AgroLD API

Une architecture orientée services pour l'extraction de connaissances dans la base de données liées AgroLD

Gildas Tagny Ngompé [1], **Aravind Venkatesan** [1], **Nordine El Hassouni** [1,2,4], **Manuel Ruiz** [1,2,4], **Pierre Larmande** [1,3,4]

1. Institut de Biologie Computationnelle (IBC)
Université de Montpellier, 860 rue St Priest, 34095 Montpellier Cedex 5, France
{aravindvenkatesan}{tagnyngompe}@gmail.com

2. AGAP, Plateforme SouthGreen, CIRAD
911 av. Agropolis, 34398 Montpellier, France
{manuel.ruiz}{nordine.el_hassouni}@cirad.fr

3. UMR DIADE, Plateforme SouthGreen, IRD
911 av. Agropolis, 34394 Montpellier, France
pierre.larmande@ird.fr

4. South Green Bioinformatics Platform, Montpellier, France

RÉSUMÉ. *L'agronomie est un domaine important constitué de divers domaines de recherche tels que la génétique, la biologie moléculaire des plantes, l'écologie et les sciences de la terre. Les dernières décennies ont vu le succès du développement de technologies à haut débit qui ont révolutionné et transformé la recherche agronomique. L'application de ces technologies a généré de grandes quantités de données et de ressources sur le web. Dans la plupart des cas, ces sources restent autonomes et déconnectées. Le projet Agronomic Linked Data (AgroLD) est une base de connaissances du web sémantique conçue pour intégrer des données provenant de diverses sources publiques de données centrées sur des plantes. L'objectif du projet AgroLD est de fournir un portail web pour les bioinformaticiens et les experts du domaine afin d'exploiter les données homogénéisées et permettre de connecter les connaissances dans ce domaine.*

ABSTRACT. *Agronomy is an overarching field constituting various research areas such as genetics, plant molecular biology, ecology and earth science. The last several decades has seen the successful development of high-throughput technologies that have revolutionised and transformed agronomic research. The application of these technologies have generated large quantities*

Ingénierie des systèmes d'information – n° 5-6/2016, 133-157

of data and resources over the web. In most cases these sources remain autonomous and disconnected. The Agronomic Linked Data project (AgroLD) is a semantic web knowledge base designed to integrate data from various publicly available plant centric data sources. The aim of AgroLD project is to provide a portal for bioinformaticians and domain experts to exploit the homogenized data towards enabling to bridge the knowledge.

MOTS-CLÉS : *biologie moléculaire, agronomie, web sémantique, données liées, RDF, SPARQL, services web RESTful*

KEYWORDS: *molecular biology, agronomy, semantic seb, linked data, RDF, SPARQL, RESTful web services*

DOI:10.3166/ISI.21.5-6.133-157

1. Introduction

L'agronomie est un champ multidisciplinaire qui comprend divers domaines de recherche tels que la biologie moléculaire végétale, la physiologie et l'agroécologie. La recherche agronomi0que vise à répondre entre autres à des questions sur l'amélioration de la production des cultures, leur résistance face aux maladies et l'étude de l'impact environnemental sur les cultures. Pour cela, les chercheurs ont besoin de comprendre en détail les implications des différents processus biologiques, et donc, de lier les données à différentes échelles (ex. la génomique, la protéomique et le phénome). Nous assistons actuellement à de rapides progrès dans les technologies de production de données biologiques conduisant à un flot d'information dans les domaines mentionnés ci-dessus. Cependant, une grande partie de cette information est dispersée dans différentes bases de données spécifiques à un domaine et, en outre, ces données sont mises à disposition sous divers formats. Par conséquent, l'utilisation efficace de ces ressources et l'adoption d'une approche intégrée reste un défi majeur.

Parmi les nombreux axes de recherche que compte la bioinformatique, la gestion des connaissances est devenue un domaine de recherche important (Goble, Stevens, 2008) axé sur l'interconnexion de l'information et la représentation des connaissances (Antezana *et al*., 2009). Dans ce domaine, les ontologies sont devenues une pierre angulaire dans la représentation des connaissances biologiques. Elles fournissent la structure nécessaire pour représenter des concepts biologiques et leurs relations. Aujourd'hui, de nombreuses applications exploitent les avantages offerts par les ontologies biologiques telles que Gene Ontology (GO) (Ashburner *et al*., 2000), Sequence Ontology (SO) (Mungall *et al*., 2011) et Plant Ontology (PO) (Cooper *et al*., 2013) pour n'en citer que quelques-unes.

De plus, la gestion efficace des connaissances exige l'adoption de méthodes robustes d'intégration de données. Cela implique une intégration efficace des sources de données hétérogènes distribuées, représentées dans un format interprétable par les machines. La technologie du web sémantique (WS) proposée par Berners-Lee *et al*. (2001), offre une solution pour faciliter cette intégration et permettre l'interopérabilité entre les machines.

Ces dernières années ont vu l'utilisation croissante du WS dans la communauté biomédicale, dont un certain nombre d'initiatives ont démontré le potentiel de ce dernier. Certaines initiatives notables incluent Bio2RDF (Belleau *et al.*, 2008), OpenPHACTS (Williams *et al.*, 2012), Life Data Linked (Momtchev *et al.*, 2009), KUPKB (Jupp *et al.*, 2011) et EBI RDF Platform (Jupp *et al.*, 2014). Pris dans son ensemble, une application WS mis en œuvre avec succès permet aux scientifiques de poser des questions très complexes par le biais d'une requête ou d'un ensemble de requêtes qui retourneront des réponses très pertinentes (Luciano *et al.*, 2011), ce qui facilite la formulation d'hypothèses de recherche (Venkatesan *et al.*, 2014).

Aujourd'hui, il y a une prise de conscience croissante similaire dans le domaine agronomique. Bien que les ontologies agronomiques soient majoritairement utilisées pour annoter les données contenues dans diverses bases de données, à la différence du domaine biomédical, l'utilisation du WS en agronomie doit encore être mieux exploitée. Afin de contribuer à cet effort, nous avons mis au point une base de connaissances sémantiques, nommée Agronomic Linked Data (AgroLD) (www.agrold.org). Le but de notre démarche est de fournir un portail d'information agronomique intégrée pour aider les experts du domaine à répondre aux questions biologiques pertinentes.

AgroLD est conçue pour intégrer des informations disponibles sur diverses espèces végétales. Le cadre conceptuel de la connaissance est basé sur des ontologies bien établies dans le domaine. En outre, compte tenu de la portée de l'effort, nous avons décidé de construire AgroLD en plusieurs phases. La phase actuelle (phase un) couvre les informations sur les gènes, les protéines, les prédictions de gènes homologues, les voies métaboliques, des phénotypes de plantes et le matériel génétique. Des informations sur les bases de données intégrées peut être trouvées sur la page de documentation d'AgroLD [1].

De manière standard, pour interroger une base de triplets RDF il faut faire exécuter des requêtes SPARQL par des points d'accès SPARQL (« sparql endpoint »). Les points d'accès SPARQL ont l'avantage de fournir une API standard. Même si le langage SPARQL est assez efficace pour construire les requêtes (maximise l'expressivité des requêtes (Ferré, 2014)), il reste difficile à prendre en main vue sa large variété de fonctionnalités.

Dans cet article, nous proposons un modèle d'architecture comprenant des systèmes implémentant des paradigmes des systèmes de recherche sémantique (Haag *et al.*, 2014). Chacun de ces paradigmes a des atouts, mais aussi des limites que leur intégration peut combler. Le reste du papier est organisé comme suit : dans la section 2, nous passons en revue les travaux connexes par rapport à notre domaine d'intérêt ; la section 3 présente le modèle d'architecture et les détails de son implémentation ; la section 4 examine des cas d'utilisation et leurs résultats et, la section 5 conclut sur la mise en place de l'architecture.

1. Documentation d'AgroLD : http://volvestre.cirad.fr:8080/agrold/documentation.jsp

2. Approches existantes d'interrogation des bases de données de triplets RDF

Dans cette section, nous proposons un état de l'art des approches nous semblant contribuer à l'interrogation des bases de données de triplets. Pour aider à la recherche sémantique, certains outils sont souvent proposés soit pour assister les utilisateurs dans la construction de leurs requêtes, soit pour cacher le langage SPARQL et fournir une interface plus facile à utiliser pour l'interrogation des bases cibles.

2.1. Approches d'aide à la construction de requêtes

2.1.1. Les éditeurs syntaxiques de requêtes SPARQL

Les fournisseurs de données RDF proposent une interface web pour l'édition et l'exécution de requêtes SPARQL sur leurs données. Les systèmes de gestions de graphes RDF tels que OpenLink Virtuoso proposent déjà un formulaire HTML simple pour indiquer les paramètres d'exécution au point d'accès SPARQL.

Certains publieurs de données améliorent l'utilisabilité de leurs clients SPARQL en intégrant des librairies d'édition de code (texte de la requête) et de gestion des résultats. Par exemple, UniProt [2] et Linked Life Data [3] joignent une grammaire de SPARQL à la librairie CodeMirror [4] pour proposer la coloration et vérification syntaxique, et l'auto-complétion. Parmi les librairies récentes, on retrouve YASQE, de la famille YASGUI (Rietveld, Hoekstra, 2015), qui est aussi basée sur CodeMirror mais propose un éditeur adapté à la syntaxe SPARQL. YASR (Rietveld, Hoekstra, 2015) associé à YASQE permet de gérer les messages d'erreur et d'afficher convenablement les résultats des requêtes. Par ailleurs, des interfaces fournissant plus de fonctionnalités, comme YASGUI [5] et Flint SPARQL Editor [6], sont disponibles en ligne pour interroger n'importe quel serveur SPARQL. YASGUI propose par exemple, l'interrogation simultanée de plusieurs serveurs SPARQL, la recherche de SPARQL endpoint, la conservation d'une requête entre deux sessions web.

2.1.2. Les langages visuels de requêtes

Il s'agit des systèmes permettant aux utilisateurs de soumettre des requêtes sous forme graphique. Leur but commun est d'augmenter l'intuitivité et l'utilisabilité du langage SPARQL. Leur principe est de construire les requêtes SPARQL en passant par des représentations graphiques. Parmi les systèmes les plus récents, nous pouvons citer ReVeaLD (Kamdar *et al*., 2014), QueryVOWL (Haag *et al*., 2015) et SPARQLGraph (Schweiger *et al*., 2014).

---

2. Point d'accès SPARQL d'UniProt : http://sparql.uniprot.org/
3. Point d'accès SPARQL de Linked Life Data : http://linkedlifedata.com/sparql
4. CodeMirror : https://codemirror.net/
5. YASGUI : http://yasgui.org/
6. Flint SPARQL Editor : http://openuplabs.tso.co.uk/demos/sparqleditor

ReVeaLD [7] est un système d'interrogation fédérée de données de recherches sur le cancer. Son principe est de partir d'un langage graphique, extensible, avec des concepts et liens spécifiques au domaine, pour permettre à l'utilisateur de représenter des questions complexes de son domaine. Un serveur SPARQL central reçoit donc la requête SPARQL construite, puis la traduit en plusieurs requêtes fédérées pour les différents serveurs de données.

QueryVOWL [8] quant à lui est défini par une représentation graphique des différents concepts du langage SPARQL (concepts, individus, disjonctions, filtres, etc.). Le graphe construit par l'utilisateur, dans ce cas, ne représente pas la requête mais un patron des données. Chaque nœud correspond à une requête SPARQL et renvoie ainsi une certaine information par rapport à cet unique nœud. Le serveur SPARQL est interrogé régulièrement pour assister l'utilisateur avec la liste des valeurs possibles pour un nœud, ou la description d'un individu par exemple. Ces systèmes ne couvrent pas toutes les fonctionnalités du langage SPARQL mais peuvent servir à la construction rapide du début d'une requête complexe. L'utilisateur pourrait affiner cette dernière par la suite dans un éditeur textuel. L'avantage de ReVeaLD par rapport à QueryVOWL, c'est qu'il propose plus de flexibilité dans la sélection des informations à retourner (QueryVOWL par exemple ne retourne pas des lignes de résultats où on peut voir la relation entre les différentes valeurs des variables de requête, mais uniquement le label du nœud sélectionné).

Comme ReVeaLD, SPARQLGraph [9] (Schweiger *et al.*, 2014), a été développé dans le contexte des sciences du vivant en prenant en compte leur spécificité. Sa particularité est de laisser le soin à l'utilisateur de choisir la source pour une certaine information, d'intégrer lui-même les différents services dans sa requête. A la différence de ReVeaLD qui passe par un médiateur de requête SPARQL, SPARQLGraph permet d'interroger plusieurs sources simultanément à partir d'une seule requête fédérée SPARQL. En plus, il permet un partage de requêtes visuelles et une collaboration entre les experts biologistes.

2.1.3. Interrogation basée sur les phrases en langage naturel

Les systèmes basés sur le langage naturel assistent l'utilisateur dans l'interrogation des bases de données en acceptant des questions en langage naturel. Ainsi, l'utilisateur n'a besoin de connaître que la terminologie des données, sans apprendre le langage SPARQL. Par exemple, dans le prototype actuel de LODQA [10] (Kim, Cohen, 2013), l'utilisateur fournit une question sur un domaine de connaissance préalablement sélectionné. La question passe ensuite par des étapes de traitement de langage naturel et de représentation intermédiaire. Les identifiants (URI) correspondant aux mots-clés de la question sont recherchés dans des sources ontologiques ou de données spécifiques. Si

7. ReVeaLD : http://srvgal78.deri.ie/revealD/
8. QueryVOWL : http://vowl.vsualdataweb.org/queryvowl/index.html
9. SPARQLGraph :http://sparqlgraph.i-med.ac.at
10. Linked Open Data Question AnWSering :http://lodqa.org/

des URI sont trouvés, le système génère enfin plusieurs requêtes SPARQL candidates pouvant correspondre à la question posée.

Si dans LODQA, il est possible d'entrer la requête librement, le système plus formel Sparklis [11] (Ferré, 2014) assiste de manière interactive l'utilisateur dans la construction de sa requête en langage naturelle. Le point d'accès SPARQL est interrogé à chaque étape pour orienter l'utilisateur avec des informations possibles pour compléter la requête. La requête SPARQL est construite en arrière-plan. La difficulté pour l'utilisateur est de retrouver à chaque étape l'information à sélectionner car la quantité proposée est souvent très grande. L'avantage de Sparklis est l'exploration à facettes personnalisée.

2.2. Approches de recherche d'informations

2.2.1. Exploration interactive

Pour rechercher de l'information, on a généralement recours à la navigation dans un ensemble de données. La recherche d'information est généralement basée sur des mots-clés et peut se faire suivant deux modes : les filtres et les facettes. Ces deux modes, par principe, analysent un grand ensemble de contenu et excluent les éléments qui ne respectent pas un certain critère (Whitenton, March 16, 2014). La navigation par facettes utilise plusieurs filtres pour donner différentes vues afin que l'utilisateur ait une meilleure compréhension des données. Toutefois, la réalisation des facettes demande plus d'effort que celle des filtres.

Le service web Virtuoso Facets facilite le développement d'application web sémantique de navigation à facettes. La requête et la réponse étant sous forme d'arbre XML, la partie de la requête SPARQL nécessaire à la navigation à facettes est mieux précisée (Erling, Mikhailov, 2009) que dans le cas d'une requête en simple texte (SPARQL).

Le système SPARQLFilterFlow [12] (Haag *et al.*, 2014) étend le modèle filtre/flux pour l'adapter au SPARQL. Il consiste à faire passer toutes les données de la source à travers un réseau de flux d'un nœud d'entrée vers un nœud de sortie. Chaque nœud représente un filtre ne laissant passer que les données respectant le critère correspondant. Dans ce système, chaque nœud peut avoir plusieurs points de connexions en entrée et en sortie.

2.2.2. API de type service web

Pour l'accès aux données, les API sous formes de services web sont généralement utilisées. Dans ce cas, les services basées sur les principes RESTful présentent plus d'avantages (figure 1 (Pautasso, 2014)) que les services dits WS-* (basés sur le protocole SOAP). REST (REpresentational State Transfer) (Fielding, Taylor, 2002) est un

11. Sparklis : http://www.irisa.fr/LIS/ferre/sparklis/osparklis.html
12. SPARQLFilterFlow : http://sparql.visualdataweb.org/

style d'architecture décrivant un ensemble de contraintes que doit respecter un service web. Le plus intéressant dans ce style est sa simplicité, la possibilité d'utiliser des technologies standards du web (HTTP, URL, etc.), et l'existance de plusieurs formats pour retourner les résultats des requêtes.

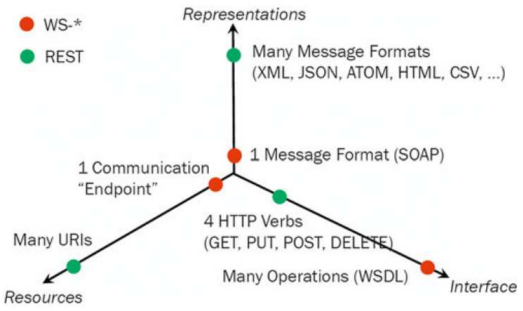


Figure 1. Avantages des services RESTful sur les services basées sur SOAP (WS-)*

Dans le domaine de l'intégration de données liées, certains projets ont démontré les avantages de fournir l'accès aux données *via des API de type RESTful*.

Le projet Open PHACTS Discovery Plateform (Groth *et al.*, 2014), par exemple, exploite les données liées pour fournir un accès intégré à des bases de connaissances pharmaceutiques à partir d'API de type RESTful. Les données de plus de 12 sources sont présentes dans le système Open PHACTS (Uniprot, ENZYME, DrugBank, etc.). Son architecture (Groth *et al.*, 2014) étend l'architecture standard des applications de données liées et ouvertes. Les modules d'accès aux données web, de mise en correspondance de vocabulaire, de résolution d'identité, et d'évaluation de qualité servent à la création de la base d'intégration de données web. Les autres modules relevant du fonctionnement de l'API sont :

– le médiateur qui est l'implémentation de l'API et donc effectue les requêtes SPARQL nécessaires vers la base de données et les services externes pour répondre aux requêtes venant des application tierces ;

– les définitions déclaratives qui décrivent précisément les liens entres les services et les données ;

– les documentations des services de l'API qui permettent aux développeurs d'applications tierces de mieux comprendre comment utiliser les services proposés par l'API, et même de les tester ;

– les librairies, spécifiques à un langage de programmation, réduisent l'effort de développement d'applications tierces ;

– les services externes sont les diverses sources de données interfacées par l'API ; ils sont interrogés pour enrichir l'information retournée.

Cette architecture facilite l'implémentation des services de l'API de manière itérative et incrémentale car chaque module a une fonction bien précise. De plus, Open

PHACTS Discovery Plateform dispose d'un contrôle d'accès à travers le gestionnaire d'API, 3scale [13], et d'un pilote pour l'intégration dans le système de workflow KNIME [14]. Le contrôle d'accès peut être important lorsqu'on veut gérer des aspects tels que des services payants, le suivi de l'utilisation de l'API, ou encore le respect de la politique d'exploitation (Groth *et al*., 2014).

SADI (*Semantic Automated Discovery and Integration*) (González *et al*., 2014) est un ensemble de principes d'implémentation et d'exposition des services web REST. A la différence d'Open Phacts Discovery Platform qui utilise les standards du protocole HTTP, SADI requiert un fichier RDF/XML, respectant une sémantique particulière, pour passer les arguments (*input*) aux services et, les résultats (*output*) sont aussi retournés uniquement en RDF/XML. Il expose les services web sur internet à partir du registre SHARE (*Semantic Health And Research Environment*). Les services ainsi exposés peuvent être découverts par des applications comme le plugin SADI-Galaxy (Aranguren *et al*., 2014) qui intègre les services SADI dans l'environnement de workflow Galaxy.

3. Développement de l'infrastructure d'accés aux données liées

3.1. Architecture proposée

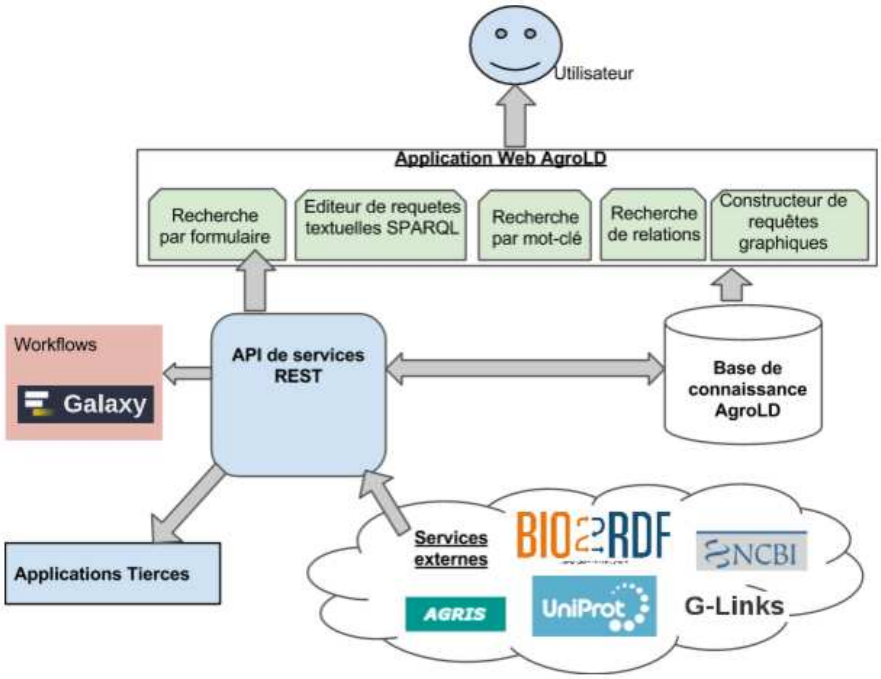
Les composants de notre architecture et leurs articulations sont présentés à la figure 2. L'architecture reprend le modèle classique 3-tiers avec l'application web, l'API de services web et la base de données liées.

Au niveau de l'application web, le module de recherche par mot-clé est destiné à explorer rapidement la base pour découvrir les données et ontologies stockées dans AgroLD à partir de mots-clés. Le constructeur de requêtes graphiques doit aider à la construction rapide de requêtes SPARQL à partir d'une représentation des termes des experts correspondant aux classes de la base. La recherche de relations permet de retrouver rapidement les relations existantes entre deux ou plusieurs entités.

L'API comprend les requêtes correspondant à des questions biologiques. Elle est utilisée lors de la recherche par formulaire et dans les workflows Galaxy. L'API peut interroger des services externes pour compléter les résultats trouvés dans AgroLD. La base de connaissance AgroLD peut ainsi être enrichie pendant l'exécution des requêtes.

Des applications tierces peuvent être développées à base des services de l'API. Notre cas d'étude ici est la recherche par formulaire qui utilise l'API. Nous proposons une recherche par formulaire basée sur le modèle itératif et exploratoire défendu par Uren *et al*. (2007) pour une meilleure utilisabilité. Il s'agit en effet d'une recherche qui propose à l'utilisateur de nouvelles voies de recherche en fonction des résultats des

13. 3scale : www.3scale.net
14. KNIME : www.knime.org/



(Les flèches indiquent le sens des données)

Figure 2. Architecture proposée pour l'application web d'AgroLD

recherches précédentes. L'utilisateur peut ainsi découvrir plus d'information relative à sa recherche.

3.2. Prototype implémenté

Parmi les systèmes d'interrogation décrits précédemment, certains peuvent être réutilisés pour accéder à AgroLD. Cependant, d'autres systèmes doivent être développés pour des besoins bien spécifiques. Afin de proposer une plateforme accessible aussi bien aux biologistes qu'aux bioinformaticiens, nous proposons d'intégrer plusieurs systèmes implémentant les différents paradigmes cités en section 2. Chaque utilisateur pourra ainsi utiliser les systèmes lui fournissant un bon compromis entre utilisabilité, expressivité, et convivialité des résultats.

3.2.1. Intégration et adaptation de systèmes existants

3.2.1.1. Système de recherche rapide

Pour ce système, nous pouvons intégrer le système de recherche à facettes, OpenLink Faceted, déjà disponible dans Openlink Virtuoso. Bien que les facettes présentées ici ne soient pas spécifiques aux concepts biologiques, Faceted permet de rapidement

retrouver toutes les entités ayant un attribut de type littéral (textuel) contenant l'expression recherchée.

3.2.1.2. Recherche de relations entre entités

La recherche des relations entre ressources peut être utile pour découvrir les liens existants entre des concepts ou entités biologiques. RelFinder (Heim *et al.*, 2009) présente l'avantage de donner une représentation graphique et de permettre une recherche rapide des entités par mot-clé lorsqu'on n'ignore leur URI. Cependant, la version originale de RelFinder a été développée (en ActionScript [15]) et configurée pour DBpedia. Nous avons proposé une configuration et une modification du système RelFinder qui soient adaptées à AgroLD. La configuration concerne principalement le point d'accès SPARQL, les propriétés à considérer pour la rechercher d'entités et pour la description des ressources.

Dans notre implémentation nous avons ajouté quelques exemples pour guider les utilisateurs. L'exécution du premier exemple permet de voir graphiquement les relations entre le gène *adenosylmethionine decarboxylase* et les deux *pathways spermine biosynthesis* et *spermidine biosynthesis* auxquels il participe (figure 14).

3.2.1.3. Système de requêtes visuelles

Nous proposons d'intégrer un système basé sur ReVeaLD (Kamdar *et al.*, 2014) mais avec un langage spécifique aux données d'AgroLD. Le langage est spécifié dans un fichier OWL en indiquant les différents concepts et propriétés des ressources d'AgroLD. ReVeaLD a ainsi besoin en entrée : des types de propriétés, des classes des sujets (*domain*) et des types possibles de valeurs (*range*) de ces propriétés. Le principal objectif est de fournir un moyen pour construire rapidement une requête SPARQL. ReVeaLD ne permet pas de faire toutes les interrogations possibles sur la base cible (par exemple, il est limité par la portée du langage qui le défini). La requête peut être visualisée et complétée dans l'éditeur textuel de requête SPARQL.

Ce système n'a pas pu être complètement développé pendant notre travail car nous ne disposions pas d'assez de temps pour définir le langage orienté utilisateur de concepts biologiques.

3.2.1.4. Dé-référencement d'URI

L'application Pubby [16], utilisée par DBpedia, nous a paru être une bonne solution pour répondre rapidement au problème de dé-référencement d'URI. Il s'agit en fait d'une application web Java, qui s'utilise dans le cas de point d'accès SPARQL supportant les requêtes DESCRIBE (de SPARQL). Nous l'avons choisie aussi parce que le serveur Virtuoso d'AgroLD supporte les requêtes DESCRIBE.

15. http://www.adobe.com/devnet/actionscript.html
16. http://wifo5-03.informatik.uni-mannheim.de/pubby/

Toutefois à l'avenir, nous souhaitons utiliser des systèmes plus récents comme TheDataTank (Vander Sande *et al*., 2012).

3.2.2. Développement de nouvelles fonctionnalités

Globalement, notre prototype est une application web développée en Java, HTML et JavaScript. Les systèmes que nous avons implémentés sont présentés dans cette section.

3.2.2.1. Editeur de requêtes SPARQL

Ce système est principalement destiné à l'usage des bioinformaticiens (développeurs d'application biologiques). Il comprend plusieurs fonctionnalités (figure 3) dont :

1) une zone d'édition des requêtes : en utilisant YASQE (Rietveld, Hoekstra, 2015), l'édition est facilitée avec les fonctionnalités d'auto-complétion, de vérification et coloration syntaxique, des raccourcis clavier, etc. ;

2) la sauvegarde des requêtes et résultats : les requêtes peuvent être sauvegardées dans des fichiers textes pour être utilisés plus tard. Les résultats quant à eux peuvent être téléchargés dans les formats RDF, CSV, JSON, etc. ;

3) une liste de patrons de requêtes correspondants à des questions biologiques ; ces patrons servent de bases à l'édition de requête ;

4) un composant de remplacement des paramètres de patrons de requêtes, pour utiliser un exemple avec différentes valeurs de paramètres ;

5) Une zone d'affichage des résultats utilisant YASR (Rietveld, Hoekstra, 2015) pour la bonne présentation des résultats et des messages d'erreurs.

3.2.2.2. API de services web

Architecture :

L'architecture globale de l'API de services web a été inspirée de celle d'Open PHACTS Discovery Plateform (Groth *et al*., 2014) car elle est modulaire et basée sur REST. Elle présente ainsi plus d'avantages que l'architecture SADI (González *et al*., 2014) au niveau des formats des résultats, de l'utilisation des standards, etc. Certains développeurs partagent leurs expériences de la conception d'API accessible en ligne, en publiant des principes et astuces à prendre en compte. Nous pouvons citer entre autres [17] :

- Gérer les versions de l'API ;
- Utiliser correctement les codes d'états de HTTP ;
- Utiliser les méthodes HTTP : GET, POST, PUT, DELETE ;

17. https://localize-software.phraseapp.com/posts/best-practice-10-design-tips-for-apis/

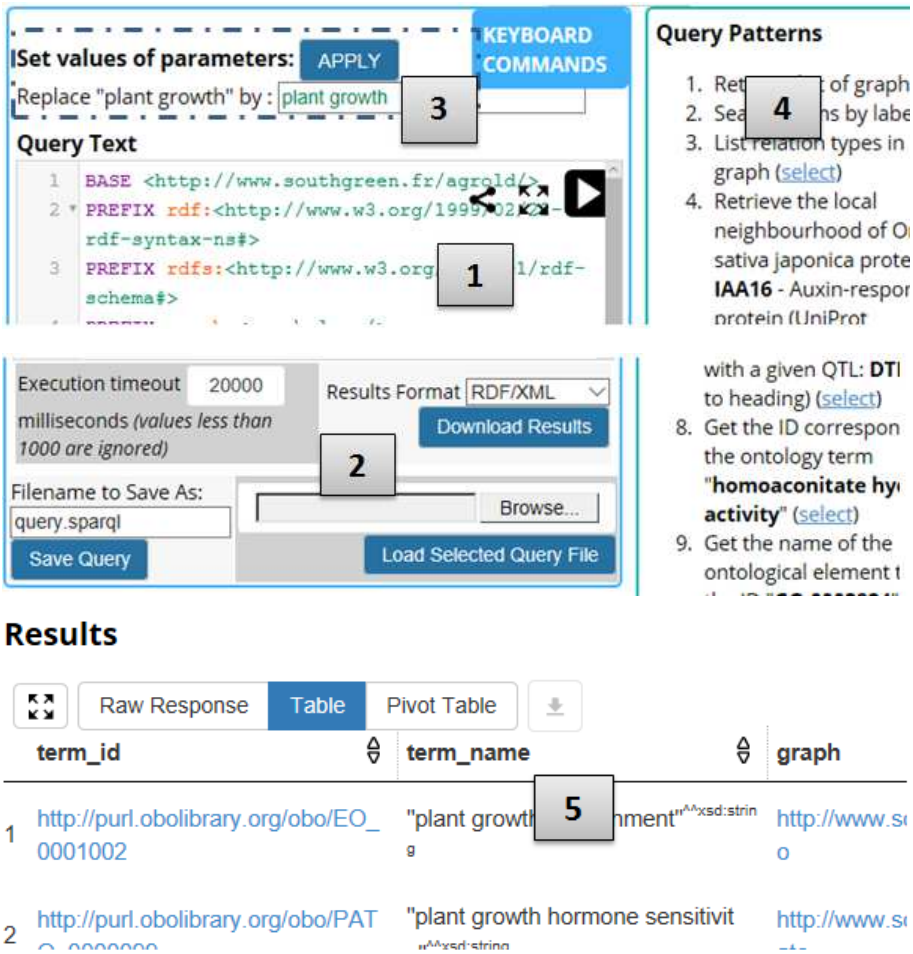


Figure 3. Editeur de requêtes textuelles SPARQL

– Renvoyer des messages d'erreurs compréhensibles et dans le même format que les résultats ;

– Retourner les résultats des requêtes en page pour éviter de lourds transfert de données sur le réseau ;

– Prévenir la surcharge du serveur en limitant le nombre d'accès simultanés ;

– Fournir une bonne documentation.

L'API que nous avons proposée comprend donc :

– **le module serveur** : correspondant au médiateur d'Open PHACTS, il implémente les services web. Il interroge le point d'accès SPARQL d'AgroLD et les services externes pour répondre aux requêtes. En plus de quelques services généraux

utiles, ce module est constitué en majeure partie des services correspondants à des questions biologiques (diagramme de classe à la figure 4). Il a été implémenté à l'aide de la librairie Jersey [18](version 2.17) parce qu'elle est *open source* et est destinée au développement de services REST en Java.

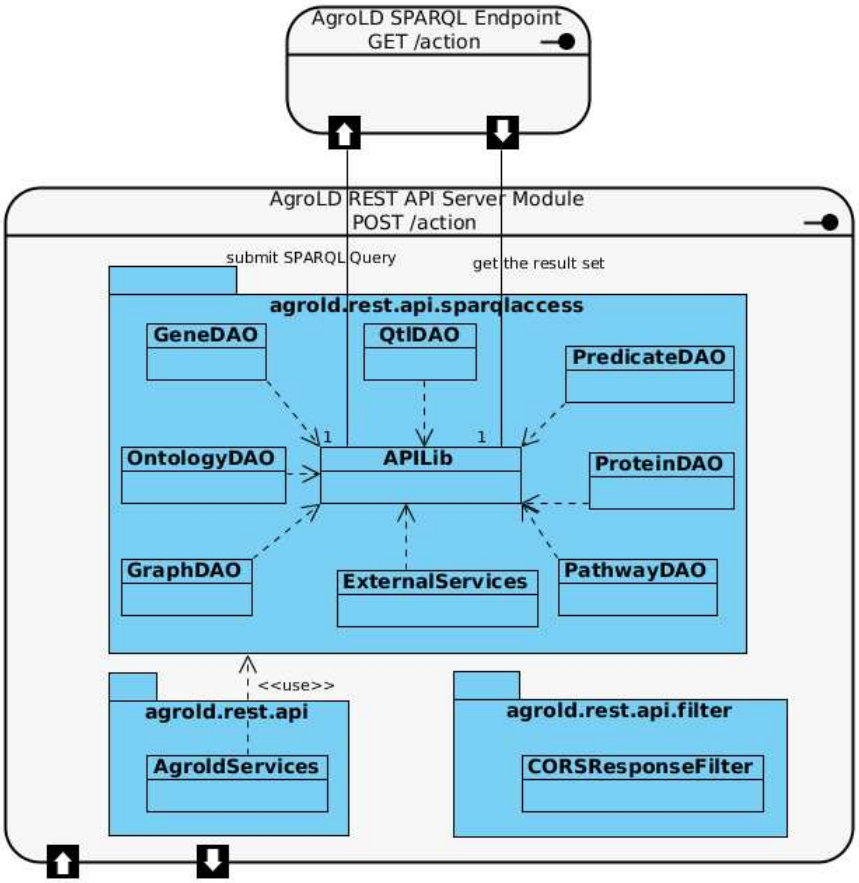


Figure 4. Module serveur de l'API d'AgroLD

– **la documentation interactive des services web** : elle donne une description complète pour comprendre et utiliser les services (rôle, paramètres, requête SPARQL exécutée, format de retour des résultats, signification des code HTTP, etc.). Son interativité permet aussi de tester les services et d'observer le comportement des services dans diverses situations. Les services sont décrits dans un fichier JSON [19]. Ce dernier

18. https://jersey.java.net/
19. Description de l'API d'AgroLD : http://volvestre.cirad.fr:8080/aldp/WSagger/agrold.json

est utilisé par le framework WSAGGER [20](2.0) pour générer la page web de documentation (figure 16).

– **Une librairie cliente pour application JavaScript** : elle est fournit par la librairie JavaScript WSagger-client.js (de WSAGGER) à partir de la description JSON des services web.

Services web disponibles :

Une trentaine de services sont actuellement accessibles à partir de l'API. Ils sont organisés en six catégories : *gene, graphs, ontologies, pathway, protein, QTL*. Nous présentons quelques exemples de services dans le tableau 1.

Tableau 1. Exemples de services web disponibles dans l'API d'AgroLD

L'URI de base est http://volvestre.cirad.fr:8080/agrold/api/1.0

URI	Information recherchée
/genes/publications/byId	des publications à partir de l'identifiant d'un gène
/genes/participatingInPathway	les gènes qui participent à un pathway dont l'identifiant est donné
/graphs/predicates	liste des URI des prédicats d'AgroLD
/ontologies/terms/children/byId	les descendant directs d'un concept ontologique étant donné son Id
/ontologies/terms/associatedWithQtl	les concepts ontologiques associés à un QTL donné et la nature de leur association
/proteins/EncodedByGene	les protéines encodées par un gène donné
/qtl/associatedWithProteinId	les QTL associés à une protéine donnée

3.2.2.3. Formulaire dynamique de recherche

Il implémente la recherche par formulaire de notre architecture. L'idée était de développer un formulaire dynamique basée sur l'API pour faire des recherches d'information dans AgroLD. Le système proposé permet en plus d'explorer les informations fournies par les services. A partir de la librairie cliente de l'API, nous l'avons implémenté en JavaScript et JSP pour cinq types d'entités : gène, protéine, QTL, *pathway*, classe d'ontologies. L'exploration est représentée par le diagramme d'activité de la figure 5. Par exemple, après avoir trouvé un QTL, on peut découvrir les informations relatives aux protéines ou concepts ontologiques auxquels il est associé.

3.2.2.4. Accès aux données à partir de Galaxy

Pour l'accès aux services web, Galaxy dispose du plugin *webservice_toolsuite* [21]. Ce dernier lit un fichier WADL (XML) décrivant les ressources RESTful disponibles et les rend accessibles dans Galaxy sous forme de sources de données. Ce plugin ne supporte que la méthode POST de HTTP. C'est la raison principale pour laquelle nous

20. http ://WSagger.io/

21. https://toolshed.g2.bx.psu.edu/view/ganjoo/webservice_toolsuite/d5cd409b8a18

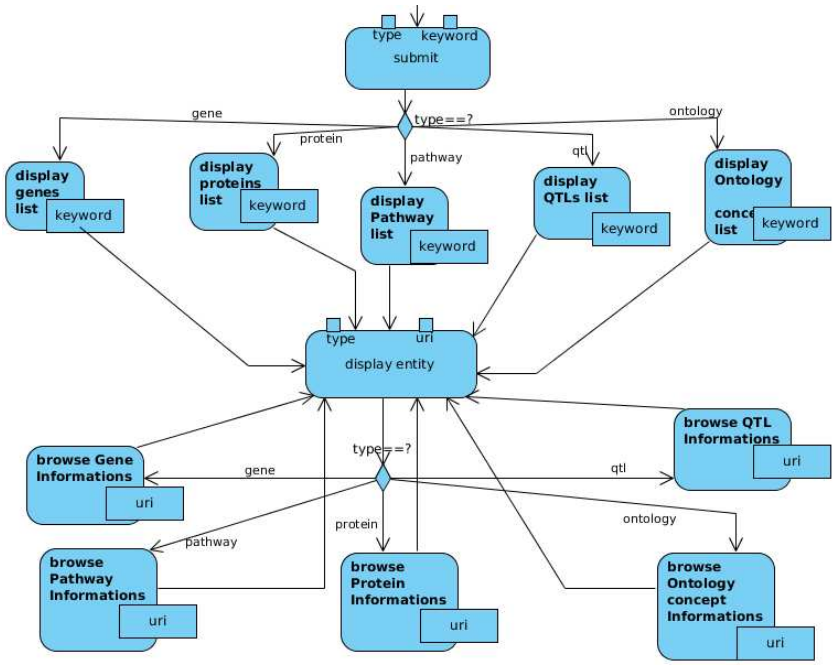


Figure 5. Activités de navigation avec le formulaire dynamique

avons défini la méthode POST pour les services de l'API au lieu de GET qui est plus indiqué lorsque les services font de la recherche d'information.

Nous avons décrit les services dans un fichier WADL [22]. Ensuite, tout utilisateur ayant installé l'outil *webservice_toolsuite* dans Galaxy peut passer par ses étapes d'intégration pour installer les ressources de l'API qu'il souhaite utiliser.

3.2.2.5. Enrichissement de l'information d'AgroLD

Dans notre travail, nous avons mis l'accent sur les résultats retournés à l'utilisateur. Nous proposons une intégration de services web au niveau du formulaire dynamique. L'avantage par rapport à l'intégration des données directement dans AgroLD est de pouvoir utiliser aussi bien des données en RDF que dans un autre format. Pour répondre aux questions nécessitant des données externes, nous avons enveloppé l'accès aux services externes dans des services de l'API (classe ExternalServices de la figure 4). Par exemple, pour la recherche de publications scientifiques relatives à une protéine, l'API passe par deux services :

1) G-Links [23] pour obtenir leur identifiant (résultats en JSON) ;

22. http://volvestre.cirad.fr:8080/agrold/agrold_api.wadl
23. http://www.g-language.org/wiki/glinks

2) puis Europe PubMed Central [24] pour obtenir leurs informations (en XML) : titre, auteurs, journal, année de publication, etc.

4. Utilisation de l'application web AgroLD

4.1. Utilisation d'AgroLD par des utilisateurs humains

Pour l'évaluation des systèmes de recherche sémantique, Elbedweihy *et al*. (2012) recommandent une enquête auprès des utilisateurs à partir de quelques requêtes à soumettre au système. Nous nous sommes basés sur cette approche pour intégrer à notre application un formulaire de recueil des avis, remarques et suggestions des utilisateurs. Le formulaire comprend quelques questions du modèle de questionnaires *System Usability Scale* (Brooke, 1996) et d'autres questions que nous avons jugées importantes.

Les trois principaux critères analysés sont :

1) l'utilisabilité qui concerne la facilité d'entrer la requête (nombre de tentatives, temps nécessaire) et, la présentation des résultats ;

2) l'expressivité qui définit pour un système de recherche quelles requêtes un utilisateur est capable de poser ;

3) la performance en rapport avec la capacité à retourner rapidement des résultats corrects et complets.

En attendant les réponses des utilisateurs qui guideront l'évolution de l'application, nous analysons ici quelques scénarios d'utilisation des fonctionnalités de recherche du prototype implémenté :

1) « recherche des entités liées au terme *plant height* » : c'est une requête générale où l'utilisateur n'a pas une idée précise du type d'entité qu'il recherche ;

2) « recherche des QTL associés à l'identifiant ontologique EO:0007403 » : c'est une requête plus précise mais plus complexe que la précédente ;

3) « recherche des publications relatives à la protéine TBP1 » : c'est une requête qui nécessite l'interrogation de services externes car la base AgroLD ne contient pas de publications ;

4) « recherche des relations existantes entre le gène AT3G25570 et les deux *pathways spermine biosynthesis* et *spermidine biosynthesis* ».

Nous analysons surtout les moyens les plus pratiques d'obtenir des résultats. Nous avons fait nos analyses avec l'instance en développement de notre prototype (http://volvestre.cirad.fr:8080/aldp/).

24. https://europepmc.org/RestfulwebService

4.1.1. Entrée des requêtes et expressivité

Pour le scénario 1, il est plus facile de soumettre la requête par la fonctionnalité *Quick Search* que par *Advanced Search* et *SPARQL Query*. Il suffit de saisir le mot-clé et de cliquer sur le bouton *Search* (figure 6).

Figure 6. Scénario 1 : entrée de la requête

Par contre, pour les requêtes plus complexes, la simple recherche par mot-clé *Quick Search* est moins facile à manipuler que les deux autres. Pour le scénario 2, l'absence de résultat par la recherche de EO:0007403 dans *Quick Search*, limite déjà l'utilisateur qui ne peut pas continuer la construction de sa requête. Avec *Advanced Search*, la construction de la requête se fera seulement en 2 étapes (figure 7) : « trouver le concept ontologique EO:0007403 » puis « accéder aux QTL associés ».

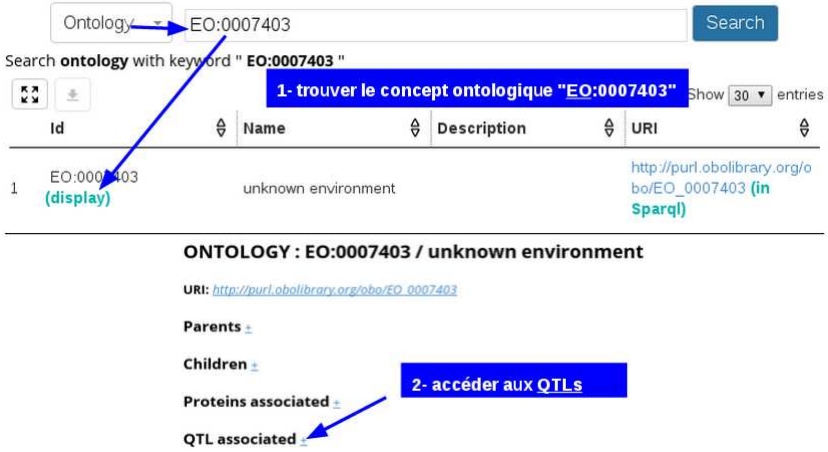


Figure 7. Scénario 2 : entrée de la requête dans le formulaire dynamique

Pour préciser les types d'informations qu'on souhaite avoir sur ces QTL, il serait préférable de passer par l'éditeur de requêtes SPARQL (figure 8). En plus cet éditeur peut être agrandi et il contient une assistance syntaxique (YASQUE (Rietveld, Hoekstra, 2015)) pour un meilleur confort dans la saisie.

Les informations sur les publications n'étant pas accessibles depuis un point d'accès SPARQL, nous ne pouvons pas utiliser de requêtes SPARQL fédérées dans le scénario 3. Par contre, avec *Advanced Search*, après avoir trouvé la protéine avec le mot-clé TBP1, l'utilisateur peut demander la recherche des publications concernant cette protéine (figure 9).

Dans le dernier scénario, il est possible d'utiliser une requête SPARQL, mais elle sera très difficile à construire. En utilisant la fonctionnalité de recherche de relations,

Figure 8. Scénario 2 : entrée de la requête dans l'éditeur de requête SPARQL

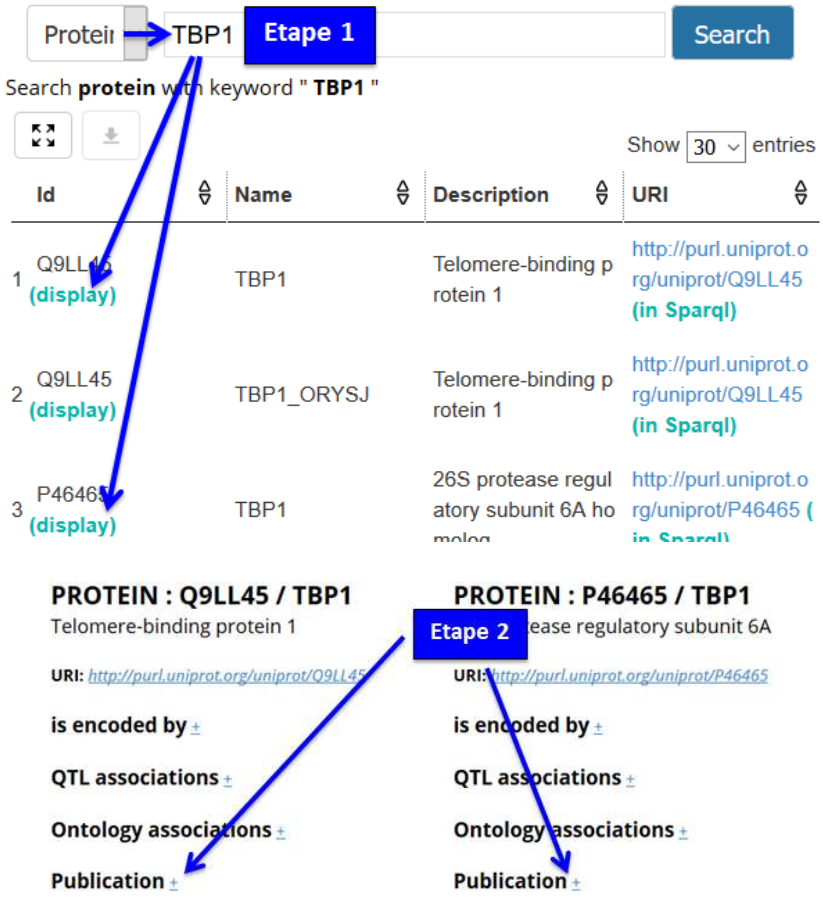


Figure 9. Scénario 3 : entrée de la requête dans le formulaire dynamique

l'utilisateur retrouve rapidement les ressources à partir des noms du gène et des *pathways*, et soumet sa requête (figure 10).

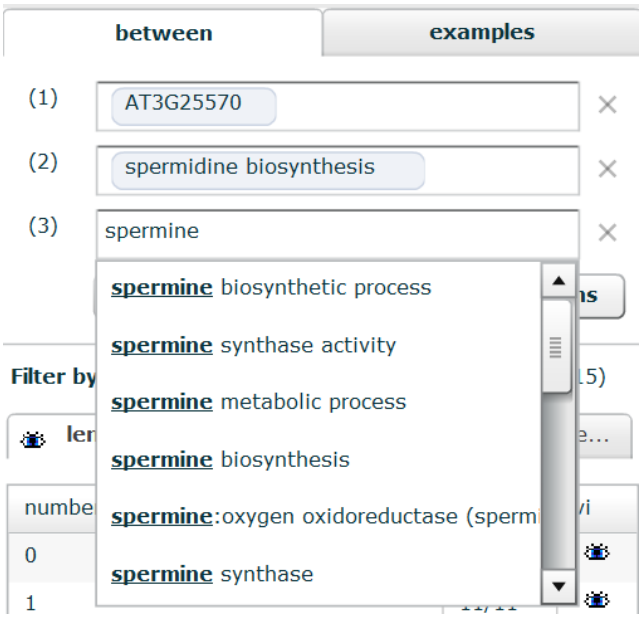


Figure 10. Scénario 4 : autocomplétion lors de l'entrée de la requête

4.1.2. Exécution des requêtes et temps de réponse

L'outil OpenLink Faceted utilisé par la fonctionnalité *Quick Search* retourne le temps de recherche effectué. Nous observons que la requête du scénario 1 s'exécute en 452 millisecondes ; un temps relativement proche de celui offert par les moteurs de recherche classiques. Par contre Faceted ne fournit pas de moyen d'effectuer la recherche sur l'aspect sémantique (par exemple rechercher uniquement les gènes).

La durée de l'exécution d'une requête par l'éditeur de requête SPARQL dépend des clauses de la requête et des performances du point d'accès SPARQL. Plus la requête sera complexe (nombre de clauses, type de filtres, etc.), plus ce temps sera long.

Nous avons noté que le temps d'exécution lors de la recherche par formulaire est de 18 secondes (13 puis 5 pour les 2 étapes) dans le scénario 2 (recherche dans la base AgroLD). Il est de 24 secondes (20 puis 4 pour les deux étapes) dans le scénario 3 (interrogation de services web externes). Ce qui indique que la recherche par mot-clé prend beaucoup plus de temps par rapport aux autres actions que peut effectuer l'utilisateur pendant son exploration.

Nous n'avons pas recueilli le temps d'exécution de la recherche de relation, mais elle semble aussi rapide que le *Quick Search*. Cependant, l'utilisateur doit patienter quelques secondes pendant l'affichage des relations qui se fait de manière séquentielle, les unes après les autres.

4.1.3. Présentation des résultats

Dans le scénario 1 avec *Quick Search*, nous observons une bonne présentation (figure 11) sous forme de tableau avec les mots-clés mis en gras pour indiquer où ils ont été retrouvés. Les liens hypertexts sur les URI permettent de continuer l'exploration des données.

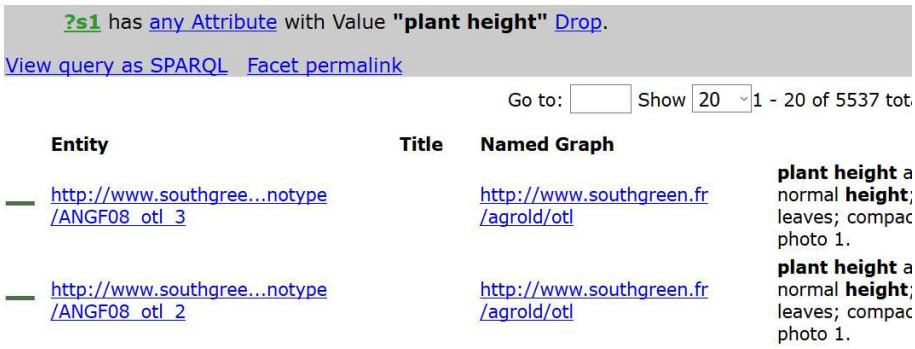


Figure 11. Scénario 1 : présentation des résultats de la recherche rapide par mot-clé

Par contre l'utilisateur rencontrera beaucoup de difficultés à parcourir tous les résultats retournés s'ils sont très nombreux (5537 pour le scénario 1).

Par le scénario 2, nous observons que ce problème peut être résolu en permettant à l'utilisateur de préciser le type d'entité qu'il recherche, mais en faisant un choix seulement parmi ceux qui sont proposés par la fonctionnalité *Advanced Search*. Les résultats obtenus ici sont plus précis (figure 12).

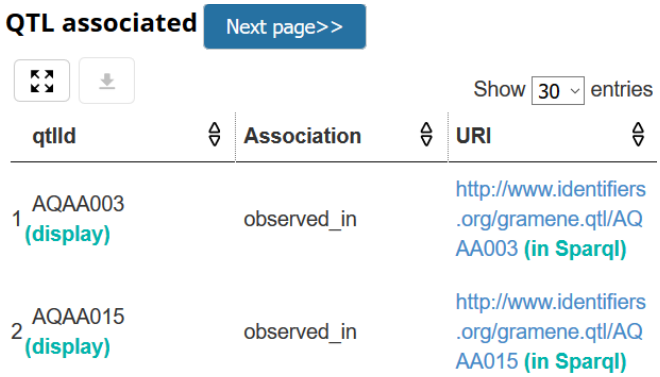


Figure 12. Présentation des résultats du scénario 2

La présentation des résultats dans *Advanced Search* et dans l'éditeur de requêtes SPARQL est aussi faite sous forme de tableaux pour une bonne lisibilité. Des liens y sont disponibles pour obtenir des informations complémentaires (liens *display* et URL dans la figure 12). *Advanced Search* propose aussi un lien pour une description de ressource dans l'éditeur SPARQL (lien *in Sparql* dans la figure 12).

Les résultats retournés dans le scénario 3 sont présentés dans un style proche des citations de documents (figure 13). Ainsi la reconnaissance des différentes informations relatives à la publication est intuitive. Un lien associé permet de se rendre à la source de ces informations pour plus de détails.

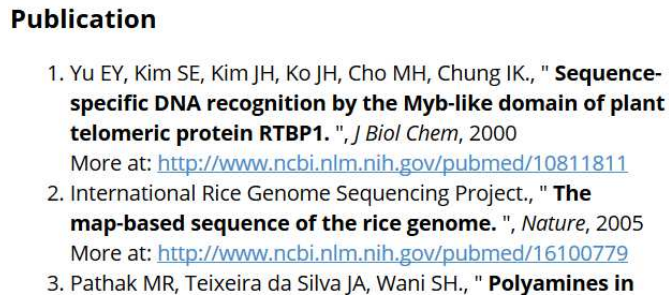


Figure 13. Présentation des résultats du scénario 3

Dans le dernier scénario, les relations sont observées sous forme graphique entre les nœuds d'un graphe pour une plus rapide lisibilité et compréhension. L'utilisateur peut filtrer les types de liens qu'il souhaite observer (par exemple, les liens *type* sont cachés dans la figure 14).

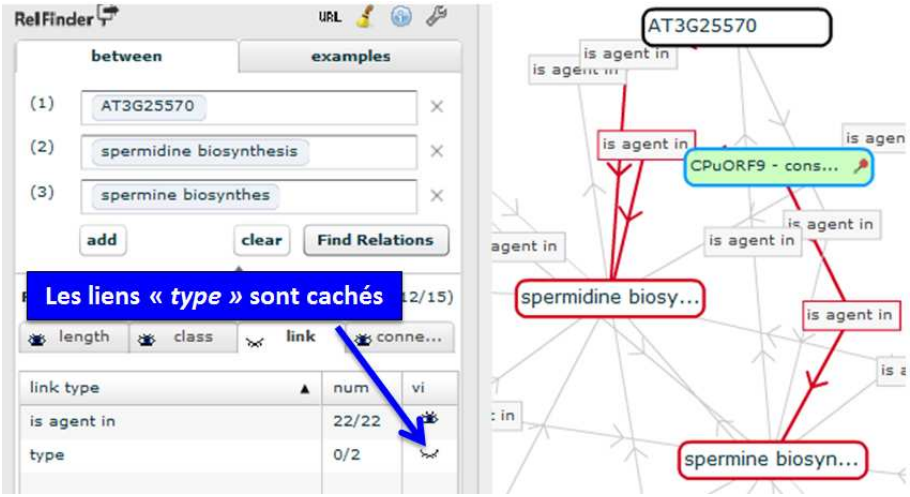


Figure 14. Scénario 4 : Relations découvertes entre le gène `adenosylmethionine decarboxylase` *(AT3G25570) et les deux pathways* `spermine biosynthesis` *et* `spermidine biosynthesis`

4.2. Utilisation d'AgroLD dans des applications

Le développement du formulaire dynamique est aussi l'occasion de démontrer l'utilisation en programmation de l'API d'AgroLD. Comme nous l'avons mentionné

précédemment, la description JSON des services de l'API peut être utilisée avec une librairie cliente de SWAGGER (par exemple *swagger-client.js* pour le JavaScript). Avec *swagger-client.js*, il suffit de préciser l'URL du fichier JSON, puis les services de l'API peuvent être appelés comme les méthodes d'un objet (*swagger.apis*) sans se préoccuper des URL des services ou de la méthode HTTP qu'ils supportent (figure 15).

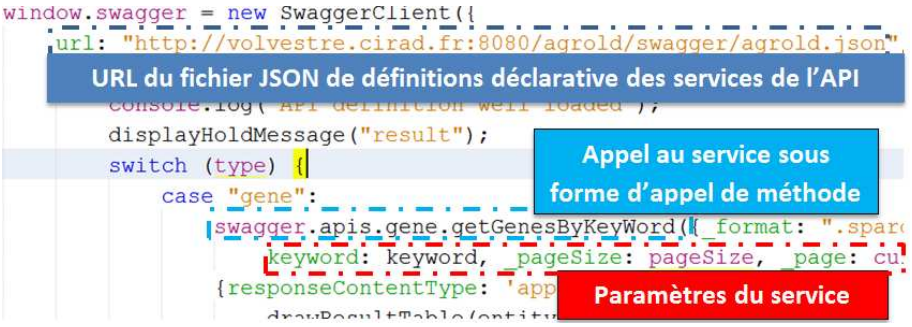


Figure 15. Programmation Javascript faisant appel au service de recherche de gène par mot-clé

En plus de cette facilité, la documentation interactive des services (figure 16) permet aux développeurs d'applications d'avoir toutes les informations sur chaque service pour savoir l'utiliser.

5. Conclusion

Parmi les nombreux axes de recherche que compte le domaine bioinformatique, la gestion de connaissance est devenue un domaine de recherche important, axé sur l'interconnexion de l'information et la représentation des connaissances. AgroLD est l'une des premières initiatives connues pour appliquer les pratiques du WS au domaine génétique et moléculaire en agromomie. Aujourd'hui, ce projet joue un rôle complémentaire des approches intégratives adoptées par cette communauté. Dans cet article, nous proposons un modèle d'architecture implémentant les paradigmes de recherche d'information sémantique (Haag *et al.*, 2014). Certains sont proposés soit pour assister les utilisateurs dans la construction de leurs requêtes, soit pour cacher le langage SPARQL et fournir une interface plus facile à utiliser pour l'interrogation des bases cibles. Chacun de ces paradigmes a des atouts, mais aussi des limites que leur intégration dans une seule infrastructure peut combler.

Dans de futures recherches, nous allons améliorer les performances et étendre les fonctionnalites de l'API de services web afin de couvrir d'autres entités représentées dans AgroLD (par exemple, l'annotation génomique et l'information d'homologie). Nous proposons également d'intégrer un système basé sur ReVeaLD (Kamdar *et al.*, 2014) mais avec un langage spécifique aux données d'AgroLD. Le principal objectif est de fournir aux utilisateurs un moyen pour construire rapidement une requête

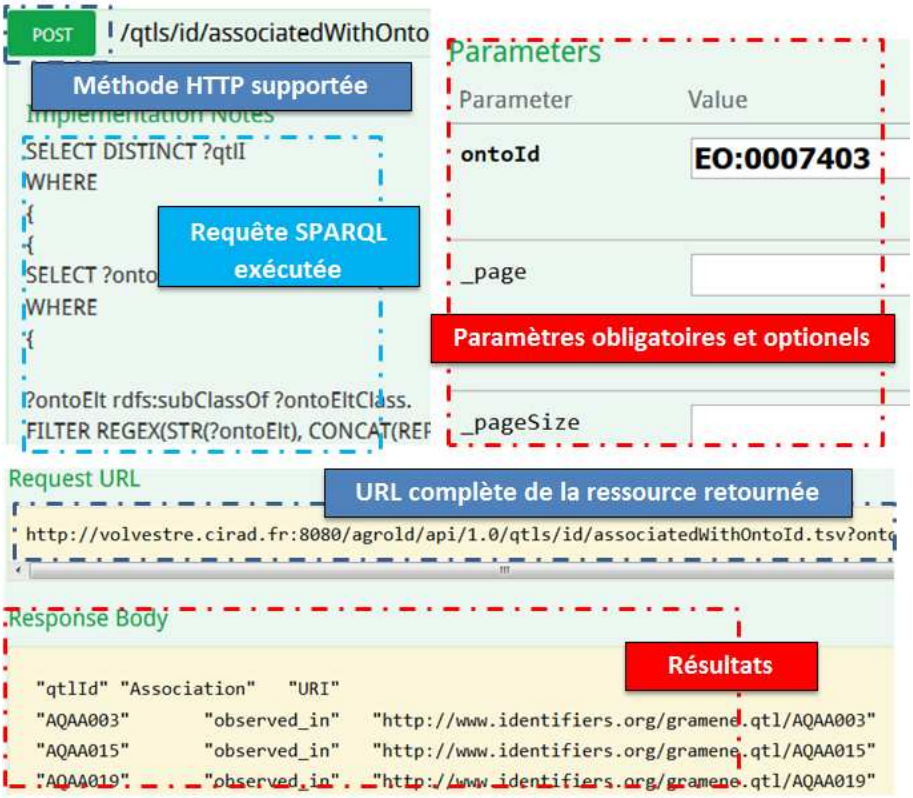


Figure 16. Documentation interactive du service de recherche des protéines associées à un identifiant ontologique

SPARQL. De plus, nous prévoyons d'explorer les fonctionnalités offertes par les outils d'indexation et de recherche dans le texte tels qu'ElasticSearch [25] et SolR [26]. Ceci permettra d'améliorer la fonctionnalité de recherche rapide en récupérant plus d'informations textuelles et ainsi de cacher les détails techniques.

Bibliographie

Antezana E., Kuiper M., Mironov V. (2009). Biological knowledge management: the emerging role of the Semantic Web technologies. *Briefings in Bioinformatics*, vol. 10, n° 4, p. 392 –407.

Aranguren M. E., González A. R., Wilkinson M. D. (2014). Executing SADI services in Galaxy. *Journal of biomedical semantics*, vol. 5, n° 1, p. 42.

25. ElasticSearch : https://www.elastic.co
26. SolR : https://lucene.apache.org/solr/

Ashburner M., Ball C. A., Blake J. A., Botstein D., Butler H., Cherry J. M. *et al*. (2000). Gene ontology: tool for the unification of biology. *Nature genetics*, vol. 25, n° 1, p. 25–29.

Belleau F., Nolin M.-A., Tourigny N., Rigault P., Morissette J. (2008). Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, vol. 41, n° 5, p. 706–716.

Berners-Lee T., Hendler J., Lassila O., Others. (2001). The semantic web. *Scientific american*, vol. 284, n° 5, p. 28–37.

Brooke J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, vol. 189, n° 194, p. 4–7.

Cooper L., Walls R. L., Elser J., Gandolfo M. A., Stevenson D. W., Smith B. *et al*. (2013). The plant ontology as a tool for comparative plant anatomy and genomic analyses. *Plant & cell physiology*, vol. 54, n° 2, p. e1.

Elbedweihy K., Wrigley S. N., Ciravegna F., Reinhard D., Bernstein A. (2012). Evaluating semantic search systems to identify future directions of research. In *The semantic web: Eswc 2012 satellite events*, p. 148–162. Springer.

Erling O., Mikhailov I. (2009). Faceted Views over Large-Scale Linked Data. In *Ldow*.

Ferré S. (2014). Expressive and scalable query-based faceted search over sparql endpoints. In *International semantic web conference*, p. 438–453.

Fielding R. T., Taylor R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, vol. 2, n° 2, p. 115–150.

Goble C., Stevens R. (2008). State of the nation in data integration for bioinformatics. *Journal of biomedical informatics*, vol. 41, n° 5, p. 687–93.

González A., Callahan A., Cruz-Toledo J., Garcia A., Egaña Aranguren M., Dumontier M. *et al*. (2014). Automatically exposing OpenLifeData via SADI semantic Web Services. *J Biomed Semantics*, vol. 5, n° 1, p. 46.

Groth P., Loizou A., Gray A. J., Goble C., Harland L., Pettifer S. (2014). API-centric Linked Data integration: The Open PHACTS Discovery Platform case study. *Web Semantics: Science, Services and Agents on the World Wide Web*, p. 1–7.

Haag F., Lohmann S., Bold S., Ertl T. (2014). Visual SPARQL Querying based on Extended Filter/Flow Graphs. In *Proceedings of the 12th international working conference on advanced visual interfaces (avi '14)*, p. 305–312. New York, NY, USA: ACM.

Haag F., Lohmann S., Siek S., Ertl T. (2015). Visual Querying of Linked Data with {QueryVOWL}. In *Joint proceedings of sumpre 2015 and hswi 2014-15*. CEUR-WS.

Heim P., Hellmann S., Lehmann J., Lohmann S., Stegemann T. (2009). RelFinder: Revealing relationships in RDF knowledge bases. In *Semantic multimedia*, p. 182–187. Springer.

Jupp S., Klein J., Schanstra J., Stevens R. (2011). Developing a kidney and urinary pathway knowledge base. *Journal of biomedical semantics*, vol. 2, n° 2, p. 1.

Jupp S., Malone J., Bolleman J., Brandizi M., Davies M., Garcia L. *et al*. (2014). The EBI RDF platform: linked open data for the life sciences. *Bioinformatics*, vol. 30, n° 9, p. 1338–1339.

Kamdar M. R., Zeginis D., Hasnain A., Decker S., Deus H. F. (2014). ReVeaLD: A user-driven domain-specific interactive search platform for biomedical research. *Journal of Biomedical Informatics*, vol. 47, p. 112–130.

Kim J.-D., Cohen K. B. (2013). Natural language query processing for SPARQL generation: A prototype system for SNOMED CT. In *Proceedings of biolink*, p. 32–38.

Luciano J. S., Andersson B., Batchelor C., Bodenreider O., Clark T., Denney C. K. *et al*. (2011). The translational medicine ontology and knowledge base: driving personalized medicine by bridging the gap between bench and bedside. *Journal of biomedical semantics*, vol. 2, n° 2, p. 1.

Momtchev V., Peychev D., Primov T., Georgiev G. (2009). Expanding the Pathway and Interaction Knowledge in Linked Life Data. In *International semantic web challenge.*

Mungall C. J., Batchelor C., Eilbeck K. (2011). Evolution of the Sequence Ontology terms and relationships. *Journal of biomedical informatics*, vol. 44, n° 1, p. 87–93.

Pautasso C. (2014). RESTful Web Services: Principles, Patterns, Emerging Technologies. *Web Services Foundations, Springer Science+Business Media New York*, p. 31–51.

Rietveld L., Hoekstra R. (2015). The YASGUI Family of SPARQL Clients. *Semantic Web Journal*.

Schweiger D., Trajanoski Z., Pabinger S. (2014). SPARQLGraph: a web-based platform for graphically querying biological Semantic Web databases. *BMC bioinformatics*, vol. 15, n° 1, p. 279.

Uren V., Lei Y., Lopez V., Liu H., Motta E., Giordanino M. (2007). The usability of semantic search tools: a review. *The Knowledge Engineering Review*, vol. 22, n° 04, p. 361–377.

Vander Sande M., Colpaert P., Van Deursen D., Mannens E., Walle R. de. (2012). The datatank: an open data adapter with semantic output. In *21st international conference on world wide web, proceedings*, p. 4.

Venkatesan A., Tripathi S., Sanz de Galdeano A., Blondé W., Lægreid A., Mironov V. *et al*. (2014). Finding gene regulatory network candidates using the gene expression knowledge base. *BMC bioinformatics*, vol. 15, n° 1, p. 386.

Whitenton K. (March 16, 2014). *Filters vs. Facets: Definitions.* Consulté sur https://www.nngroup.com/articles/filters-vs-facets/

Williams A. J., Harland L., Groth P., Pettifer S., Chichester C., Willighagen E. L. *et al*. (2012). Open phacts: semantic interoperability for drug discovery. *Drug discovery today*, vol. 17, n° 21, p. 1188–1198.