.

TFT possède d'autres fonctionnalités décrites ci-après.

Les résultats sont accessibles via un service SPARQL. En parallèle, TFT partage les résultats (le score et la description des erreurs) au travers d'un service SPARQL.

Ce score d'interopérabilité alimente la plateforme LinkedWiki (PAAS) qui facilite la mise à disposition de bases de données de dernière génération auprès des chercheurs tout en permettant de vérifier l'interopérabilité de leur service SPARQL.

Les résultats sont consultables via des interfaces Web. La figure 5.2 montre les interfaces qui permettent à des humains de consulter les résultats publics. Ces trois interfaces ont été imaginées pour trois types d'utilisateurs :

— les développeurs de requêtes SPARQL : le site Web SPARQL Score permet de consulter et visualiser le détail des requêtes supportées par les logiciels ;

— les développeurs de services SPARQL : ce site Web permet également de consulter les détails des erreurs permettant le débogage de leurs logiciels ;

— les membres du W3C : nous avons commencé à créer un exemple de rapport d'avancement, basé sur nos résultats, permettant de voir l'avancement de l'implémentation de SPARQL 1.1 (avec uniquement les tests officiels et nos trois tests de protocole).

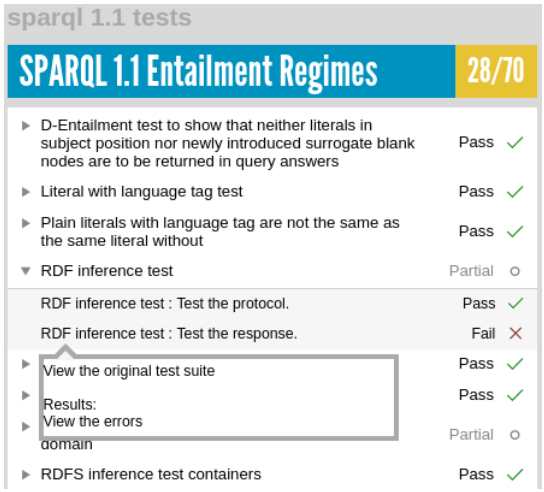
L'intégration continue des services SPARQL. TFT sauvegarde les résultats sous le format *JUnit*, qui est supporté par les outils de développement et les solutions d'intégration continue comme Jenkins [80], Phabricator, Travis CI, etc.

Si un éditeur intègre TFT dans son serveur d'intégration, il pourra ainsi rejeter automatiquement la dernière version de son logiciel si un résultat dans l'un des fichiers JUnit montre une régression de l'interopérabilité. De cette manière, TFT pourra améliorer la qualité de ses services SPARQL.

Ajouter simplement de nouveaux tests. Les DSI peuvent également vérifier que leurs bases de données RDF sont interopérables entre elles avant de les mettre à disposition de leurs utilisateurs. Pour cela, une DSI peut créer ses propres tests.

Cela facilitera leur travail dans leur mission de préservation des données en s'assurant que leurs données seront capables de migrer d'une base de données à une autre.

Dans la section suivante, nous décrivons la manière de créer un banc de tests avec TFT.



(a) Le site Web SPARQL Score permet d'afficher les résultats des tests et les causes des échecs (lien Web).

nb	Specification	Nb_Tests	Jena	Blazegraph	Stardog	Virtuoso
1	SPARQL 1.1 Protocol	3	100%	100%	100%	100%
2	SPARQL 1.1 Query Language	264	91%	88%	91%	73%
3	SPARQL 1.1 Update	156	100%	96%	95%	66%
4	SPARQL 1.1 Query Results CSV and TSV Formats	6	50%	50%	0%	0%
5	SPARQL 1.1 Query Results JSON Format	4	50%	50%	50%	50%
6	SPARQL 1.1 Federation Extensions	10	60%	40%	60%	30%
7	SPARQL 1.1 Entailment Regimes	70	40%	40%	40%	40%

(b) A partir des résultats, on peut générer un compte rendu de l'implémentation des spécifications SPARQL 1.1 pour chaque logiciel testé. (lien Web)

nb	Specification	Nb_tests	Implemented	Percent
1	SPARQL 1.1 Protocol	3	3	100%
2	SPARQL 1.1 Query Language	265	256	97%
3	SPARQL 1.1 Update	156	156	100%
4	SPARQL 1.1 Query Results CSV and TSV Formats	6	3	50%
5	SPARQL 1.1 Query Results JSON Format	4	2	50%
6	SPARQL 1.1 Federation Extensions	10	6	60%
7	SPARQL 1.1 Entailment Regimes	70	28	40%

(c) A partir des résultats, on peut générer un compte rendu de l'implémentation des spécifications SPARQL 1.1 tous logiciels confondus. (lien Web)

Server name	Version	test tool	Score
Jena Fuseki	3.8.0	TFT v1.0	445/525
Stardog	5.3.3	TFT v1.0	433/525
Blazegraph	v2.2.0	TFT v1.0	422/525
Virtuoso Open-Source Edition	stable/7	TFT v1.0	336/525

(d) Le site Web SPARQL Score affiche le nombre de tests réussis, qui nous servira d'indicateur d'interopérabilité. (lien Web).

Figure 5.2 – Interfaces pour visualiser les résultats des tests en fonction des besoins : (a) pour voir le détail des échecs afin de les corriger, (b) pour comparer les logiciels, (c) pour voir l'avancement de l'implémentation définie dans la spécification SPARQL 1.1 et (d) pour visualiser l'indicateur d'interopérabilité.

5.6 Installation de bancs de tests

Cette section décrit la méthode pour construire et configurer les bancs de tests.

Définition des tests à effectuer. Pour l'instant, il existe trois collections de tests : la suite de tests SPARQL 1.1 du W3C (525 tests le 20/08/2018) et deux petites suites de tests (12 tests) pour la norme GeoSPARQL [100] et pour le projet GO, qui était un projet de recherche [53]. Le fichier `config.ini` à la racine du projet TFT [107] définit les collections de tests et peut être étendu si nécessaire. Chaque collection de tests est accessible via une adresse Web qui contient à sa racine un fichier nommé `manifest-all.ttl` renfermant les liens vers tous les fichiers nécessaires aux tests dans le format défini par le W3C. Nous hébergeons les fichiers de tests (GO et GeoSPARQL) via GitHub, qui les sauvegarde et les publie sur le Web [158]. Le W3C héberge maintenant ses tests [152] de la même manière pour permettre aux développeurs de les améliorer et au framework TFT de les télécharger.

La majorité des tests fournis avec la spécification de SPARQL 1.1 sont suffisamment détaillés pour les rendre automatisables, sauf 34 tests de protocoles et trois tests de *description de service* car ils sont décrits en langage naturel. Pour vérifier que l'on peut appliquer TFT à ce service, nous avons implémenté et activé seulement trois tests de protocole (vus en 5.3.3) pour contrôler que l'API minimale nécessaire à TFT est au moins supportée.

Les tests automatisables les plus difficiles à mettre en œuvre sont ceux qui portent sur des requêtes fédérées, car cela nécessite de mettre en réseau simultanément plusieurs services SPARQL avec des adresses réseaux précises.

Les tests de requêtes fédérées du W3C, les plus exigeants, appellent trois services SPARQL simultanément aux adresses fictives suivantes : `http://example.org/sparql`, `http://example1.org/sparql` et `http://example2.org/sparql`.

Le nombre de services SPARQL nécessaires durant les tests détermine la difficulté de construction du banc de tests.

Créer un banc de tests. Le nombre de services SPARQL supplémentaires nécessaires durant les tests de requêtes fédérées peut varier. Il est de trois actuellement, mais rien n'empêche de rajouter de nouveaux tests et d'augmenter ce nombre.

Pour faire varier ce nombre de services SPARQL à la demande, nous avons utilisé des machines virtuelles avec OpenStack dans un premier temps pour exécuter ces tests. Ensuite, nous avons opté pour un environnement Docker afin d'instancier automatiquement un banc de tests. Dans un environnement Docker, le concept de conteneur remplace celui de machine virtuelle, mais le concept est le même. Cependant, la manière d'allouer les ressources matérielles est différente et permet à Docker d'être moins gourmand en ressources. Pour démarrer un conteneur, il faut, comme pour une machine virtuelle, construire au préalable une image qui servira à instancier ce conteneur. Les logiciels d'intégration continue utilisent maintenant de manière native Docker afin de faciliter les tests logiciels. Travis CI, une des solutions d'intégration continue associée à GitHub, utilise également Docker et offre gratuitement la possibilité de tester les logiciels *open source*. En construisant un banc de tests avec GitHub et Travis CI, n'importe quel développeur peut dupliquer (*Fork*)

un banc de tests et reproduire nos résultats sans aucune ressource matérielle de son côté ou compétences systèmes ou réseaux.

Un banc de tests est construit à partir de trois images Docker, toutes basées sur l'image d'un serveur CentOS 7. Les bases de données RDF avec leur service SPARQL s'installent de la même manière qu'en situation réelle, c'est-à-dire sur une machine dédiée avec un système d'exploitation comme CentOS. Cela permet de tester la manière d'installer et de configurer le service en conditions réelles et de réaligner son API avec le logiciel Varnish si l'API minimale attendue n'est toujours pas implémentée par l'éditeur.

Voici la liste des 3 images nécessaires que nous utilisons :

— une image de serveur avec la base de données Jena-Fuseki pour stocker les tests et les résultats obtenus ;

— une image de serveur Virtuoso Openlink pour les services nécessaires aux tests des requêtes fédérées ;

— une image de serveur qui héberge le service SPARQL à tester.

Les deux premières images sont déjà construites et peuvent être réutilisées pour chaque nouveau banc de tests. Après chaque nouvelle version des bases de données, les images sont compilées, passent les tests et sont automatiquement mises à disposition sur le service Docker.io pour servir dans les autres bancs de tests (ID docker : bordercloud/tft-virtuoso7-stable et bordercloud/tft-jena-fuseki).

La base de données Jena-Fuseki est utilisée pour le stockage car elle supporte les requêtes SPARQL avancées (avec la syntaxe *property path*) dont a besoin TFT pour fonctionner. Ce type de requêtes est encore peu supporté par les bases de données RDF disponibles.

Comme indiqué dans la figure 5.1, on préconise d'utiliser des *mocks*, c'est-à-dire des simulacres de services SPARQL pour vérifier réellement les requêtes émises par le service SPARQL à tester. Malheureusement, ces simulacres restent à développer. Pour des raisons historiques, on utilise à la place Virtuoso Openlink pour les images de services nécessaires aux tests des requêtes fédérées.

Pour l'image du service SPARQL à tester, il n'est pas suffisant de récupérer une version Docker du service que l'on peut trouver sur le Web, car il faut autoriser les requêtes SPARQL en écriture pour les utilisateurs anonymes au démarrage du service (ce qui n'est jamais autorisé par défaut). De plus, il faut également réaligner l'API SPARQL, si nécessaire.

Pour chaque nouveau banc de tests, seule l'image du service SPARQL à tester est à fabriquer.

En 2018, pour démontrer la simplicité de cette méthode avec Docker et le réalignement d'API avec Varnish, nous avons créé quatre projets publics GitHub connectés au service Travis CI qui permet d'instancier un nouveau banc de tests pour les solutions suivantes : Virtuoso Openlink, Jena Fuseki, Blazegraph et Stardog. Chacun de ces projets contient les scripts nécessaires pour compiler (et télécharger) les images Docker nécessaires au banc de tests, ainsi que le script `.travis.yml` qui instancie le banc de tests (avec la bonne configuration réseau) et l'exécute. Un banc de tests est exécuté automatiquement à chaque modification de son dépôt GitHub ou en un clic dans le service Travis CI.

```
:test_10  rdf:type mf:QueryEvaluationTest ; #Type of test
          mf:name    "Query to calculate ERT-ART" ;
          dawgt:approval dawgt:Approved;
          #Type of tool to run the test
          mf:feature sd:BasicFederatedQuery ;
          mf:action
                [ qt:query  <q10.rq> ;

                  qt:serviceData [
                        qt:endpoint <http://example1.org/sparql> ;
                        qt:data     <pbs.ttl>
                                  ];
                  qt:serviceData [
                        qt:endpoint <http://example2.org/sparql> ;
                        qt:data     <bdii.ttl>
                                  ]
                ] ;
          mf:result  <q10.srx> .
```

Figure 5.3 – Exemple de test de type requête fédérée.

Indiquer les adresses réseaux des services SPARQL dans le banc de tests. La figure 5.3 montre un exemple de test pour une requête fédérée. Ce test nécessite la présence de deux autres services SPARQL en plus du service qui exécute cette requête. Avant l'exécution du test, le fichier `pbs.tll` contient les données à charger dans le premier service SPARQL distant, et le fichier bdii.ttl les données dans le second service SPARQL distant. Pendant les tests, TFT remplacera les URI factices des services SPARQL dans les tests par les URI contenus dans le fichier `config.ini`.

Dans la figure 5.4, on peut lire la configuration qui remplace les URI factices par les adresses réelles des services après le démarrage des conteneurs Docker dans le banc de tests.

Chargement des tests. C'est le script `tft-testsuite` qui télécharge tous les fichiers de type Turtle (ttl) pour les enregistrer dans la base de données locale du banc de tests. Si la base de données est permanente, cette opération est à faire une seule fois.

La figure 5.5 (5) montre l'appel de ce script permettant de charger les tests dans la base de données locale du banc de tests. Ce script a été codé avant que l'on réaligne les API. Il faudra le mettre à jour pour simplifier son paramétrage. En attendant, pour utiliser une base de données Jena Fuseki, il faut préciser son point d'accès en écriture et lecture ainsi que son type de comportement, ici « fuseki ».

Exécuter les tests. Le script `tft` exécute uniquement les tests contenus dans les graphes précisés dans le fichier `config.ini` au sein de la base de données RDF du banc de tests.

La figure 5.5 (6) montre l'appel du script permettant d'exécuter les tests. On peut y voir que les scripts `tft-testsuite` (figure 5.5 (5)) et `tft` utilisent les mêmes paramètres pour accéder à la base de données du banc de tests.

Si l'API minimale (5.3.3) est respectée, on peut utiliser le paramètre unique `-te` pour préciser l' URI du point d'accès du service SPARQL à tester.

Si le comportement du service SPARQL quant à la suppression et au chargement des données (5.4) est déjà pris en compte par TFT, on utilise le paramètre `-tt` pour

```
[CONFIG]
; (1) Liste des comportements de bases de données connus de \TFT
listTriplestore[] = "standardSparql11"
listTriplestore[] = "4store"
listTriplestore[] = "sesame"
listTriplestore[] = "fuseki"
listTriplestore[] = "virtuoso"
listTriplestore[] = "allegrograph"

; (2) Liste des suites de tests à télécharger sur le Web
listTestSuite["https://bordercloud.github.io/rdf-tests/sparql11/data-sparql11/"] = "
    ↪ tests/rdf-tests/sparql11/data-sparql11/"
listTestSuite["https://bordercloud.github.io/TFT-tests/GO3/"] = "tests/TFT-tests/GO3/"
listTestSuite["https://bordercloud.github.io/TFT-tests/geosparql/"] = "tests/TFT-tests/
    ↪ geosparql/"

; (3) Liste des points d'accès fictifs à remplacer durant les tests
[SERVICE]
endpoint["http://example.org/sparql"] = "http://172.17.0.3/sparql"
endpoint["http://example1.org/sparql"] = "http://172.17.0.4/sparql"
endpoint["http://example2.org/sparql"] = "http://172.17.0.5/sparql"
```

Figure 5.4 – Le fichier de configuration de TFT contient (1) la liste des comportements de bases de données supportées, (2) la liste des suites de tests à télécharger dans un répertoire de destination, et (3) la liste des adresses fictives de service SPARQL à remplacer durant les tests (fichier config.ini dans le logiciel TFT [107]).

le préciser. Dans cet exemple de script, le service de Blazegraph que l'on souhaite tester se comporte comme Jena Fuseki, on précise donc `-tt fuseki`.

Consulter les résultats localement ou à distance. Le script `tft` exécute les tests et sauvegarde leurs résultats.

Pour consulter simplement les résultats localement, le paramètre `-o` précise la destination des fichiers JUnit que le script `tft` génère.

Si l'on souhaite partager les résultats via un service SPARQL ou calculer l'indicateur d'interopérabilité, il est nécessaire d'utiliser le paramètre `-r`. Il précise le graphe de destination des résultats dans la base de données du banc de tests. L'IRI de ce graphe doit être unique à chaque exécution des tests.

Dans les scripts `.travis.yml` pour le service Travis CI, on utilise la variable d'environnement `TRAVIS_JOB_ID` pour générer cet IRI unique (quand on veut exécuter à nouveau le banc de tests via Travis CI, il faut utiliser la fonction « Trigger Build » dans les outils optionnels, et non l'outil « Rebuild », qui réutilise le même ID). Nos exemples de bancs de tests contiennent chacun un exemple de script pour Travis CI [77].

Les paramètres `softwareName`, `softwareDescribeTag` et `softwareDescribe` servent à décrire correctement le logiciel testé si les résultats sont rendus publics au travers du site Web SPARQL Score.

Calculer un indicateur d'interopérabilité. Il serait discutable de choisir un poids particulier pour chacun des 525 tests. Le script `tft-score` calcule et sauvegarde simplement un score où un point est donné pour chaque test réussi.

Il utilise le paramètre `-r` qui précise le graphe de la base de données du banc de test où se trouvent les résultats qui serviront à calculer le score.

```
# (1) Téléchargement des images Dockers
docker pull bordercloud/tft-jena-fuseki
docker pull bordercloud/tft-virtuoso7-stable

# (2) Compilation de l'image Docker du service \SPARQL à tester
docker build -t tft-stardog .

# (3) Déploiement du Linked Data local

# 172.17.0.2
docker run --privileged --name instance.tft-stardog -h tft-stardog -d tft-stardog

# 172.17.0.3
docker run --privileged --name instance.tft.example.org \
          -h example.org -d bordercloud/tft-virtuoso7-stable

# 172.17.0.4
docker run --privileged --name instance.tft.example1.org \
          -h example1.org -d bordercloud/tft-virtuoso7-stable

# 172.17.0.5
docker run --privileged --name instance.tft.example2.org \
          -h example2.org -d bordercloud/tft-virtuoso7-stable

# 172.17.0.6 base de données locale pour sauvegarder les tests et les résultats
docker run --privileged --name instance.tft_database -d tft-jena-fuseki

# (4) Installation de TFT
git clone --recursive https://github.com/BorderCloud/TFT.git
cd TFT
# Installation du SPARQL client dans TFT
composer install
# install JMeter
wget http://.../apache/jmeter/binaries/apache-jmeter-4.0.tgz
tar xvzf apache-jmeter-4.0.tgz
mv  apache-jmeter-4.0 jmeter
rm apache-jmeter-4.0.tgz

# (5) Chargement des tests
php ./tft-testsuite -a -t fuseki -q http://172.17.0.6:8080/test/query \
                    -u http://172.17.0.6:8080/test/update

# (6) Exécution des tests
php ./tft -t fuseki -q http://172.17.0.6:8080/test/query \
                    -u http://172.17.0.6:8080/test/update \
          -tt fuseki -te http://172.17.0.2/blazegraph/namespace/test/sparql \
          -r http://example.org/buildid   \
          -o ./junit  \
          --softwareName="Jena" \
          --softwareDescribeTag=X.X.X \
          --softwareDescribe="Name"

# (7) Calcul du score/indicateur d'interopérabilité
php ./tft-score -t fuseki -q http://172.17.0.6:8080/test/query \
                          -u http://172.17.0.6:8080/test/update \
                -r  http://example.org/buildid
```

Figure 5.5 – Ce script permet de calculer localement l'indicateur d'interopérabilité avec TFT. Ici, il est appliqué à la base de données RDF Blazegraph (lien Web). En voici les étapes : (1) télécharger les images Docker et (2) les compiler, (3) déployer un Linked Data localement, (4) installer TFT, (5) charger les tests, (6) exécuter les tests en sauvegardant les résultats et (7) calculer l'indicateur d'interopérabilité.

Cette méthode d'installation et de configuration nous semble la plus transparente possible et la plus simple pour reproduire nos résultats par n'importe qui en quelques clics. C'est donc la méthode la plus fiable pour produire un indicateur d'interopérabilité transversale à toutes les solutions de services SPARQL qui constituent le Linked Data.

Dans la section suivante, nous évoquons brièvement l'implémentation de nos contributions.

5.7 Implémentation

Cette section vise à décrire principalement les choix technologiques utilisées durant la conception de TFT et du banc de tests permettant de le faire fonctionner.

Implémentation 100% Linked Data. Pour tester les technologies du Linked Data, nous nous sommes imposé d'utiliser exclusivement ces technologies dans notre implémentation.

Ainsi, les tests sont au format *Turtle* 1.0 (Turtle 1.1 a été défini en 2014 après la recommandation de SPARQL 1.1 en 2013) et les bases de données sont alimentées et interrogées uniquement par des requêtes SPARQL quand cela est possible.

De plus, le client dans TFT est le même client SPARQL qu'utilise la plateforme LinkedWiki. Cela a permis de le consolider en le soumettant à ces centaines de tests.

Rendre reproductible notre indicateur d'interopérabilité. Le logiciel TFT est sous licence CC BY-SA 4.0 (Attribution et partage dans les mêmes conditions). Le but de cette licence est de partager le même logiciel pour tester et comparer objectivement les bases de données sur le marché. Les tests TFT, les collections de tests et les bancs de tests avec leurs images Docker sont tous disponibles via leurs dépôts Git [77].

Nous avons utilisé de nombreuses technologies comme PHP pour développer les scripts TFT, Docker pour créer les environnements d'exécution des services SPARQL reproductibles, JMeter pour tester les transactions HTTP et Varnish pour réaligner les API des services SPARQL.

Rendre évolutif notre indicateur d'interopérabilité. Les tests sont disponibles via leurs dépôts Git [158, 153]. Cela simplifie la possibilité d'insérer un nouveau test ou de les corriger en dupliquant le projet d'origine (*Fork*, en anglais) puis en le soumettant (*Pull request*, en anglais).

Rendre accessible notre indicateur d'interopérabilité. Pour les développeurs, le format JUnit suffit généralement à consulter les résultats de ces tests. Nous avons développé le site Web SPARQL Score (figure 5.2) pour visualiser les résultats et les erreurs, mais il est trop complexe à déployer. Un démonstrateur, en vue de le remplacer à terme, est déjà en ligne [77] et permet de visualiser l'avancement d'implémentation et de comparer les éditeurs entre eux pour chaque test. Ce démonstrateur est uniquement en HTML et Javascript (avec Sgvizler2) ; il servira d'exemple pour améliorer SPARQL Score.

Dans la section suivante, nous évaluons l'impact de nos contributions dans la mise en œuvre du Linked Data.

5.8 Évaluation

Dans cette section, nous décrivons le comportement des éditeurs vis-à-vis de nos contributions pour ainsi évaluer l'impact de nos contributions sur la mise en œuvre d'un Linked Data réellement interopérable.

Évaluation de la méthode. En 2015, après le lancement du site SPARQL Score, quatre fournisseurs nous ont contactée pour inclure leur logiciel dans nos tests et trois ont accepté d'ouvrir leurs résultats. Trois fournisseurs ont spécifiquement mis en place un point de terminaison SPARQL pour nos tests et corrigé certains de leurs problèmes d'interopérabilité.

En 2018, nous avons rafraîchi le projet TFT avec l'utilisation de Docker au sein de GitHub avec Travis CI pour simplifier le déploiement des bancs de tests et donc la reproductibilité des résultats.

Aujourd'hui, au moins deux éditeurs utilisent TFT en interne (version avec Docker) et quatre éditeurs nous autorisent à partager les résultats afin d'aider leurs communautés à identifier les erreurs potentielles au sein de la partie open source de leurs logiciels.

Pour améliorer encore TFT, il faudrait implémenter un simulacre (*mock*, en anglais) de service SPARQL afin d'évaluer de manière indiscutable toutes les transactions HTTP entre les services, surtout lorsqu'il s'agit de tests concernant des requêtes fédérées qui impliquent plusieurs services de différents éditeurs simultanément.

Dans le paragraphe suivant, nous décrivons l'impact qu'a l'indicateur d'interopérabilité que calcule TFT pour chaque service SPARQL.

Évaluation de l'indicateur d'interopérabilité. Cet indicateur a été l'un des facteurs qui ont encouragé les éditeurs au sein du W3C à rediscuter en 2017 de la recommandation SPARQL 1.1 de 2013, à corriger de nombreux tests figurant dans la spécification et à créer environ 50 nouveaux tests (concernant le support aux inférences avec l'ontologie RDFS).

Globalement, il n'y a pas de remise en cause de l'indicateur, car TFT recouvre 100% des tests existants (qui ne sont pas en langage naturel).

De plus, on estime que 10% des échecs aux tests exécutés au travers de TFT peuvent être dus à des erreurs d'implémentation ou à des cas acceptables mais que TFT ne sait pas reconnaître. Une solution envisageable pour améliorer ce dernier point serait de permettre aux éditeurs d'étendre eux-mêmes les réponses possibles aux tests.

Cependant, ces quelques problèmes n'empêchent pas certains éditeurs de demander à figurer publiquement dans SPARQL Score pour rendre visible le travail qu'ils fournissent afin de construire un Linked Data plus interopérable.

Évaluation de l'API minimale. Côté protocole, la prise en compte par les éditeurs de notre API minimale est encore trop récente, mais nous l'appliquerons à partir de maintenant dans le déploiement de toutes les nouvelles machines virtuelles qui seront déployées au travers de la plateforme LinkedWiki.

5.9 Conclusion

Dans ce chapitre, nous avons décrit un outil permettant l'évaluation de l'interopérabilité des services SPARQL. Cet outil est indispensable pour mettre en œuvre réellement le Linked Data et à terme le Web sémantique.

Ce travail apporte deux contributions.

La première est évidemment la solution TFT, permettant de calculer un indicateur d'interopérabilité transparent et reproductible localement ou à distance et gratuitement. Cela va faciliter grandement le travail des éditeurs dans la correction de leurs logiciels. De plus, le partage de ce score auprès de services comme la plateforme LinkedWiki permettra d'encourager les utilisateurs de ces technologies à prendre en compte ce facteur d'interopérabilité à l'avenir.

La seconde contribution est moins évidente, mais tout aussi importante. Elle consiste à garantir que chaque service SPARQL testé au travers de TFT respecte une API SPARQL minimale pour permettre à n'importe quel client SPARQL de l'interroger. TFT, après trois ans d'existence, a facilité la correction par les éditeurs des problèmes de protocoles qui subsistent dans leurs logiciels, mais cela reste insuffisant pour offrir une API commune entre tous les éditeurs. En définissant une API SPARQL minimale pour fonctionner avec n'importe quel client SPARQL, nous espérons accélérer la mise en œuvre du Linked Data et nous commençons par l'appliquer aux services SPARQL déployés au sein de l'Université Paris-Saclay. Chacune des images des machines virtuelles dans la plateforme LinkedWiki ou dans chacune des images Docker de TFT (servant aux tests) sera maintenant alignée (si nécessaire) sur cette API minimale.

Il serait possible de faire évoluer TFT pour générer dynamiquement le banc de tests et tous les tests de requêtes fédérés afin de tester toutes les configurations possibles de réseaux avec des services SPARQL différents. Avant cela, il faudra simplifier TFT en retirant les adaptations successives aux différents éditeurs qui sont maintenant inutiles, et ainsi faciliter la participation d'autres développeurs ou chercheurs dans les évolutions de TFT.

Même si beaucoup reste à faire, notre travail apporte des progrès majeurs dans le déploiement du Linked Data au sein d'une université.

Résumé du chapitre 5

Dans ce chapitre, nous introduisons un cadre de tests, nommé TFT (*Tests For Triplestores*), que nous avons implémenté, pour tester l'interopérabilité des services SPARQL des bases de données RDF. 100% de la suite de tests de la recommandation SPARQL 1.1 du W3C est exécuté. De plus, de nouveaux tests d'interopérabilité peuvent y être insérés. Ces tests peuvent être reproduits simplement et offrent ainsi un indicateur fiable de l'interopérabilité actuelle des services SPARQL implémentés. On le nomme SPARQL Score : http://sparqlscore.com. Des exemples de bases de données et leurs résultats sont synthétisés dans la page Web « SPARQL (Core) 1.1 : Test Suite Report » http://tft-reports.bordercloud.com. Cette synthèse propose un instantané objectif de l'implémentation actuelle de la spécification et donc de la mise en œuvre du Linked Data.
Cet indicateur peut aider les utilisateurs finaux à choisir leurs futures bases de données RDF sur le critère de leur réelle compatibilité avec le Linked Data, comme dans la plateforme LinkedWiki du chapitre 3 (voir la figure 3.4).
Seule condition pour effectuer ces tests et calculer cet indicateur : il faut qu'un minimum de la spécification soit respectée par le service SPARQL à tester. Nous définissons dans ce chapitre ce qui nous semble indispensable de respecter dans la spécification SPARQL pour les clients SPARQL et pour passer les tests de la recommandation SPARQL.
Après avoir décrit le problème, nos solutions et les conditions à respecter pour passer ces tests, nous décrivons comment quiconque peut utiliser TFT pour faire son propre banc de tests sans avoir besoin d'aucune ressource matérielle, ainsi que son implémentation et son évaluation par les éditeurs de services SPARQL.

Résultats publiés du chapitre 5

[120] Article publiée : « Certifying the interoperability of RDF database systems ». Karima Rafes, Julien Nauroy et Cécile Germain. Dans : LDQ 2015-2nd Workshop on Linked Data Quality. 2. Springer. 2015.

[121] Démonstration publiée : « TFT, Tests For Triplestores ». Karima Rafes, Julien Nauroy et Cécile Germain. Dans : Semantic Web Challenge, part of the International Semantic Web Conference. 2014.

Chapitre 6

Conclusion

Dans le cadre de cette thèse, nous avons mis en œuvre le Linked Data dans des laboratoires de l'Université Paris-Saclay. Nous avons ainsi pu identifier des freins que rencontraient les chercheurs dans l'utilisation opérationnelle de ces technologies.

Pour prendre en compte les besoins des chercheurs, nous avons développé, au-dessus du cloud de l'université (IAAS), une plateforme (PAAS) adaptée à la création d'environnements virtuels de recherche (VRE) modulaires et compatibles avec le Linked Data. Nous avons ainsi pu offrir aux chercheurs une solution pour découvrir, produire et réutiliser les données de la recherche disponibles au sein du Linked Open Data, c'est-à-dire du système global d'information en train d'émerger à l'échelle du Web.

Cette expérience nous a permis de montrer que l'utilisation opérationnelle du Linked Data au sein d'un laboratoire est parfaitement envisageable. Cependant, des problèmes persistent et sont de deux types :

1. L'industrialisation du Linked Data, c'est-à-dire son utilisation massive, n'a pas encore eu lieu. Cette industrialisation permettrait de lever les derniers verrous techniques, comme le contrôle de la qualité des services SPARQL avant leur déploiement au sein d'un système d'information compatible avec le Linked Data.
2. Le manque d'outils adaptés aux différents métiers est, aussi, bloquant pour le Linked Data, car tant que ses outils n'atteindront pas une maturité équivalente à des outils tels que SQL, une majorité d'utilisateurs sera réticente à basculer vers ces nouvelles technologies.

Pour résoudre les problèmes de déploiement au sein des universités, deux approches sont possibles. La première est d'encourager les chercheurs et leurs organisations à basculer progressivement vers un système d'information compatible avec le Linked Data. Pour soutenir cet effort, les éditeurs d'outils devront se conformer aux exigences du Linked Data et fabriquer les outils métiers indispensables.

Une deuxième approche pourrait se résumer en : il faut créer la « *Killer App* » du Linked Data. Quand les avantages d'une nouvelle technologie sont trop limités pour convaincre les chercheurs de changer de pratique, on peut se contenter d'attendre l'émergence spontanée d'une « *Killer App* ». Plutôt que d'attendre cette éventuelle arrivée, nous préférons la première approche qui permet de démarrer sans attendre

le déploiement du Linked Data. En effet, nous avons besoin dès aujourd'hui d'un système global d'information fiable et décentralisé.

La société a commencé à utiliser massivement les données alors que les algorithmes ne savent pas distinguer la fiabilité des données sur le Web, avec toutes les dérives que cela peut impliquer, comme les « *Fake news* ». Ce phénomène ne peut que s'accentuer, car il découle de l'existence d'informations distribuées de façon massive et incontrôlée. Le Linked Data est une des meilleures réponses dont nous disposons pour permettre aux machines de confronter automatiquement les sources d'informations contradictoires afin d'identifier le degré de fiabilité des informations sur le Web.

Comme ce fut le cas avec le Web, c'est aux universités et aux chercheurs de montrer le chemin en déployant des systèmes d'information compatibles avec le Linked Data, comme nous l'avons décrit dans le chapitre 3. Une fois en place dans plusieurs universités, il sera possible de concevoir en situation réelle les futurs outils indispensables aux consommateurs de ces données comme nous l'avons montré dans le chapitre 4.

La constitution d'un système d'information dédié à la recherche au sein d'une université est un sujet d'étude passionnant, un défi complexe pour les enseignants, les chercheurs, les développeurs, les experts impliqués. Il faudra en permanence identifier les verrous technologiques rencontrés au fur et à mesure du déploiement de ce système d'information ; ils soulèveront de nouveaux problèmes de recherche. Le but unique est de construire un système d'information fiable, décentralisé et sécurisé véritablement au service de la recherche. Quand une ou plusieurs universités auront montré l'exemple, on peut s'attendre à les voir connecter leurs systèmes et à voir d'autres universités appliquer la même stratégie. C'est le chemin pour arriver à un système global d'information, où le Web sémantique pourra alors véritablement prendre son essor et livrer enfin toutes ses promesses.

Pour accélérer cette transformation vers le Web sémantique, nous envisageons de travailler sur une hypothèse que les travaux du chapitre 4 permettent d'envisager : que des agents intelligents puissent utiliser le Linked Open Data sans connaissance préalable ni des schémas RDF, ni des services SPARQL disponibles. A partir de là, par exemple, des agents personnels pourraient se fournir directement en données, voire passer des commandes automatiquement. Mais pour y arriver, des verrous doivent être levés. Les travaux sur les requêtes SPARQL doivent être étendus à des requêtes fédérées. Il faut aussi de nouveaux protocoles et algorithmes, et de nouvelles infrastructures réseaux pour pouvoir trouver automatiquement les sources d'information qui pourraient participer au résultat d'une requête. Il est évident que, dans un cadre comme le Linked Open Data, une solution centralisée n'a pas de sens. L'objectif que nous souhaitons suivre dans les prochaines années est de collaborer à la construction d'un tel système d'information global, décentralisé et accessible à tous.

Annexe A

Les types d'autocomplétions pour SPARQL

Cette annexe est un extrait de notre publication « Une autocomplétion générique de SPARQL dans un contexte multi-services » [118] dont la taxonomie a été mise à jour après la rédaction de cette thèse.

En 2016, nous avons collecté les besoins en autocomplétion avec SPARQL, durant un an, auprès d'utilisateurs aussi bien novices qu'experts : 103 étudiants issus de formations variées (en informatique, en administration des entreprises, du L3 au M2), dans le cadre de travaux pratiques de SPARQL, et 60 professionnels au travers de formations internes et événements proposés au sein du *Center for Data Science* de l'Université Paris-Saclay.

Après cette taxonomie, nous présentons l'état de l'art de l'implémentation actuelle des autocomplétions attendues par les utilisateurs.

A.1 Taxonomie des autocomplétions

Voici la taxonomie des autocomplétions pour SPARQL fondée sur les retours de nos utilisateurs.

Autocomplétion syntaxique. Elle propose à l'utilisateur des mots-clés ou des éléments de syntaxe du langage SPARQL. Elle peut se calculer sans difficulté à l'aide de la notation EBNF du langage SPARQL.

Néanmoins, en pratique, chaque service SPARQL ne supporte pas nécessairement l'ensemble de la syntaxe du langage. Par exemple, le service SPARQL de Wikidata ne supporte pas les requêtes contenant le mot-clé `GRAPH`. En conséquence, un éditeur, selon le service SPARQL auquel il accède, peut proposer des complétions déclenchant des erreurs parce qu'il génère des requêtes SPARQL syntaxiquement correctes mais qui utilisent des éléments de syntaxe non supportés par le service.

Une autocomplétion tenant compte de la syntaxe supportée par un service fait partie des fonctionnalités les plus attendues par les utilisateurs novices et experts. Ce besoin est encore plus fort dans le contexte de requêtes fédérées.

Autocomplétion des variables. Elle propose l'utilisation de variables qui apparaissent déjà dans la requête. Elle doit pouvoir être activée automatiquement par l'éditeur, qui détecte que l'utilisateur commence à écrire une variable (commençant par `?`, notation EBNF `VARNAME`) et propose la liste des variables disponibles. Cette autocomplétion est attendue, bien que certains développeurs préfèrent faire des copier-coller.

Autocomplétion du service. C'est l'autocomplétion de l'*endpoint* du service qui résoudra une des sous-requêtes. Elle propose une liste de services SPARQL opérationnels et accessibles à l'utilisateur pour interroger plusieurs bases en une seule requête. Cette autocomplétion nécessite de disposer d'une base de connaissances des services SPARQL disponibles et accessibles par chaque utilisateur. Le service SPARQL qui calcule la requête fédérée doit également autoriser l'utilisateur à effectuer ce type de requête et les autres services doivent respecter le protocole SPARQL.

Il est important de noter que les implémentations du protocole SPARQL concernant les requêtes fédérées restent disparates entre les éditeurs, entraînant des problèmes d'interopérabilité. Très peu de bases de données RDF permettent concrètement de rédiger des requêtes fédérées [120] et très peu d'utilisateurs accèdent donc à cette fonctionnalité, pourtant fondamentale.

L'autocomplétion du service est donc indispensable pour permettre à l'utilisateur de découvrir les sources de données disponibles.

Autocomplétion du graphe. Cette autocomplétion permet de choisir un graphe (notation `GraphRef` et `NamedGraphClause`) sur lequel portera la (sous-)requête. Il est aussi à noter que dans certaines implémentations, interroger le graphe par défaut du service revient à interroger tous les graphes du service simultanément, alors que dans d'autres implémentations, interroger le graphe par défaut revient à n'interroger qu'un seul graphe. Préciser le nom du graphe permet donc de désambiguïser la requête et d'améliorer le temps de réponse. Cette fonctionnalité est attendue par les utilisateurs experts.

Autocomplétion des IRIs absolus. Elle permet de proposer une liste d'IRIs absolus (notation EBNF `IRIREF`) en s'appuyant sur l'IRI que l'utilisateur a commencé à écrire. Pour un utilisateur expert, l'utilisation d'un IRI absolu se fait par copier-coller. Pour un novice, écrire un IRI absolu est source d'erreurs. On notera que les utilisateurs utilisent surtout des IRIs relatifs, et le besoin de ce type de complétion est donc relativement faible.

Autocomplétion des IRIs relatifs via préfixe. Elle permet la complétion d'IRI relatifs (notation EBNF `PrefixedName`). Dans la spécification d'une ontologie (décrite au moyen d'IRIs), un préfixe est toujours suggéré. L'utilisateur expert exploite souvent ce préfixe. L'autocomplétion des IRIs relatifs via préfixe propose alors une liste de préfixes à partir de quelques caractères, puis, après sélection du préfixe, elle propose les IRIs associés à ce préfixe (types, propriétés...). Cette fonctionnalité est demandée en particulier par les utilisateurs experts.

Autocomplétion des IRI via mots-clés. Cette autocomplétion d'IRIs, à la différence des précédentes, ne présuppose aucune connaissance préalable du service SPARQL et des ontologies qu'il contient. L'utilisateur choisit des mots-clés, dans la

langue de son choix, pour obtenir une liste de suggestions d'IRIs relatifs. Il suffit alors d'en choisir un pour que l'outil puisse l'insérer dans la requête en cours avec la définition du préfixe. Ce type de fonctionnalité est particulièrement demandé par tous nos utilisateurs.

Autocomplétion des préfixes. C'est l'autocomplétion qui insère les déclarations des préfixes non explicitement spécifiés par l'utilisateur au sein d'une requête. Elle répond à un besoin récurrent rencontré par les utilisateurs qui réutilisent des exemples de requêtes disponibles sur le Web dans un nouveau contexte où les préfixes doivent être explicités. Cette fonctionnalité est très demandée par l'ensemble de nos utilisateurs.

Autocomplétion par modèle. Elle propose à l'utilisateur des modèles, c'est-à-dire une liste de morceaux de code SPARQL dans laquelle l'utilisateur peut piocher pour compléter sa requête. Cette fonctionnalité est demandée par l'ensemble de nos utilisateurs.

Suggestion de snippets. Cette autocomplétion suggère des morceaux de requêtes (des snippets) à l'utilisateur quand il conçoit sa requête. Cette fonctionnalité est très attendue par nos utilisateurs expérimentés et permettrait de guider les novices lors de leurs premières requêtes.

A.2 Etat de l'art

Dans [123], dix-huit éditeurs textuels pour SPARQL ont été recensés, parmi lesquels sept intègrent des fonctionnalités d'autocomplétion. Ils ont servi de base à notre étude. Nous présentons ici un rapide état de l'art des fonctionnalités d'autocomplétion qui existent au sein de ces éditeurs.

A notre connaissance, aucun éditeur ne propose de solution au problème de l'autocomplétion *syntaxique* capable prendre en compte les différences entre les services SPARQL qui ne supportent pas toute la syntaxe du langage. On peut faire le même constat sur l'autocomplétion *du graphe* : nous n'avons pas observé d'éditeur en mesure d'optimiser le temps de réponse d'une requête en limitant sa portée à un ou plusieurs graphes nommés au sein du service SPARQL. A contrario, l'autocomplétion *des variables* est une fonctionnalité que l'on retrouve dans plusieurs éditeurs.

L'autocomplétion *du service* n'est pas implémentée par les éditeurs, qui ne considèrent aujourd'hui qu'un service à la fois. La prise en compte de cette fonctionnalité dans un contexte multi-services nécessiterait la constitution d'une plateforme collaborative pour référencer les services SPARQL. LinkedWiki [119] ou Datahub.io [48] permettraient d'alimenter ce type d'autocomplétion.

Les outils d'autocomplétion des *IRIs relatifs et absolus* utilisent plusieurs types d'approches.

Certains éditeurs exploitent les bases de données référençant toutes les ontologies du Web constituées par des approches comme [7] ou LOV [147]. Les IRIs peuvent aussi être extraits du service SPARQL à interroger, comme le font Flint [144] ou YASQE [123]. D'autres systèmes comme SPACE [86] observent les développeurs d'un

service SPARQL (via les logs du service, généralement) et le contenu du service pour constituer un système de recommandations d'IRIs. Certaines approches exploitent l'emplacement de l'IRI dans la requête. Le calcul peut alors se faire en temps réel pour chaque service SPARQL [27] ou en constituant à nouveau une unique base de données RDF avec toutes les données que peut utiliser le développeur [28].

La recherche des IRIs par *mots-clés* reste peu mise en œuvre et il n'existe, à notre connaissance, aucune recherche par mots-clés en langue naturelle.

L'*autocomplétion par modèle* est un outil assez simple à implémenter ; il existe dans de nombreux éditeurs, comme au sein de l'éditeur de Wikidata [164].

Enfin, la *suggestion de snippets* a été introduite dans le cadre de langages de requêtes, par exemple pour SQL avec le système *SnipSuggest* [82]. Nous ne connaissons pas de tels outils pour SPARQL. Une approche sur la base du recensement des requêtes SQL précédentes des développeurs est esquissée dans [82].

Il apparaît de cet état de l'art qu'il n'existe pas d'éditeur proposant un panel complet de fonctionnalités d'autocomplétion et capable de traiter des requêtes fédérées (contexte multi-services). Si l'on peut s'inspirer des approches suivies par certains éditeurs, les limitations des fonctionnalités actuelles sont nombreuses à devoir être surmontées pour permettre leur utilisation en environnement de production. Face aux masses de données RDF disponibles et à leur caractère fortement distribué, l'autocomplétion doit, en particulier, fonctionner dans un cadre multi-services, en isolation (pour protéger les données), et offrir de l'autocomplétion de snippets, comme le proposent typiquement les éditeurs de code de type professionnel (ne gérant pas du code SPARQL).

A l'issue de ce rapide et premier état de l'art, nous avons sélectionné les quatre autocomplétions les plus intéressantes à fournir aux utilisateurs. Puis, comme nous l'avons décrit dans 3.2.3.3, nous avons fait un état de l'art plus poussé sur ces quatre autocomplétions, avant de les implémenter.

Bibliographie

[1] Serge Abiteboul et al. « Auto-completion learning for XML ». Dans : *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM. 2012, p. 669–672.

[2] Shadi Abou-Zahra et W3C/WAI. *Evaluation and Report Language (EARL) 1.0 Schema*. 2011.

[3] Ben Adida et al. « RDFa in XHTML : Syntax and processing ». Dans : *W3C recommendation* (2008).

[4] AKSW. *RDFUnit : an RDF Unit-Testing suite*. 2016.

[5] Bahram Amini et al. « A reference ontology for profiling scholar's background knowledge in recommender systems ». Dans : *Expert Systems with Applications* 42.2 (2015), p. 913–928.

[6] Ricardo Carvalho Amorim et al. « A Comparative Study of Platforms for Research Data Management : Interoperability, Metadata Capabilities and Integration Potential ». Dans : *New Contributions in Information Systems and Technologies*. Springer, 2015, p. 101–111.

[7] Aleksandar Andreevski et al. « Semantic Web Integration with SPARQL Autocomplete ». Dans : *The 12th International Conference on Informatics and Information Technologies*. 2015, p. 1–4.

[8] Jayanta Basak et Raghu Krishnapuram. « Interpretable hierarchical clustering by constructing an unsupervised decision tree ». Dans : *IEEE transactions on knowledge and data engineering* 17.1 (2005), p. 121–132.

[9] Dave Beckett et Brian McBride. « RDF/XML syntax specification (revised) ». Dans : *W3C recommendation* 10.2.3 (2004).

[10] David Beckett et Tim Berners-Lee. *Turtle–Terse RDF Triple Language*. Jan. 2008.

[11] Claude Bernard et Paul Bert. *La science expérimentale*. JB Baillière & fils, 1878.

[12] T Berners-Lee, R Fielding et L Masinter. *World Wide Web Consortium supports the IETF URI standard and iri proposed standard*. Jan. 2004.

[13] Tim Berners-Lee. « Tim Berners-Lee : I invented the web. Here are three things we need to change to save it' ». Dans : *The Guardian* 12.03 (2017).

[14] Tim Berners-Lee. « The next web ». Dans : *TED.com* (2009).

[15] Tim Berners-Lee. « Linked data-design issues ». Dans : *W3C* (2006).

[16] Timothy J Berners-Lee. *Information management : A proposal*. Rapp. tech. 1989.

[17] Tim Berners-Lee, James Hendler, Ora Lassila et al. « The semantic web ». Dans : *Scientific american* 284.5 (2001), p. 28–37.

[18] Adrian Bielefeldt, Julius Gonsior et Markus Krötzsch. « Practical Linked Data Access via SPARQL : The Case of Wikidata ». Dans : *Proc. WWW2018 Workshop on Linked Data on the Web (LDOW-18). CEUR Workshop Proceedings, CEUR-WS. org*. 2018.

[19] Christian Bizer, Tom Heath et Tim Berners-Lee. « Linked data : Principles and state of the art ». Dans : *World wide web conference*. 2008, p. 1–40.

[20] Christian Bizer, Tom Heath et Tim Berners-Lee. « Linked data : The story so far ». Dans : *Semantic services, interoperability and web applications : emerging concepts*. IGI Global, 2011, p. 205–227.

[21] Christian Bizer et Andreas Schultz. « The Berlin SPARQL benchmark ». Dans : *Int. Jal. On Semantic Web and Information Systems* 4.2 (2009), p. 1–24.

[22] Blazegraph. *Documentation : SPARQL Update*.

[23] Daniel Boley. « A scalable hierarchical algorithm for unsupervised clustering ». Dans : *Data Mining for Scientific and Engineering Applications*. Springer, 2001, p. 383–400.

[24] Kurt Bollacker et al. « Freebase : a collaboratively created graph database for structuring human knowledge ». Dans : *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM. 2008, p. 1247–1250.

[25] Dan Brickley, Ramanathan V Guha et Brian McBride. « RDF Schema 1.1 ». Dans : *W3C recommendation* 25 (2014), p. 2004–2014.

[26] Sergey Brin et al. « What can you do with a web in your pocket ? » Dans : *IEEE Data Eng. Bull.* 21.2 (1998), p. 37–47.

[27] Stéphane Campinas. « Live SPARQL auto-completion ». Dans : *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*. CEUR-WS. org. 2014, p. 477–480.

[28] Stephane Campinas et al. « Introducing RDF graph summary with application to assisted SPARQL formulation ». Dans : *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*. IEEE. 2012, p. 261–266.

[29] Leonardo Candela, Donatella Castelli et Pasquale Pagano. « Virtual research environments : an overview and a research agenda ». Dans : *Data Science Journal* 12 (2013), GRDI75–GRDI81.

[30] Jeremy J Carroll et al. « Named graphs, provenance and trust ». Dans : *Proceedings of the 14th international conference on World Wide Web*. ACM. 2005, p. 613–622.

[31] Alejandra Casas-Bayona et Hector G Ceballos. « Integrating semi-structured information using Semantic Technologies ». Dans : *Proceedings of 3rd International Conference on Data Management Technologies and Applications*. SCITEPRESS-Science et Technology Publications, Lda. 2014, p. 357–364.

[32] Kārlis Čerāns et al. « ViziQuer : A Web-Based Tool for Visual Diagrammatic Queries Over RDF Data ». Dans : *European Semantic Web Conference*. Springer. 2018, p. 158–163.

[33] Gong Cheng et al. « Generating Illustrative Snippets for Open Data on the Web ». Dans : *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM. 2017, p. 151–159.

[34] Beckett Dave. *Rapper — Raptor RDF parsing and serializing utility*. 2014.

[35] Gianluca Demartini. « Finding experts using wikipedia ». Dans : *Proceedings of the 2nd International Conference on Finding Experts on the Web with Semantics-Volume 290*. Citeseer. 2007, p. 33–41.

[36] DERI, NUI Galway. *Prefix.cc : namespace lookup for RDF developers*. 2010.

[37] Sara Di Bartolomeo et al. « SPARQLING : painlessly drawing SPARQL queries over GRAPHOL ontologies ». Dans : 2187 (2018).

[38] Renata Queiroz Dividino et Gerd Gröner. « Which of the following SPARQL Queries are Similar ? Why ? » Dans : *LD4IE@ ISWC*. 2013.

[39] H Else. « Europe's open-access drive escalates as university stand-offs spread. » Dans : *Nature* 557.7706 (2018), p. 479.

[40] Ivan Ermilov et al. « LODStats : The Data Web Census Dataset ». Dans : *Proceedings of 15th International Semantic Web Conference - Resources Track (ISWC'2016)*. 2016.

[41] Fredo Erxleben et al. « Introducing Wikidata to the Linked Data Web ». Dans : *The Semantic Web–ISWC 2014*. Springer, 2014, p. 50–65.

[42] DG CONNECT of the European Commission. *Repository of the Open Data Interoperability Platform*. 2014.

[43] Ju Fan, Guoliang Li et Lizhu Zhou. « Interactive SQL query suggestion : Making databases user-friendly ». Dans : *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE. 2011, p. 351–362.

[44] Said Fathalla et al. « Analysing Scholarly Communication Metadata of Computer Science Events ». Dans : *International Conference on Theory and Practice of Digital Libraries*. Springer. 2017, p. 342–354.

[45] Steve Faulkner et al. « HTML 5.2 ». Dans : *W3C Recommendation* (déc. 2017).

[46] Sébastien Ferré. « Sparklis : an expressive query builder for SPARQL endpoints with guidance in natural language ». Dans : *Semantic Web* 8.3 (2017), p. 405–418.

[47] Chris Ferris. « Web services architecture ». Dans : *Standard, W3C World* (2004), p. 10.

[48] Open Knowledge Foundation. *Datahub*. 2015.

[49] The Open Knowledge Foundation. *CKAN, the world's leading open-source data portal platform*. 2015.

[50] Shudi Gao et al. « W3C XML schema definition language (XSD) 1.1 part 1 : Structures ». Dans : *W3C candidate recommendation* 30.7.2 (2009), p. 16.

[51] Paul Gearon, Alexandre Passant et Axel Polleres. « SPARQL 1.1 Update ». Dans : *W3C recommendation* 21 (2013).

[52] Cécile Germain, Julien Nauroy et Karima Rafes. *The Grid Observatory 3.0 - Towards reproducible research and open collaborations using semantic technologies*. EGI Community Forum 2014. Mai 2015.

[53] Cecile Germain-Renaud et al. « The grid observatory ». Dans : *Cluster, Cloud and Grid Computing (CCGrid), 11th IEEE/ACM Int. Symp. on*. IEEE. 2011, p. 114–123.

[54] Stijn Goedertier. « Description of DCAT application profile for data portals in Europe ». Dans : *Europa.eu* (2013).

[55] Thomas Gottron et al. « Lodatio : using a schema-level index to support users infinding relevant sources of linked data ». Dans : *Proceedings of the seventh international conference on Knowledge capture*. ACM. 2013, p. 105–108.

[56] Gregory Grefenstette et Karima Rafes. « Transforming Wikipedia into an Ontology-based Information Retrieval Search Engine for Local Experts using a Third-Party Taxonomy ». Dans : *Joint Second Workshop on Language and Ontology & Terminology and Knowledge Structures (LangOnto2 + TermiKS) LO2TKS*. Portoroz, Slovenia, mai 2016.

[57] W3C Working Group et al. « Web services architecture ». Dans : *W3C* (2004).

[58] Marie Le Guilly, Jean-Marc Petit et Vasile-Marian Scuturici. « SQL Query Completion for Data Exploration ». Dans : *arXiv preprint arXiv :1802.02872* (2018).

[59] Aniko Hannak et al. « Measuring personalization of web search ». Dans : *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, p. 527–538.

[60] Sandro Hawke, D Beckett et J Broekstra. « SPARQL query results XML format ». Dans : *W3C Recommendation* (2013).

[61] Marcus Held et Joachim M Buhmann. « Unsupervised on-line learning of decision trees for hierarchical data analysis ». Dans : *Advances in neural information processing systems*. 1998, p. 514–520.

[62] Alan R Hevner. « A three cycle view of design science research ». Dans : *Scandinavian journal of information systems* 19.2 (2007), p. 4.

[63] Alan Hevner et Samir Chatterjee. « Design science research in information systems ». Dans : *Design research in information systems*. Springer, 2010, p. 9–22.

[64] Paul Hoffman, Andrew Sullivan et K Fujiwara. *DNS terminology*. Rapp. tech. 2015.

[65] Li Hong-qin et al. « Application Research of Read-write Linked Data Platform Based on Apache Marmotta ». Dans : *Library Theory and Practice* 7 (2017), p. 020.

[66] Stratos Idreos, Olga Papaemmanouil et Surajit Chaudhuri. « Overview of data exploration techniques ». Dans : *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, p. 277–281.

[67] INPI. *Le cahier de laboratoire national Comment l'utiliser ?* 2018.

[68] INPI. *Le cahier de laboratoire national Pourquoi l'utiliser ?* 2018.

[69] Ian Jacobs et Norman Walsh. « Architecture of the world wide web ». Dans : *W3C Recommendation* (2004).

[70] Magali Jaillard et al. « A Comprehensive Microbial Knowledge Base to Support the Development of In-vitro Diagnostic Solutions in Infectious Diseases. » Dans : *I-SEMANTICS (Posters & Demos)*. 2013, p. 55–59.

[71] Nandish Jayaram et al. « Querying knowledge graphs by example entity tuples ». Dans : *IEEE Transactions on Knowledge and Data Engineering* 27.10 (2015), p. 2797–2811.

[72] Jena. *Bulk loader with tdbloader2*.

[73] Jena Fuseki. *Serving RDF data over HTTP*.

[74] Apache JMeter. *Apache JMeter*. 2010.

[75] « JSON-LD 1.0 : a JSON-based serialization for linked data ». Dans : *W3C Recommendation* (2014).

[76] Simon Jupp et al. « The EBI RDF platform : linked open data for the life sciences ». Dans : *Bioinformatics* 30.9 (2014), p. 1338–1339.

[77] Karima Rafes. *SPARQL 1.1 : Test Suite Report with TFT*. 2018.

[78] Rafes Karima. *Sgvizler2, a Javascript wrapper for easy visualisation of SPARQL result sets*. 2018.

[79] Rafes Karima. *Lib PHP for SPARQL 1.1*. 2018.

[80] Kohsuke Kawaguchi et al. « Jenkins ». Dans : *Jenkins* ().

[81] Emilien Kenler et Federico Razzoli. « MariaDB Essentials ». Dans : Packt Publishing Ltd, 2015. Chap. 7, p. 141.

[82] Nodira Khoussainova et al. « SnipSuggest : Context-aware autocompletion for SQL ». Dans : *Proceedings of the VLDB Endowment* 4.1 (2010), p. 22–33.

[83] Graham Klyne, Jeremy Carroll et Brian McBride. « RDF 1.1 Concepts and Abstract Syntax ». Dans : *W3C Recommendation* (2014).

[84] Holger Knublauch et Kontokostas Dimitris. « Shapes Constraint Language (SHACL) ». Dans : *W3C Recommendation* (juil. 2017).

[85] Michael Kohlhase et al. « Knowledge-based interoperability for mathematical software systems ». Dans : *International Conference on Mathematical Aspects of Computer and Information Sciences*. Springer. 2017, p. 195–210.

[86] Kasjen Kramer, Renata Dividino et Gerd Gröner. « Space : Sparql index for efficient autocompletion ». Dans : *Proceedings of the 2013th International Conference on Posters & Demonstrations Track-Volume 1035*. CEUR-WS. org. 2013, p. 157–160.

[87] Markus Krötzsch, Denny Vrandečić et Max Völkel. « Semantic mediawiki ». Dans : *International semantic web conference*. Springer. 2006, p. 935–942.

[88] Michaël R Laurent et Tim J Vickers. « Seeking health information online : does Wikipedia matter ? » Dans : *Journal of the American Medical Informatics Association* 16.4 (2009), p. 471–479.

[89] Wangchao Le et al. « Scalable multi-query optimization for SPARQL ». Dans : *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE. 2012, p. 666–677.

[90] Michèle Leduc. « La fraude scientifique ». Dans : Institut des Hautes Etudes pour la Science et la Technologie, 2016.

[91] Jens Lehmann et al. « DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia ». Dans : *Semantic Web* 6.2 (2015), p. 167–195.

[92] Li Ma et al. *Towards a complete OWL ontology benchmark*. Springer, 2006.

[93] Fadi Maali, John Erickson et Phil Archer. « Data catalog vocabulary (DCAT) ». Dans : *W3C Recommendation* (2014).

[94] Bruno Menon. « Web sémantique et traitement automatique des langues ». Dans : *Congrès i-Expo 2004, Le web sémantique : théorie et mise en œuvre*. 2004.

[95] Albert Meroño-Peñuela et Rinke Hoekstra. « SPARQL2Git : transparent SPARQL and linked data API curation via Git ». Dans : *European Semantic Web Conference*. Springer. 2017, p. 143–148.

[96] Daniel Mietchen et al. *Enabling Open Science : Wikidata for Research*. 2015.

[97] NCBI. *PubMed*.

[98] Axel-Cyrille Ngonga Ngomo et al. « Sorry, i don't speak SPARQL : translating SPARQL queries into natural language ». Dans : *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, p. 977–988.

[99] Finn Årup Nielsen. « Wikipedia research and tools : Review and comments ». Dans : *Available at SSRN 2129874* (2012).

[100] Open Geospatial Consortium. *GeoSPARQL - A Geographic Query Language for RDF Data*. 2012.

[101] OpenLink. *iSPARQL*. 2011.

[102] OpenLink Virtuoso. *RDF Insert Methods in Virtuoso*.

[103] Bijan Parsia et al. « The OWL reasoner evaluation (ORE) 2015 competition report ». Dans : *Journal of Automated Reasoning* 59.4 (2017), p. 455–482.

[104] PiLOD. *Platform Linked Data Nederland*. 2015.

[105] Simone Paolo Ponzetto et Roberto Navigli. « Large-Scale Taxonomy Mapping for Restructuring and Integrating Wikipedia. » Dans : *IJCAI*. T. 9. 2009, p. 2083–2088.

[106] Karima Rafes. *Share, discover and reuse the Linked Data has never been easier with a LinkedWiki Platform.* 2018.

[107] Karima Rafes. *Repository Git of software TFT*. 2014.

[108] Karima Rafes. *Search articles with the ACM classification in Wikipedia*. 2017.

[109] Karima Rafes. *Display the artefacts of the research in Wikipedia*. 2015.

[110] Karima Rafes. *Demo : Display the artefacts of the research in Wikipedia*. 2016.

[111] Karima Rafes. *Demo : Display the laboratories of Inria in Wikipedia*. 2016.

[112] Karima Rafes. *Demo : Display the thesis of LRI in Wikipedia*. 2017.

[113] Karima Rafes. *Demo : Display scientists in Wikipedia*. 2017.

[114] Karima Rafes. *Creating synergies and interactions among the analysts and the producers of data in the University of Paris-Saclay*. 2018.

[115] Karima Rafes. *Mix'n'match with Wikidata : ACM Classification Code*. 2012.

[116] Karima Rafes. *Manual of LinkedWiki*. 2013.

[117] Karima Rafes, Sarah Cohen-Boulakia et Serge Abiteboul. « Designing scientific SPARQL queries using autocompletion by snippets ». Dans : *eScience, 2018 IEEE 14th International Conference on*. IEEE. 2018.

[118] Karima Rafes, Sarah Cohen-Boulakia et Serge Abiteboul. « Une autocomplétion générique de SPARQL dans un contexte multi-services ». Dans : *BDA*. 2017.

[119] Karima Rafes et Cécile Germain. « A platform for scientific data sharing ». Dans : *BDA2015-Bases de Données Avancées*. 2015.

[120] Karima Rafes, Julien Nauroy et Cécile Germain. « Certifying the interoperability of RDF database systems ». Dans : *LDQ 2015-2nd Workshop on Linked Data Quality*. 2. Springer. 2015.

[121] Karima Rafes, Julien Nauroy et Cécile Germain. « TFT, Tests For Triplestores ». Dans : *Semantic Web Challenge, part of the International Semantic Web Conference*. 2014.

[122] République Française. « LOI n° 2016-1321 pour une République numérique ». Dans : (oct. 2016).

[123] Laurens Rietveld et Rinke Hoekstra. « The YASGUI family of SPARQL clients ». Dans : *Semantic Web* 8.3 (2017), p. 373–383.

[124] Greg Robson. *Laboratory notebook skills*. 2014.

[125] Jean-François Rouet et André Tricot. « Chercher de l'information dans un hypertexte : vers un modèle des processus cognitifs ». Dans : *Les hypermédias, approches cognitives et ergonomiques* (1998), p. 57–74.

[126] Muhammad Saleem et al. « LSQ : the linked SPARQL queries dataset ». Dans : *International Semantic Web Conference*. Springer. 2015, p. 261–269.

[127] John Samuel. *WDProp : Wikidata's classes*. 2018.

[128] John Samuel. *WDProp : Wikidata's properties*. 2018.

[129] António Paulo Santos et Fátima Rodrigues. « Multi-label hierarchical text classification using the acm taxonomy ». Dans : *14th Portuguese Conference on Artificial Intelligence (EPIA)*. 2009, p. 553–564.

[130] Bahar Sateli et René Witte. « Personal Research Agents on the Web of Linked Open Data ». Dans : *International Conference on Language, Data and Knowledge*. Springer. 2017, p. 10–25.

[131] Leo Sauermann, Richard Cyganiak et M Völkel. « Cool URIs for the Semantic Web ». Dans : *World Wide Web Consortium (W3C)* (2008).

[132] Sergio M Savaresi et al. « Cluster selection in divisive clustering algorithms ». Dans : *Proceedings of the 2002 SIAM International Conference on Data Mining*. SIAM. 2002, p. 299–314.

[133] Max Schmachtenberg, Christian Bizer et Heiko Paulheim. « Adoption of the linked data best practices in different topical domains ». Dans : *International Semantic Web Conference*. Springer. 2014, p. 245–260.

[134] Nigel Shadbolt, Tim Berners-Lee et Wendy Hall. « The semantic web revisited ». Dans : *IEEE intelligent systems* 21.3 (2006), p. 96–101.

[135] Jon Siegel et Dan Frantz. *CORBA 3 fundamentals and programming*. T. 2. John Wiley & Sons New York, NY, USA : 2000.

[136] Martin G Skjæveland. « Sgvizler : A javascript wrapper for easy visualization of sparql result sets ». Dans : *Extended Semantic Web Conference*. Springer. 2012, p. 361–365.

[137] Steve Speicher, John Arwe et Ashok Malhotra. « Linked data platform 1.0 ». Dans : *W3C Recommendation* 26 (2015).

[138] Stardog. *Stardog documentation : Updating*.

[139] Mariela Tapia-Leon et al. « Application of ontologies in higher education : A systematic mapping study ». Dans : *Global Engineering Education Conference (EDUCON), 2018 IEEE*. IEEE. 2018, p. 1344–1353.

[140] Sana Tfaili et al. *Data acquisition for analytical platforms : Automating scientific workflows and building an open database platform for chemical analysis metadata*. Chimiométrie XVII. Poster. Jan. 2016.

[141] The W3C SPARQL Working Group. « SPARQL 1.1 (Protocol and RDF Query Language) ». Dans : *W3C recommendation* (mar. 2013).

[142] Gregory Todd Williams. « SPARQL 1.1 Service Description ». Dans : *W3C recommendation* (2013).

[143] Quoc Trung Tran, Chee-Yong Chan et Srinivasan Parthasarathy. « Query by output ». Dans : *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM. 2009, p. 535–548.

[144] TSO (The Stationery Office). *Flint SPARQL editor released into semantic web community*. 2011.

[145] Tania Tudorache et al. « WebProtégé : A collaborative ontology editor and knowledge acquisition tool for the web ». Dans : *Semantic web* 4.1 (2013), p. 89–99.

[146] European Union. *European Union Open Data Portal*. 2015.

[147] Pierre-Yves Vandenbussche et al. « Linked Open Vocabularies (LOV) : a gateway to reusable semantic vocabularies on the Web ». Dans : *Semantic Web* 8.3 (2017), p. 437–452.

[148] Pierre-Yves Vandenbussche et al. « SPARQLES : Monitoring public SPARQL endpoints ». Dans : *Semantic Web* 8.6 (2017), p. 1049–1065.

[149] Varnish. *Introduction to Varnish*. 2018.

[150] Openlink Virtuoso. *RDF Graphs Security with Virtuoso's database*. 2015.

[151] W3C. *EARL reports of SPARQL 1.1 Test Results for each editors*. 2013.

[152] W3C. *SPARQL1.1 : Test case structure*. 2012.

[153] W3C. *SPARQL1.1 : Test case structure*. 2016.

[154] W3C. « Linked data glossary ». Dans : (2013).

[155] W3C. *All recommendations of W3C*.

[156] W3C. *SPARQL 1.1 Test Results (official)*. 2013.

[157] W3C. *Official implementation report for SPARQL 1.1*. Mar. 2013.

[158] W3C SPARQL Working Group and The grid observatory. *Repository git TFT-tests with the test suite of SPARQL1.1 and Grid Observatory*. 2014.

[159] W3Techs. *Historical trends in the usage of server-side programming languages for websites*. 2018.

[160] Anita de Waard. « Research data management at Elsevier : Supporting networks of data and workflows ». Dans : *Information Services & Use* 36.1-2 (2016), p. 49–55.

[161] Timo Weithöner et al. « What's wrong with OWL benchmarks ». Dans : *Proc. of the Second Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*. Citeseer. 2006, p. 101–114.

[162] Wikidata. *Statistics*. 2018.

[163] Wikimedia. *Wikidata Query Service Metrics*. 2018.

[164] Wikimedia. *Wikidata Query*. 2018.

[165] Wikipedia. « Global user pages ». Dans : *Wikipedia, The Free Encyclopedia* (2015).

[166] Wikipedia. « Wikipedia, Help :Category ». Dans : *Wikipedia is an encyclopedia* (2018).

[167] Wikipedia. « Wikipedia is an encyclopedia ». Dans : *Wikipedia is an encyclopedia* (2018).

[168] Wikipedia. « Wikibase ». Dans : *Wikipedia, The Free Encyclopedia* (2015).

[169] Peng Yan et Wei Jin. « Building semantic kernels for cross-document knowledge discovery using Wikipedia ». Dans : *Knowledge and Information Systems* 51.1 (2017), p. 287–310.

[170] Nazar Zaki et Chandana Tennakoon. « BioCarian : search engine for exploratory searches in heterogeneous biological databases ». Dans : *BMC bioinformatics* 18.1 (2017), p. 435.

[171] Torsten Zesch et Iryna Gurevych. « Analysis of the Wikipedia category graph for NLP applications ». Dans : *Proceedings of the Second Workshop on TextGraphs : Graph-Based Algorithms for Natural Language Processing*. 2007, p. 1–8.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Le Linked Data à l'université : la plateforme LinkedWiki

Mots clés : Linked Data, LOD, SPARQL, VRE, Cloud, autocomplétion, Apprentissage non supervisé

Résumé : Le Center for Data Science de l'Université Paris-Saclay a déployé une plateforme compatible avec le Linked Data en 2016. Or, les chercheurs rencontrent face à ces technologies de nombreuses difficultés. Pour surmonter celles-ci, une approche et une plateforme appelée LinkedWiki, ont été conçues et expérimentées au-dessus du cloud de l'université (IAAS) pour permettre la création d'environnements virtuels de recherche (VRE) modulaires et compatibles avec le Linked Data. Nous avons ainsi pu proposer aux chercheurs une solution pour découvrir, produire et réutiliser les données de la recherche disponibles au sein du Linked Open Data, c'est-à-dire du système global d'information en train d'émerger à l'échelle du Web. Cette expérience nous a permis de montrer que l'utilisation opérationnelle du Linked Data au sein d'une université est parfaitement envisageable avec cette approche. Cependant, certains problèmes persistent, comme (i) le respect des protocoles du Linked Data et (ii) le manque d'outils adaptés pour interroger le Linked Open Data avec le langage SPARQL. Nous proposons des solutions à ces deux problèmes.

Afin de pouvoir vérifier le respect d'un protocole SPARQL au sein du Linked Data d'une université, nous avons créé l'indicateur SPARQL Score qui évalue la conformité des services SPARQL avant leur déploiement dans le système d'information de l'université.

De plus, pour aider les chercheurs à interroger le Linked Open Data, nous avons implémenté le démonstrateur SPARQLets-Finder qui facilite la conception de requêtes SPARQL à l'aide d'outils d'autocomplétion sans connaissance préalable des schémas RDF au sein du Linked Open Data.

Title : Linked Data at university : the LinkedWiki platform

Keywords : Linked Data, LOD, SPARQL, VRE, Cloud, autocompletion, unsupervised learning

Abstract : The Center for Data Science of the University of Paris-Saclay deployed a platform compatible with Linked Data in 2016. Because researchers face many difficulties utilizing these technologies, an approach and then a platform we call LinkedWiki were designed and tested over the university's cloud (IAAS) to enable the creation of modular virtual search environments (VREs) compatible with Linked Data. We are thus able to offer researchers a means to discover, produce and reuse the research data available within the Linked Open Data, i.e., the global information system emerging at the scale of the Internet. This experience enabled us to demonstrate that the operational use of Linked Data within a university is perfectly possible with this approach. However, some problems persist, such as (i) the respect of protocols and (ii) the lack of adapted tools to interrogate the Linked Open Data with SPARQL. We propose solutions to both problems.

In order to be able to verify the respect of a SPARQL protocol within the Linked Data of a university, we have created the SPARQL Score indicator which evaluates the compliance of the SPARQL services before their deployments in a universitys information system.

In addition, to help researchers interrogate the Linked Open Data, we implemented a SPARQLets-Finder, a demonstrator which shows that it is possible to facilitate the design of SPARQL queries using autocompletion tools without prior knowledge of the RDF schemas within the Linked Open Data.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France