# Vers un système intelligent de capitalisation de connaissances pour l'agriculture durable : construction d'ontologies agricoles par transformation de sources existantes

Fabien Amarger


# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :** *l'Université Toulouse - Jean Jaurès*

---

**Présentée et soutenue le** *18 Décembre 2015* **par :**

**Fabien Amarger**

**Vers un système intelligent de capitalisation de connaissances pour l'agriculture durable : Construction d'ontologie agricoles par transformation de sources existantes**

---

**JURY**

| | | |
|---|---|---|
| Marie-Hélène Abel | Heudiasyc | Examinatrice |
| Jean-Pierre Chanet | Irstea | Encadrant |
| Jean-Pierre Chevalet | LIG | Examinateur |
| Juliette Dibie-B. | AgroParisTech | Rapporteur |
| Ollivier Haemmerlé | IRIT | Directeur |
| Nathalie Hernandez | IRIT | Encadrante |
| Chantal Reynaud | LRI | Rapporteur |
| Catherine Roussey | Irstea | Encadrante |

---

**École doctorale et spécialité :**
*MITT : Domaine STIC : Intelligence Artificielle*
**Unité de Recherche :**
*IRIT / Irstea*
**Directeur de Thèse :**
*Ollivier Haemmerlé*
**Rapporteurs :**
*Juliette Dibie-Barthelemy* et *Chantal Reynaud*

## Résumé

Les données disponibles sur le Web sont généralement de deux natures : (1) des données non structurées ou semi-structurées difficilement exploitables de manière automatique ou (2) des données structurées destinées à une utilisation particulière, difficilement réutilisables par d'autres applications. Le Web de données est une application du Web sémantique facilitant l'accès, le partage et l'alignement des données. Il existe actuellement de très nombreuses données disponibles sur le Web, mais qui ne sont pas publiées en suivant les principes du Web de données liées. Elles nécessiteraient d'être transformées en bases de connaissances. Nous proposons une méthodologie innovante qui permet de transformer plusieurs sources simultanément et non séquentiellement. Cette méthodologie permet la fusion de plusieurs sources de données orientée par des patrons de conception du domaine. Notre méthodologie spécifie la modélisation attendue du domaine en définissant la partie haute d'un module ontologique. Une chaîne de processus enrichit ce module par des éléments issus des sources : transformation syntaxique des sources, alignement, identification des éléments équivalents pour construire des candidats, calcul de score de confiance des candidats, filtrage des candidats. Notre travail part de l'hypothèse suivante : si un élément apparaît dans plusieurs sources, alors la possibilité qu'il appartienne au domaine d'étude est accrue. Nous avons défini différentes fonctions de calcul de la confiance consensuelle d'un candidat en mettant en évidence plusieurs caractéristiques comme le consensus entre sources ou la connectivité entre éléments d'un même candidat. Nous posons une deuxième hypothèse : un élément ne doit apparaître que dans un seul candidat pour obtenir une modélisation correcte. Cette hypothèse nous amène à définir la notion d'incompatibilité entre candidats. Nous pouvons considérer alors l'extraction des candidats qui ne partagent pas d'éléments, ce qui permet de faciliter le travail de validation. Pour évaluer nos propositions, nous avons mené trois expérimentations. La première a porté sur le domaine de la classification taxonomique des blés. Cette expérimentation nous a permis d'analyser la qualité des candidats générés avec l'aide de trois experts du domaine. La deuxième expérimentation a porté sur le même domaine et nous a permis de valider le temps gagné par un expert lors de la validation des candidats en considérant les incompatibilités. Pour la dernière expérimentation nous avons utilisé les données d'une campagne d'évaluation de systèmes d'alignements. Nous avons adaptés ces données pour évaluer la génération de candidats et la définition du score de confiance sur un grand jeu de données. Nous proposons une implémentation de cette proposition dans un outil réutilisable et paramétrable : Muskca. Celui-ci permet la fusion multi-sources pour la génération d'une base de connaissances consensuelle. L'application de nos travaux dans le domaine de l'agriculture nous a permis de constituer une base de connaissances sur la taxonomie des plantes. Cette base de connaissances permettra la représentation d'observations des attaques des agresseurs sur les cultures, ainsi que les techniques de traitement des agresseurs. Cette base de connaissances permettra de publier les données disponibles mais aussi d'annoter les nombreux documents mobilisables pour faire évoluer les pratiques agricoles.

## Abstract

The data available on the Web are generally of two kinds : (1) non structured data or semi structured data, which are difficult to exploit automatically ; or (2) structured data, dedicated to a specific usage, which are difficult to reuse for a different application. The Linked Open Data is a Semantic Web application facilitating access, shareability and alignment of data. There are many data available on the Web, but these are not always published using the Linked Open Data theory and thus need to be transformed into knowledge bases. An innovative methodology is proposed in this work : one that transforms several sources simultaneously, not sequentially. This methodology merges several data sources oriented by domain design patterns and defines the expected domain representation using the upper part of an ontological module. A process chain enriches this module with elements from the sources : syntactic transformation of the sources, alignment, identification of equivalent elements for the construction of candidates, computation of the candidates' trust scores and candidate filtering. This work is based on the following hypothesis : if an element appears in several sources then the possibility that it belongs to the studied domain is increased. Several functions were defined in order to compute the consensual trust score of a specific candidate by bringing out such characteristics as the consensus between the sources or the connectivity between the elements within a given candidate. A second hypothesis is put forward : to obtain a valid design, an element must be part of one candidate only. This hypothesis resulted in the definition of the notion of incompatibility between the candidates. The extraction of the candidates that do not share elements can then be considered, which made the experts' validation task easier. To evaluate the proposals, three experiments were conducted. The first one dealt with the taxonomic classification of wheat. With the assistance of three experts, this experiment made for the analysis of the validation of the generated candidates. The second experiment, still in the same domain, lead to the evalution of the time an expert saved using the notion of incompatibility during the validation of the candidates. As for the last experiment, the data from an evaluation campaign of alignment systems were used. These data had to be adapted to evaluate the generation of the candidates and the definition of the consensual trust score on a large data set. These three proposals were implemented in a new reusable and configurable tool : Muskca. This tool allows a multi-source fusion for the generation of a consensual knowledge base. This methodology was applied to agriculture, which allowed the creation of a knowledge base on plant taxonomy. The knowledge base will be used to represent the observations of pest attacks on crops along with pest treatment techniques. Not only will this knowledge base help the publication of the available data but it will also allow the annotation of the various documents that will be used, so as to improve agricultural practices.

## Remerciements

Je tient tout d'abord à remercier Jean-Pierre Chanet, Ollivier Haemmerlé, Nathalie Hernandez et Catherine Roussey pour m'avoir encadré et soutenu durant ces trois années de thèse. Leur complémentarité a été une grande force qui m'a permis de découvrir les différents aspects du monde de la recherche et développer ma passion pour l'informatique.

Je souhaite témoigner toute ma gratitude à Juliette Dibie-Barthelemy et Chantal Reynaud pour avoir accepté de rapporter sur mon manuscrit. Ces rapports m'ont été particulièrement précieux pour déterminer les forces et les faiblesses du-dit manuscrit afin d'en améliorer la qualité. Je remercie également Marie-Hélène Abel et Jean-Pierre Chevalet pour leur participation à mon jury de thèse.

Je remercie l'équipe COPAIN de l'Irstea à Clermont-Ferrand qui m'a accueilli durant ma première année. Je remercie également l'équipe MELODI de l'IRIT qui m'a accueilli durant les deux années suivantes. Ces équipes m'ont, toutes deux, permis d'intégrer deux dynamiques de recherches différentes et leurs conseils m'ont été d'une grande aide.

Je souhaite également remercier Franck Jabot, Jacques Le Gouis et Vincent Soulignac qui sont les experts qui m'ont permis d'obtenir les données de références pour les expérimentations. Cette aide m'a été très précieuse car ces expérimentations ont permis de valider mon approche.

Je tient aussi à remercier la société Arvalis qui m'a permis de travailler sur leur base de données sur les observations d'attaques agricoles en France. Cette base de données m'a permis de mettre en évidence les différents besoins en agricultures en terme de données.

Je remercie les doctorants de l'équipe COPAIN et de l'équipe MELODI pour tous les moments de soutient, discussions plus ou moins scientifiques et surtout pour leurs bonnes humeurs et sympathie.

Je remercie également les représentants des doctorants à la commission recherche de l'UT2J pour tout le travail qui a été fourni et qui le sera encore dans le futur pour l'amélioration de la vie des doctorants.

Je souhaite aussi témoigner toute ma gratitude envers les différentes personnes qui m'ont aidé scientifiquement à l'élaboration de ma thèse. Nicolas Seydoux pour tous les graphes dessinés sur le tableau blanc et le développement d'une librairie. Elodie Thieblin pour son aide précieuse pour la mise en place du cadre expérimental pour mes expérimentations. Romain Guillaume pour ses conseils éclairés et explications tout aussi claires concernant l'intégrale de Choquet.

Je remercie également ma famille pour leur soutient sans faille depuis le début de mes études. Ils m'ont donné l'opportunité et la motivation de m'investir au maximum dans ma passion pour l'informatique.

Je remercie particulièrement mon père, Eric Amarger, qui m'a inculqué cette passion pour l'informatique qui me permet aujourd'hui d'écrire ce manuscrit. Je remercie aussi Ida Bäckstedt pour son précieux soutient quotidien durant ces trois années. Enfin je remercie ma grand-mère, Simone Eymard, pour toutes les valeurs qu'elle m'a insufflées, notamment la persévérance et l'abnégation parmi tant d'autres. Je souhaite lui dédier ce manuscrit pour lui témoigner toute ma gratitude.

# Table des matières

# Introduction

Le Web permet la publications en ligne de documents qui deviennent alors accessibles pour toutes les personnes disposant d'une connexion internet.

La technologie mise en place pour le déploiement du Web permet un partage de documents et de toutes les informations qu'ils comportent. Ces dernières années, le Web a singulièrement évolué, avec l'apparition d'applications en plus de simples documents statiques. Ces applications permettent, entre autre, d'accéder à des données structurées par l'intermédiaire de formulaires. Plus récemment, de nombreuses initiatives ont mené à la publication de données structurées disponibles sur le Web. C'est ce que nous appelons l'ouverture des données. Cette ouverture permet à quiconque de pouvoir accéder directement aux données. L'intérêt de l'ouverture des données est de pouvoir les utiliser dans différentes applications ou encore de les croiser avec d'autres données pour les analyser.

Les données ouvertes ne sont pas forcément interopérables. Le format des données n'étant pas nécessairement identique pas plus que le niveau de formalisme, il peut être difficile de réutiliser des données à ce point hétérogènes. Le Web de données liées (ou Linked Open Data) propose de représenter les données sous un format unique afin de favoriser l'interopérabilité des données. Le Web de données liées cherche aussi à favoriser les liens et la réutilisation des données. Pour cela, il propose de représenter les données en utilisant un vocabulaire défini à partir d'une ontologie. Celle-ci permet de définir formellement les concepts nécessaires à l'interprétation des données de façon à lever toute ambiguité. En informatique, le résultat de cette reformulation des données par le biais de concepts s'appelle une représentation des connaissances.

Un ensemble de formalismes de représentation de la connaissance a été défini par le W3C [1]. Ainsi, les données peuvent non seulement être comprises et interprétées par des humains mais également par des machines. Ces formalismes permettent alors la déduction de nouvelles connaissances. De plus, il devient possible de créer des liens d'équivalence entre les connaissances provenant de sources différentes. De cette manière, un ensemble de connaissances est disponible sur Internet, ces connaissances étant toutes liées entre-elles. C'est ce que l'on appelle le Web sémantique.

Un des verrous majeurs au développement du Web de données est la possibilité de réutiliser efficacement les nombreuses sources de données structurées pour participer au Web de données liées.

Un domaine d'étude particulièrement intéressant pour illustrer ce verrou est l'agriculture. Ce domaine comporte de nombreuses sources structurées mais peu sont présentes sur le Web de données liées. La mise à disposition de ces sources sur le Web de données liées serait un avantage pour l'évolution de l'agriculture vers une agriculture durable.

En effet, dans ce domaine, un des défis majeurs pour les prochaines années réside dans l'amélioration des pratiques agricoles, autant d'un point de vue économique qu'écologique. Ce défi transparaît dans les priorités définies dans le programme de financement européen

1. http://www.w3.org/

"Horizon 2020". C'est dans ce cadre que le Ministère français de l'agriculture a lancé en 2009 le plan Ecophyto [2] afin de diviser par deux l'utilisation de pesticides dans les cultures en 2018. Ce plan, proposé lors du Grenelle de l'environnement en 2007, est le résultat de nombreuses observations concernant les effets néfastes sur l'environnement de l'utilisation intense de produits phytosanitaires. Ce plan est divisé en trois enjeux majeurs.

Le premier enjeu concerne la santé publique. Un des effets collatéraux de l'utilisation de produits phytosanitaires est l'apparition de résidus de produits toxiques dans les récoltes des cultures traitées. Ces résidus apparaissent aussi dans les ruisseaux qui coulent près de cultures. L'apparition de ces résidus dans les eaux et les récoltes nous amène à considérer les impacts sur la santé publique.

Le deuxième enjeu de ce plan provenant du Grenelle est écologique. Le Grenelle de l'environnement avait pour but de résoudre différents problèmes concernant l'environnement et l'écologie. L'utilisation de produits phytosanitaires dans les cultures, bien que permettant la lutte contre les bio-agresseurs [3], engendre aussi un certain nombre de dommages collatéraux. La pulvérisation de ces produits dans les champs va aussi tuer d'autres organismes vivants qui sont eux nécessaires au bon développement des cultures. Leur disparition affecte également l'écosystème.

Le plan Ecophyto ne propose pas de stopper totalement l'utilisation de produits phytosanitaires, car ils sont nécessaires pour obtenir une agriculture permettant de subvenir aux besoins de la population. L'arrêt total de l'utilisation de produits phytosanitaires apporterait non seulement une production agricole moins rentable mais aussi d'autres problèmes sanitaires sur la qualité des produits à consommer. C'est ici le dernier enjeu majeur du plan Ecophyto : produire autrement. Ce plan propose de favoriser la transition vers une agriculture durable.

Pour répondre à ces différents enjeux, le plan Ecophyto propose, notamment, plusieurs systèmes de surveillance des pratiques agricoles et d'épidémiosurveillance. L'un d'entre-eux se fonde sur des bulletins d'alertes qui informent les acteurs de la sphère agricole d'attaques de bio-agresseurs sur les cultures : les "Bulletin de Santé du Végétal" (BSV) [4]. Ces bulletins proposent de présenter régulièrement un état des lieux des différentes attaques de bio-agresseurs dans les cultures d'une région donnée. Les agriculteurs et les conseillers techniques peuvent alors utiliser ces BSV pour déterminer le niveau de risque d'une attaque dans leur région et agir en conséquence en traitant par exemple uniquement les cultures à risque élevé. Afin de suivre l'évolution des attaques de bio-agresseurs sur plusieurs décennies et sur tout l'espace français, ces bulletins ont besoin d'être rassemblés, analysés et annotés. De cette manière, il serait alors possible de faire des analyses plus fines sur l'évolution des bio-agresseurs dans le temps et dans l'espace. C'est ce que propose de faire le projet VESPA. La première étape nécessaire au processus d'annotation consiste à construire une source de référence, constituée d'un réseau de bases de connaissances, concernant tous les organismes susceptibles d'apparaître dans les champs (plante cultivée,

2. http://agriculture.gouv.fr/ecophyto
3. Organismes vivants attaquant les cultures (maladies, champignons, insectes, ...)
4. http://agriculture.gouv.fr/ecophyto-BSV

auxiliaire de culture, agresseur).

Un avantage pour la construction de cette source est qu'en agriculture, de nombreuses données concernant les cultures sont disponibles dans différents formats électroniques : thésaurus, bases de données... En plus de la réutilisation de ces données pour la création de la source de référence, un défi pour les années à venir est de rendre ces données accessibles à tous les acteurs (agriculteurs, conseillers agricoles, chercheurs en agronomie, etc.). La référence sert dans un premier temps à faciliter la recherche (l'accès) aux BSV . Elle propose un langage commun d'interrogation à toutes les régions. Ensuite, elle pourra être réutilisée pour faciliter l'intéropérabilité des données.

Le travail présenté dans ce manuscrit se place dans ce contexte. Notre objectif est de réutiliser les sources existantes sur le Web pour en générer une base de connaissances. Pour cela, nous proposons une méthodologie permettant de considérer simultanément plusieurs sources non-ontologiques afin de générer une base de connaissances finale. Nous avons centré nos travaux autour d'une hypothèse : si un élément apparaît dans plusieurs sources, alors la possibilité qu'il appartienne au domaine d'étude est accrue. Les travaux actuels concernant la transformation de sources non-ontologiques considèrent les sources de manière séquentielle. Ce traitement séquentiel n'apporte pas la possibilité de déterminer un score de confiance aux éléments d'après le consensus pouvant exister entre les sources.

Pour effectuer ce traitement simultané, nous avons défini trois étapes principales : (1) analyse des sources, (2) transformation automatique des sources et (3) fusion des sources. La première étape permet de dresser la liste des sources à considérer pour un projet particulier. Les sources présentes sur le Web sont variées et abordent différents sujets. Il faut donc pouvoir déterminer quelles sont les sources qui sont pertinentes pour notre cas d'étude. Nous effectuons ensuite une transformation automatique des sources. Cette étape étant automatisée, il est possible d'obtenir des erreurs dans le résultat final après la transformation. L'avantage de notre proposition réside dans la détection de ces erreurs. Nous partons de l'hypothèse selon laquelle les erreurs apparaissant dans les sources transformées (qu'elles proviennent de la source ou de la transformation automatique) n'apparaîtront pas dans les autres sources. Ainsi, nous espérons que la recherche du consensus entre sources éliminera une bonne partie des erreurs. C'est pour cette raison que nous définissons une fonction de confiance permettant de représenter le niveau de consensus entre les sources concernant cet élément.

Ce manuscrit est consacré à la définition de cette méthodologie. Pour cela, nous présenterons dans la partie I le contexte dans lequel nous appliquons notre méthodologie. Nous étudions les différents travaux existants sur lesquels nous avons fondé notre méthodologie dans la partie II. Nous présentons ensuite les différentes étapes de cette méthodologie dans la partie III. Nous aborderons dans la partie IV les différents aspects principaux de l'implémentation de cette méthodologie dans l'outil Muskca. Enfin, nous avons expérimenté cette méthodologie sur deux jeux de données, concernant deux domaines différents. Nous présenterons ces expérimentations dans la partie V.

# Première partie .
# Contexte

Le Web, tel que nous le connaissons aujourd'hui, permet de mettre en relation des informations sous forme de documents compréhensibles principalement par des humains. Il prend la forme de pages Web accessibles sur le réseau internet. Le Web sémantique apporte une extension à ce Web en proposant une représentation formelle de la connaissance pour qu'elle puisse également être interprétée par les machines. Ceci permet de faire abstraction de l'interprétation des informations et de manipuler directement de la connaissance. Le W3C a proposé un certain nombre de recommandations de technologies pour mettre en œuvre le Web sémantique. Une application concrète du Web sémantique est le Web de données liées, qui exploite les technologies du Web sémantique pour améliorer l'interopérabilité des données. Le Web de données liées propose une représentation des données sous une forme favorisant les liens entre les silos de données.

Le milieu de l'agriculture est en pleine transformation depuis le dernier Grenelle de l'environnement. Ce Grenelle a fortement encouragé les agriculteurs à passer à une agriculture dite durable et plus particulièrement à une agriculture de précision. Ce type d'agriculture préconise l'utilisation de produits phytosanitaires uniquement quand cela est nécessaire et non plus par précaution. La détection d'alertes agricoles pouvant motiver l'utilisation de produits phytosanitaires n'est pas toujours évidente à cause de la disparité des données disponibles dans ce domaine. Nous retrouvons aussi ce phénomène de cloisonnement des données concernant la documentation des techniques alternatives à l'usage des produits phytosanitaires. Pour pouvoir favoriser la transition vers une agriculture durable, il est nécessaire de croiser des données présentes dans ces différents silos. Le Web de données liées propose une solution pour décloisonner ces données et donc favoriser la transition vers l'agriculture durable.

Nous allons étudier ces différents aspects dans ce chapitre en présentant d'abord le Web sémantique et son application : le Web de données liées. Nous présenterons ensuite nos motivations pour l'enrichissement du Web de données liées agricoles et son état actuel.

## 1. Le Web sémantique

### 1.1. Historique

Tim Berners-Lee, un physicien du CERN [5] était, dans les années 80, en charge du développement d'un outil pour faciliter l'échange de documents entre les différents chercheurs. À cette époque, la rédaction numérique en est à ses balbutiements : les formats des documents sont hétérogènes et les outils incompatibles. Le CERN étant une structure internationale, cette problématique de partage de documents en est d'autant plus difficile. À cette même époque apparaissent les prémisses du Web, avec l'utilisation

5. Organisation européenne pour la Recherche nucléaire - http://home.web.cern.ch/fr

de document de type "hypertexte" [6]. Tim Berners-Lee proposa donc un outil collaboratif, "ENQUIRE" [Berners-Lee, 1989], pour permettre le partage de documents entre les chercheurs du CERN en se fondant sur les documents hypertexte.

Suite au développement de cet outil, Tim Berners-Lee proposa une amélioration des documents hypertextes s'appuyant sur la technologie connue sous le nom d'internet. Si une URL fait référence à un document, représenté sous un format standard, tout le monde peut lire ce document dès que l'on connaît son URL. Tim Berners-Lee développa le premier navigateur Web, et le premier serveur Web, à la fin de l'année 1990. Le premier langage qui avait pour vocation à devenir le standard de représentation des documents fut le SGML [ISO 8879, 1986] mais ce langage fut jugé trop complexe par Tim Berners-Lee qui préféra proposer une application plus simple du SGML en 1991 : HTML.

Il faudra attendre la fin de l'année 1995 pour voir apparaître la première spécification RFC sur le HTML [Berners-Lee and Connolly, 1995], aidé par l'appui du W3C [7] à cet époque nouvellement créée.

Lors de sa première proposition pour le système ENQUIRE [Berners-Lee, 1989], Tim Berners-Lee a non seulement posé les bases du Web, mais il a aussi, dès cette époque, commencé les réflexions autour de la représentation des connaissances et plus spécifiquement du Web sémantique. Cette proposition contient un chapitre intitulé "Linked information systems" posant les premières idées qui donneront naissance au Web sémantique, comme la notion de triplet, d'URI unique, etc. Dans son ouvrage "Weaving the Web" [Berners-Lee et al., 2000], dans lequel Tim Berners-Lee présente sa vision du future du Web, une citation permet de résumer son ambition concernant le Web sémantique :

> I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

Son idée est de représenter la signification des données et de lier ces données entre elles afin de faciliter les analyses et les interactions, que ce soit entre machines ou entre une machine et un humain. Cette représentation du sens des données revient à trouver un formalisme suffisamment expressif pour représenter ce que nous appellerons des connaissances.

## 1.2. Représentation des connaissances

Nous allons ici présenter ce que nous appelons la représentation des connaissances. Pour cela, nous définirons ce que nous appelons une connaissance en faisant la différence avec les notions de donnée et d'information. Nous présenterons ensuite le modèle de graphe sur lequel repose la représentation des connaissances sur le Web sémantique. Puis nous présenterons la définition d'une ontologie qui modélise le vocabulaire utilisé pour

6. Contenant des liens numériques (URL) vers d'autres documents
7. World Wide Web Consortium - http://www.w3.org/

définir des connaissances et leurs contraintes. Enfin, nous définirons ce qu'est une base de connaissances qui repose sur l'union d'une ontologie et de connaissances assertionnelles.

### 1.2.1. Données/Informations/Connaissances

Nous parlons ici de formaliser le sens associé aux données dans le but de représenter de la connaissance. Afin de pouvoir effectuer cette action, il est nécessaire de désambiguïser les termes données et connaissances. De plus, il existe un statut intermédiaire qui est l'information. La distinction entre ces trois notions est souvent ambiguë et, dans le meilleur des cas, subjective. Il n'y a aucune définition consensuelle de ces trois termes [Zins, 2007], bien qu'ils soient utilisés régulièrement. Nous allons donc donner nos définitions de ces termes afin de désambiguïser leur utilisation dans ce manuscrit. Nous nous sommes particulièrement inspiré des travaux [Pradel, 2013].

**Donnée :** Une donnée est une mesure dite brute. C'est une valeur, typée ou non, obtenue grâce à un capteur, une base de données, une interaction avec un utilisateur ou autre. Une donnée a vocation à être stockée, transmise et analysée. Elle ne véhicule aucune signification. Pour pouvoir traiter cette donnée, une notion est importante : son contexte. Nous pouvons prendre l'exemple d'une donnée de type entier : "42". Bien que cet entier puisse, pour vous, avoir un sens, elle n'en reste pas moins, pour un système informatique, qu'une valeur dénuée de sens.

**Information :** Une information est une donnée associée à son contexte. Grâce à ce contexte, la donnée peut être traitée et analysée. Cette donnée gagne en expressivité, puisqu'elle véhicule maintenant un sens. Cette information est interprétable soit par une machine, soit par un humain. Par exemple la valeur entière précédente, "42", peut être associée au contexte comme étant l'âge de Pierre. De cette manière, la donnée prend sens, et donc devient une information.

**Connaissance :** Une connaissance est la formalisation d'une information dans un langage de représentation des connaissances. Cette formalisation, une fois associée à d'autres connaissances, permet la déduction de nouvelles connaissances. Cette nouvelle connaissance vient enrichir l'ensemble des connaissances initiales et permet donc de nouvelles déductions. Pour reprendre l'exemple précédent, la formalisation de l'information "Pierre est âgé de 42 ans" sous une forme de connaissance, si elle est associée à la connaissance initiale "Un quadragénaire est une personne dont l'âge est compris entre 40 et 49 ans" nous permet d'en déduire que "Pierre est un quadragénaire".

### 1.2.2. Le modèle de graphe RDF

Pour représenter la connaissance telle que nous venons de la définir et que cette représentation puisse être interprétable par une machine, il est nécessaire d'utiliser un formalisme. Pour cela, le W3C a proposé de représenter les connaissances sous forme de graphes. Les formalismes proposés par le W3C reposent sur deux principes fondamentaux.

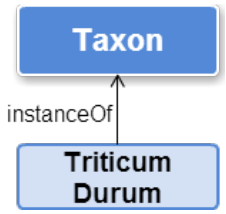


FIGURE 1 – Exemple de triplet RDF

Le premier est l'utilisation d'URI [8] déréférençables identifiant chaque entité. Comme nous l'avons vu, le Web utilise une chaîne de caractères appelée URL pour identifier un document. Une URI est une chaîne de caractères identifiant une entité qui peut être une connaissance. Nous pouvons prendre l'exemple de l'URI suivante :

`http://aims.fao.org/aos/agrovoc/c_7955`

Cette URI permet l'identification de la ressource représentant "Triticum Durum" [9] dans Agrovoc [10]. Celle-ci est déréférencable puisque si nous interrogeons cette URI par l'intermédiaire d'un navigateur Web, alors nous obtenons un rendu compréhensible par un humain. Néanmoins, si cette URI est interrogée par un système informatique, alors le résultat obtenu n'est pas présenté sous une forme compréhensible par un humain, mais sous une forme interprétable par la machine. Pour que ce résultat soit interprétable par une machine il est nécessaire qu'il soit représenté sous un format spécifique. Une ressource est une entité identifiée par une URI. Une IRI [11] est une évolution de l'URI en considérant la table de caractères Unicode et non plus ASCII comme c'est le cas pour les URI : une IRI permet l'utilisation de caractères internationaux.

Le deuxième principe proposé par le W3C est de représenter les liens entre entités à partir de triplets RDF. Le format RDF [12] propose de représenter toute la connaissance à partir de trois composants : le sujet, le prédicat et l'objet. Il n'est possible de représenter en RDF que ce qui peut être traduit sous forme de triplets.

Triplet RDF

| $< sujet >< predicat >< objet > .$ |
|---|

Nous pouvons représenter l'exemple précédent : "Triticum Durum est un Taxon" sous forme d'un triplet RDF, comme le montre la figure 1. Cet exemple est présenté sous une forme graphique. Le sujet du triplet est "Triticum Durum", le prédicat est représenté par l'étiquette de l'arc "instanceOf" et l'objet est "Taxon".

8. Uniform Resource Identifier
9. Un rang taxonomique dans la classification taxonomique des plantes. Ce rang représente le blé dur.
10. http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural-thesaurus
11. International Resource Identifier
12. Resource Description Framework - http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

Le W3C a proposé plusieurs formats de représentation des triplets RDF, comme RDF/XML [13] ou JSON-LD [14]. Nous utiliserons dans ce manuscrit le format Turtle [15] lorsque nous souhaiterons représenter des triplets dans un format compréhensible par la machine. Ce format permet la représentation de chaque élément d'un triplet RDF entre chevrons et chaque triplet se termine par un point. Si nous souhaitons représenter le triplet "Triticum Durum est un Taxon" en Turtle nous obtenons :

```
<http://base-uri.fr/Triticum_Durum>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://base-uri.fr/Taxon>.
```

Nous pouvons observer que les URI du sujet et de l'objet utilisent le même espace de nom "http ://base-uri.fr/". Pour simplifier l'écriture et utiliser moins de caractères pour représenter le triplet (dans le but de minimiser la quantité de données à transférer), il est possible de spécifier des préfixes. Ces préfixes permettent de définir une chaîne de caractères plus courte pour remplacer un nom de domaine récurrent. Dans l'exemple précédent, nous pouvons utiliser un préfixe pour "http ://base-uri.fr/" et un pour "http ://www.w3.org/1999/02/22-rdf-syntax-ns#" [16] :

```
@base <http://base-uri.fr/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
:Triticum_Durum rdf:type :Taxon.
```

Avec le mot clef "@base", nous définissons le préfixe par défaut. Ce préfixe est utilisé lorsqu'il n'y a pas de préfixe défini. C'est par exemple le cas dans " :Triticum_Durum" qui correspond à l'URI "http ://base-uri.fr/Triticum_Durum". Nous définissons aussi le préfixe "rdf :" utilisé pour le prédicat.

### 1.2.3. Ontologie

Lorsque nous parlons de représentation de connaissances, nous faisons la distinction entre la partie ontologique et la partie assertionnelle de cette représentation. La partie ontologique permet de définir le vocabulaire utilisé dans la partie assertionnelle. Nous utiliserons dans ce manuscrit la définition d'une ontologie qui fait référence dans la communauté de l'ingénierie des connaissances, introduite par [Gruber, 1993] et étendue par [Borst, 1997] :

> une ontologie est une spécification formelle explicite d'une conceptualisation partagée d'un domaine donné.

Nous pouvons expliquer cette définition en utilisant les travaux de [Studer et al., 1998] :
— formelle se réfère au fait que la spécification doit être lisible par une machine,

13. http://www.w3.org/TR/rdf-syntax-grammar/
14. http://www.w3.org/TR/json-ld/
15. http://www.w3.org/TR/2014/REC-turtle-20140225/
16. Nom de domaine pour le vocabulaire défini pour RDF

— explicite signifie que les types des concepts et les contraintes sur leur utilisation sont explicitement définis,
— conceptualisation se réfère à un modèle abstrait d'un certain phénomène du monde reposant sur l'identification des concepts pertinents de ce phénomène et
— partagée se rapporte à la notion selon laquelle une ontologie capture la connaissance consensuelle, qui n'est pas propre à un individu mais validée par un groupe.

Il est nécessaire de faire la distinction entre deux types de connaissances : les connaissances assertionnelles et les connaissances ontologiques. Une connaissance assertionnelle est la formalisation d'une information factuelle, décrivant un fait précis. Elle est à différencier de la connaissance ontologique (ou terminologique) qui permet de définir le vocabulaire utilisé dans la partie assertionnelle. La connaissance "Un quadragénaire est une sorte de personne" est une connaissance ontologique puisqu'elle permet de définir le concept de "quadragénaire" ; à la différence de "Pierre est âgé de 42 ans" qui représente une information factuelle et est donc une connaissance assertionnelle.

Afin de définir la partie ontologique d'une base de connaissances, deux vocabulaires ont été définis par le W3C. Le premier est RDFS [17]. Ce vocabulaire permet principalement la définition de relations hiérarchiques entre les classes et entre les propriétés. Le deuxième vocabulaire est OWL [18]. Ce deuxième vocabulaire permet une définition plus formelle des connaissances. Il permet entre autres de définir des classes disjointes ou encore de permettre l'identification du type d'un individu en fonction de ses caractéristiques et de définitions de classes. Nous pouvons reprendre notre exemple en précisant que "Un quadragénaire est une personne dont l'âge est compris entre 40 et 49 ans". Cette restriction sur l'âge d'un individu de type quadragénaire peut être exprimée en OWL et peut permettre de déduire le type d'un individu. Ces déductions logiques sont appelées des inférences. Les seules inférences possibles en utilisant le vocabulaire RDFS sont les inférences hiérarchiques ou sur les domaines et co-domaines des propriétés (que nous définirons plus loin dans cette section). Par exemple, si "Quadragénaire" est une sous classe de "Personne" et que "Roger" est un "Quadragénaire" alors "Roger" est une "Personne". Nous considérons alors que les ontologies utilisant le vocabulaire RDFS et non le vocabulaire OWL sont des ontologies dites "légères" alors que les ontologies utilisant le vocabulaire OWL sont des ontologies dites "lourdes". Il peut être intéressant de manipuler uniquement une ontologie légère par volonté de simplification de la modélisation et de temps nécessaire pour effectuer toutes les inférences possibles.

### 1.2.4. Base de connaissances

Une base de connaissances est le regroupement de la partie ontologique et de la partie assertionnelle. La partie ontologique se compose d'un ensemble de classes et de propriétés qui peuvent être utilisées dans la partie assertionnelle. Cette partie assertionnelle est constituée d'individus et de relations définis avec les classes et les propriétés de la partie ontologique. Chaque classe, individu ou propriété peut être associé à un ou plusieurs

17. RDF Schema - http://www.w3.org/TR/rdf-schema/
18. Web Ontology Language - http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

labels. Nous faisons la distinction entre étiquette et label dans ce manuscrit. Les labels sont des chaînes de caractères représentant la ressource alors qu'une étiquette est une URI identifiant une ressource.

Nous définissons une base de connaissances :
$KB = (V_{kb}, E_{kb}, \Sigma V_{kb}, \Sigma E_{kb}, etiqV_{kb}, etiqE_{kb}, sourceE_{kb}, cibleE_{kb})$
$KB$ est un multigraphe orienté étiqueté. Les éléments de la base de connaissances $KB$ sont définis de la manière suivante :

base de connaissances

$V_{kb} = C \cup PropO \cup PropDT \cup I \cup L$ est un ensemble fini de sommets qui sont répartis dans des sous-ensembles disjoints tels que :

- $C$ est l'ensemble des classes.
- $PropO$ est l'ensemble des propriétés d'objets.
- $PropDT$ est l'ensemble des propriétés de types de données.
- $I$ est l'ensemble des individus.
- $L$ est l'ensemble des sommets représentant les littéraux, dont les labels

$C \cup PropO \cup PropDT \cup L$ sont les sommets de la partie ontologique et $I \cup L$ sont les sommets de la partie assertionnelle. Les sommets labels sont dans les deux parties puisqu'ils peuvent représenter une ressource de la partie ontologique ou une ressource de la partie assertionnelle. $\Sigma V_{kb}$. est l'ensemble des étiquettes des sommets de la base de connaissances. Ces identifiants sont des chaînes de caractères alphanumériques pour les littéraux ou des URI pour les autres sommets.
$etiqV_{kb} : V_{kb} \rightarrow \Sigma V_{kb}$ est une application qui, à tout sommet $v$ de $V_{kb}$, associe une seule étiquette $l \in \Sigma V_{kb}$.
$\Sigma E_{kb} = \{subClassOf, subPropertyOf, instanceOf, rdfs : label\} \cup \Sigma_{propO} \cup \Sigma_{propDT}$ est un ensemble fini d'étiquettes d'arcs.

- $subClassOf$ est la relation de spécialisation entre classes,
- $subPropertyOf$ est la relation de spécialisation entre propriétés (propriété d'objet ou de type de données),
- $rdfs : label$ est la propriété d'annotation associant un label à une classe, à une propriété ou à un individu.
- $instanceOf$ exprime l'appartenance d'un individu à une classe.
- *domain* (ou domaine en français) exprime le type maximal des sujets des triplets dont le prédicat a la même étiquette que la propriété. catherine autre proposition d'écriture : cette relation exprime une contrainte de domaine sur l'usage d'une propriété : tout triplet ayant comme prédicat la propriété doit avoir comme sujet un individu du type de la classe.
- *range* (ou co-domaine en français) exprime le type maximal des objets des triplets dont le prédicat a la même étiquette que la propriété.
- $rdfs : label$ est la relation vers une chaîne de caractères lisible par un humain.

— $\Sigma_{propO}$ est l'ensemble des étiquettes des sommets $v \in PropO$ correspondant aux propriétés d'objets qui vont être utilisées comme étiquettes d'arcs dans $KB$. $\forall l \in \Sigma_{propO}$ $\exists v \in PropO$ tel que $etiqV_{kb}(v) = l$. En effet, pour qu'une propriété d'objet soit utilisée dans une base de connaissances, elle doit d'abord être définie par un sommet dans l'ontologie associée.

— $\Sigma_{propDT}$ est l'ensemble des étiquettes des sommets $v \in PropDT$ correspondant aux propriétés de types de données qui vont être utilisées comme étiquettes d'arcs dans $KB$. $\forall l \in \Sigma_{propDT}$ $\exists v \in PropDT$ tel que $etiqV_{kb}(v) = l$.

$E_{kb} \subset \Sigma E_{kb} \times V_{kb} \times V_{kb}$ est l'ensemble fini d'arcs (ou relations) étiquetés de la forme $p(v_1, v_2)$ ayant une étiquette $p \in \Sigma E_{kb}$, un sommet sujet $v_1 \in V_{kb}$ et un sommet objet $v_2 \in V_{kb}$.

Les arcs de $KB$ remplissent l'une des conditions suivantes :

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = rdfs : label \Rightarrow v_1 \in C \cup PropO \cup PropDT \cup I$ et $v_2 \in L$

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = subClassOf \Rightarrow v_1 \in C$ et $v_2 \in C$

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = subPropertyOf \Rightarrow v_1 \in PropO \cup PropDT$ et $v_2 \in PropO \cup PropDT$,

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p \in \Sigma_{propO} \Rightarrow v_1 \in I$ et $v_2 \in I$,

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p \in \Sigma_{propDT} \Rightarrow v_1 \in I$ et $v_2 \in L$,

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = instanceOf \Rightarrow v_1 \in I$ et $v_2 \in C$.

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = domain \Rightarrow v_1 in PropO \cup PropDT$ et $v_2 \in C$.

— $\forall p(v_1, v_2)$ un arc étiqueté tel que $p = range \Rightarrow v_1 in PropO \cup PropDT$ et $v_2 \in C$.

Une base de connaissances est donc un multigraphe tel qu'il n'existe qu'un seul arc étiqueté par une même étiquette reliant un sommet sujet à un sommet objet. Nous définissons trois applications sur les arcs de la base de connaissances :

$etiqE_{kb} : E_{kb} \rightarrow \Sigma E_{kb}$ est une application qui associe à chaque arc son étiquette.

$sourceE_{kb} : E_{kb} \rightarrow V_{kb}$ est une application qui associe à chaque arc son sommet sujet.

$cibleE_{kb} : E_{kb} \rightarrow V_{kb}$ est une application qui associe à chaque arc son sommet objet.

Nous ne considérons ici que les étiquettes :
$\{subClassOf, subPropertyOf, domain, range, instanceOf, rdfs : label\}$
en plus de celles définies dans $\Sigma_{propO} \cup \Sigma_{propDT}$ car nous ne considérerons dans ces travaux que ces propriétés des vocabulaires RDF et RDFS.

La décision de formaliser une base de connaissances sous forme d'un graphe étiqueté est fondée sur la volonté de clarifier le lien entre une propriété définie dans l'ontologie et une relation lui faisant référence dans la partie assertionelle. Il est en revanche primordial de ne pas confondre les étiquettes, qui sont les URI, et les labels qui sont des littéraux de la base de connaissances. Ces labels sont des commentaires des ressources auxquelles ils sont rattachés. Dans l'ensemble de ce manuscrit, nous simplifierons la notation en spécifiant "la relation X" au lieu de précisier "la relation dont l'étiquette fait référence à la propriété X".


FIGURE 2 – Définition du vocabulaire graphique

Les exemples illustrés par des figures dans la suite de ce document utiliseront le vocabulaire graphique défini par la légende présentée dans la figure 2.

La partie ontologique de la figure 3 montre un exemple d'une ontologie représentant une sous-partie d'une taxonomie des plantes. Nous représentons la classe "Taxon", qui est la généralisation des classes "Species" et "Genus" (matérialisée par l'arc "subClassOf" reliant ces classes). La propriété "hasHigherRank" est définie avec les arcs "domain" et "range" (domaine et co-domaine en francais) permettant de connaître le type du sujet et de l'objet lorsque cette propriété sera utilisée ; le sujet et l'objet d'un triplet dont le prédicat est étiqueté par la propriété "hasHigherRank" sont forcément des "Taxon". La propriété "hasDirectHigherRank" est une spécialisation de la propriété "hasHigherRank" (matérialisée par l'arc "subPropertyOf"). Deux labels ("Genus" et "Genre") sont associées à la classe "Genus".

La partie assertionnelle de la figure 3 présente deux individus : "Triticum Durum" et "Triticum", le premier étant une instanciation de la classe "Species" et le deuxième de la classe "Genus". Ces deux individus sont reliés par l'arc "hasDirectHigherRank" qui est la propriété définie dans l'ontologie. Deux labels ("Wheat" et "Triticum") sont définis pour l'individu "Triticum".

## 1.3. Le Web de données liées

La figure 4 présente le nuage du Web de données liées généré en août 2014. Le Web de données liées est une initiative qui vise à réduire l'isolement des silos de données. En effet, la gestion des données grâce à des bases de données traditionnelles conduit à un isolement des données, puisqu'il est impossible de partager des données entre bases de

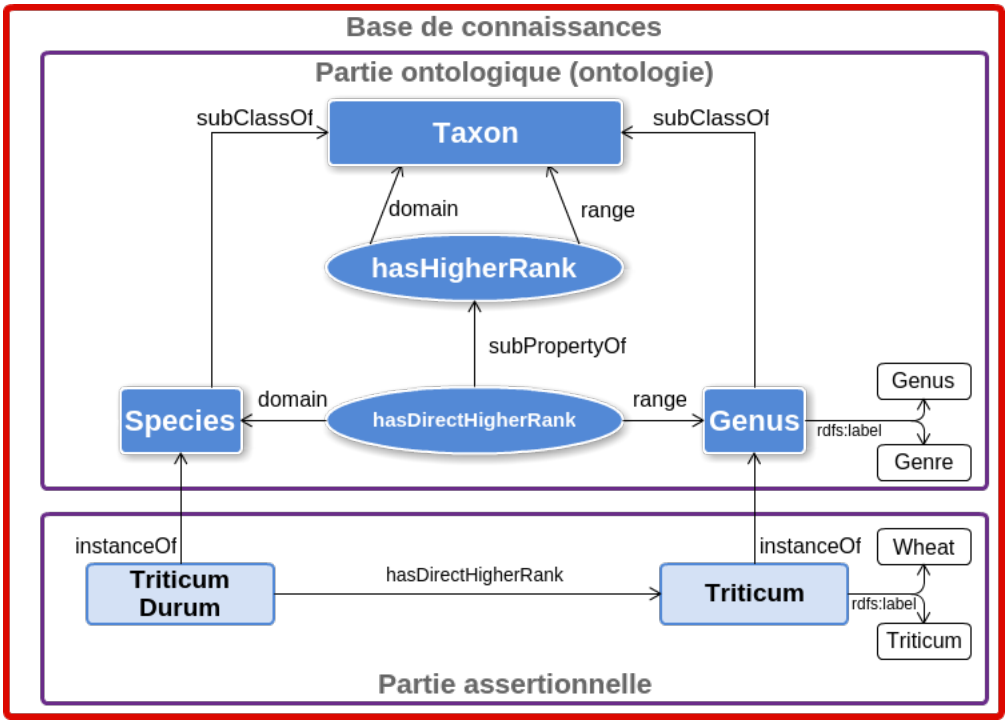


FIGURE 3 – Exemple d'une base de connaissances

données différentes sans les dupliquer. Le Web de données liées apporte une solution permettant de partager des données communes à plusieurs jeux de données. Il est aussi possible de dupliquer des données et d'indiquer qu'il s'agit de données similaires. Grâce à ces relations de similitude, il est possible pour un système d'analyser un jeu de données puis de continuer l'analyse sur un autre jeu de données en connaissant les liens de passage de l'un à l'autre.

Pour cela, le Web de données liées (ou le LOD [19]) est fondé sur trois éléments. Les données doivent tout d'abord être représentées au format RDF afin de permettre l'uniformité des formats entre les jeux de données et donc de permettre l'interopérabilité. Le deuxième élément permettant l'utilisation du LOD est l'utilisation d'URI déréférencables. Une URI de ce type permet d'identifier un élément d'un jeu de données tel que nous l'avons défini précédemment. Le troisième et dernier élément nécessaire à la mise en place du LOD est la réutilisation de ressources dans différents jeux de données. Certaines ressources définies dans des jeux de données sont réutilisables dans un autre jeu de données. Cela permet d'éviter de dupliquer la ressource dans chaque jeu de données. Nous pouvons par exemple citer le jeu de données "foaf" [20] qui définit la classe "Person". Cette classe permettant de représenter une personne peut être réutilisée dans un autre jeu de données, par exemple pour modéliser des étudiants dans une université. Une alternative à la réutilisation des

19. Linked Open Data
20. Friend Of A Friend - http://xmlns.com/foaf/spec/

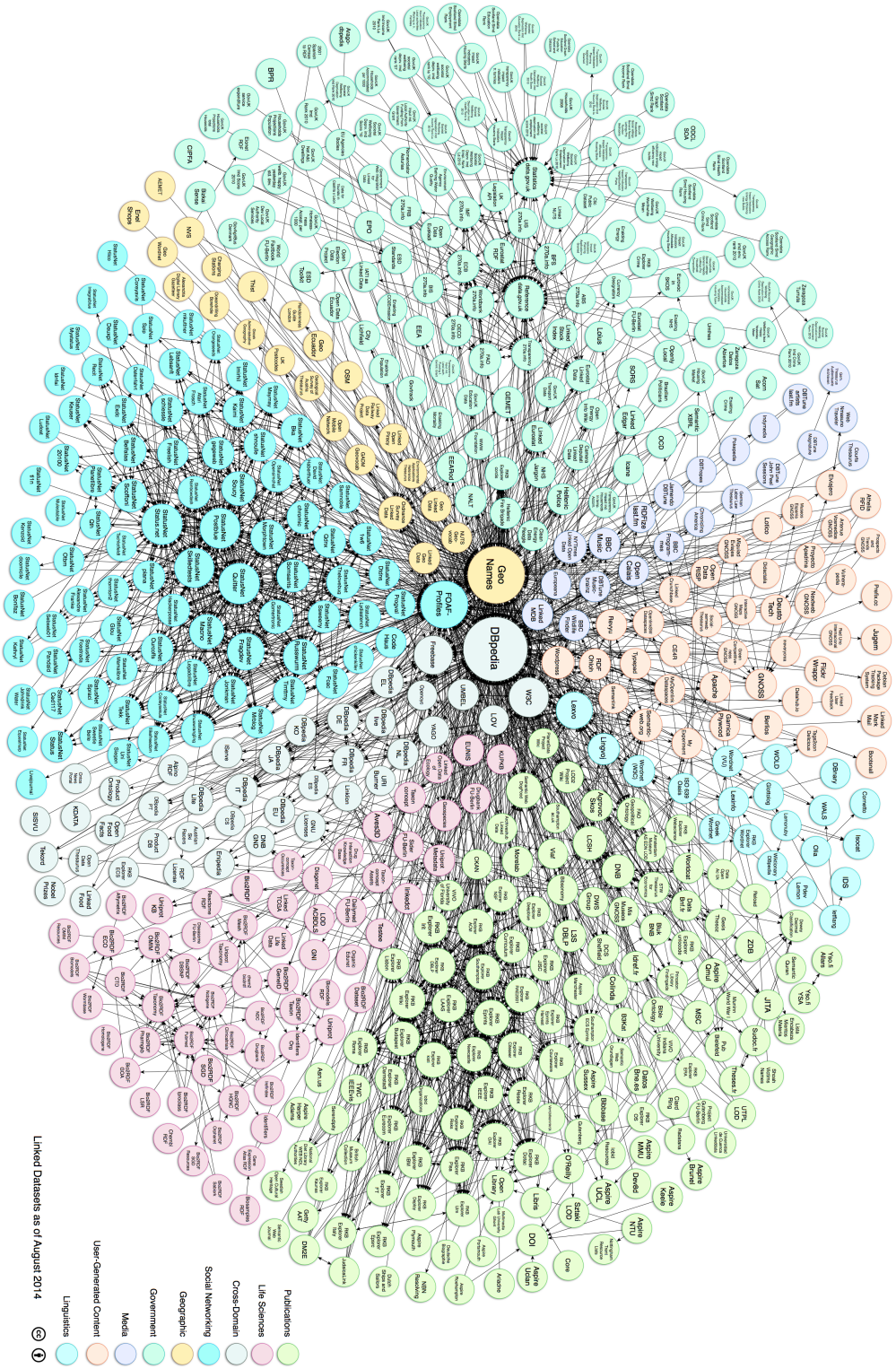


FIGURE 4 – Le graphe du Web de données liées en 2014

ressources est la mise en correspondances de deux ressources. La relation "sameAs" du vocabulaire OWL permet de définir une équivalence sémantique entre deux ressources.

La figure 4 présente les jeux de données utilisant ce principe (ce sont les nœuds du graphe sur la figure) et les relations existant entre les différents jeux de données (les liens sur la figure). N'apparaissent sur cette figure que les jeux de données pour lesquels les administrateurs ont renseigné l'existence auprès de l'organisme [21] auteur du nuage présenté dans la figure.

Nous pouvons remarquer sur cette figure un nœud central : DBPedia. Ce jeu de données, que nous présenterons dans la section 2.4.4 du chapitre I, joue un rôle important dans le LOD grâce à la grande quantité de domaines qu'il couvre. Il est en effet courant de trouver une ressource que l'on souhaite modéliser dans DBPedia. L'avantage est que ce jeu de données est considéré comme un intermédiaire pour établir des liens entre d'autres jeux de données. En effet, même si deux jeux de données ne partagent pas directement une ressource, mais qu'ils partagent la même ressource avec DBPedia, alors un lien indirect existe entre ces deux jeux de données.

Le LOD est donc une application des technologies du Web sémantique favorisant l'interopérabilité de jeux de données différents, permettant le décloisonnement de ces jeux de données. La considération du LOD facilite alors les analyses croisées des données présentes dans plusieurs jeux de données. Nous pouvons observer le succès du LOD grâce aux nombreux jeux de données présents sur la figure 4.

## 1.4. Le Web sémantique aujourd'hui

Comme nous avons pu l'observer, le Web sémantique repose sur un certain nombre de technologies. Ces différentes technologies sont dépendantes les unes des autres. C'est par exemple le cas de RDF qui repose sur l'utilisation d'URI. La figure 5 présente ces technologies et leurs dépendances.

Cette figure présente le "cake du Web sémantique". Les technologies sont représentées sous la forme de couches du cake et la superposition des couches représente les dépendances. Pour reprendre notre précédent exemple, la couche RDF est bien au-dessus de la couche URI. La couche XML permet de préciser qu'il est possible de représenter du RDF en XML. Nous avons aussi abordé les vocabulaires RDFS et OWL présents dans cette figure. La couche SPARQL représente le langage d'interrogation de données au format RDF. Ce langage, proche de SQL, permet la récupération, l'ajout, la suppression et la modification de triplets RDF. Le langage RIF [22] permet quant à lui de définir des règles d'inférences spécifiques sous un format interopérable et partageable. La couche "Unifying Logic" permet de définir des méthodes de déduction de nouvelles connaissances en exploitant les vocabulaires utilisés et les règles d'inférences définies. La couche "Proof" permet de spécifier la preuve de la pertinence des données, à partir notamment de leur provenance. Toutes ces couches sont en parallèle avec la couche "Crypto". En effet, afin de garantir une communication sécurisée et donc un partage de données non altéré, il est

21. http://datahub.io/
22. Rule Interchange Format

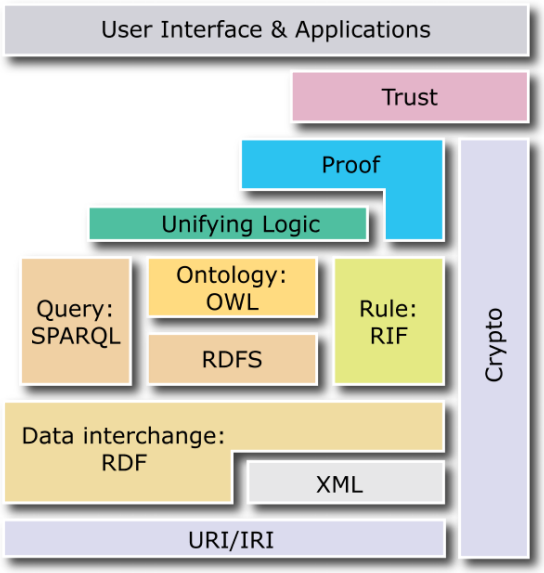


FIGURE 5 – Cake des technologies du Web sémantique

intéressant de proposer une connexion cryptée.La couche "Trust" permet de définir la confiance que l'on peut apporter à une donnée. La couche présente au sommet du cake représente l'application et l'interface utilisant les données.

Certaines couches précisent des technologies spécifiques et sont fondées sur des recommandations du W3C (URI, RDF, RDFS, RIF, SPARQL et OWL) alors que d'autres couches ne sont que des préconisations (Unifying Logic, Proof, Trust, Crypto et User Interface). Il n'existe pas encore de recommandation du W3C pour ces couches.

Nous pouvons prendre l'exemple de la couche "Trust". Nous l'avons vu, le nuage représentant le Web de données liées devient de plus en plus imposant car de plus en plus de données sont ouvertes et partagées de cette manière. Avec l'arrivée d'une telle masse de données apparaît aussi la problématique de la pertinence des données. Un effet similaire est apparu losque le Web de documents a été de plus en plus utilisé. Il est aujourd'hui évident que les documents présents sur le Web peuvent contenir des erreurs. Il en est de même pour les données sur le Web de données liées. La couche "Trust" a donc un rôle particulièrement important à jouer dans le développement du Web de données liées pour permettre la réutilisation de données présentant un niveau de confiance suffisant. Néanmoins, aucune recommandation n'existe à l'heure actuelle concernant cette couche. Il n'existe donc pas de critères permettant de représenter la confiance de manière exhaustive, ni une représentation générique de cette confiance à travers les différents systèmes. C'est aussi le cas de la couche "Proof" sur laquelle se fonde la couche "Trust". Cette couche "Proof" inclut notamment la notion de provenance des données représentées. Il apparaît que la confiance d'une donnée est partiellement liée à sa provenance.

Un verrou majeur existant pour le développement du Web de données liées est la

transformation de sources non ontologiques pour exploiter ces technologies. Il existe sur le Web un grand nombre de sources qui n'exploitent pas directement ces technologies du Web sémantique et donc ne participent pas au Web de données liées. Certaines de ces sources sont le résultat de plusieurs années d'évolution et d'amélioration. Il est donc particulièrement intéressant de les réutiliser.

Nous concentrons l'étude présentée dans ce manuscrit sur ces deux verrous du Web sémantique. Nous cherchons à transformer des sources non-ontologiques pour obtenir des bases de connaissances en considérant plusieurs bases de connaissances simultanément. Cette considération simultanée nous permet de pouvoir découvrir des éléments communs à plusieurs sources. Comme nous l'avons vu, la confiance peut être définie en fonction de la provenance des ressources.

Comme nous allons le présenter dans le prochain paragraphe, l'agriculture est un domaine qui permet d'illustrer ce verrou car il existe un grand nombre de sources non-ontologiques présentes sur le Web, mais le Web de données liées agricole reste encore très peu fourni.

## 2. Domaine d'application : la protection des cultures agricoles

Comme nous venons de le voir, nous orientons l'étude faite dans ce manuscrit sur la transformation de plusieurs sources non-ontologiques pour participer au Web de données liées. Nous avons cité l'exemple du domaine de l'agriculture qui est particulièrement représentatif grâce aux nombreuses sources non-ontologiques existantes.

Dans cette section nous étudierons les besoins du domaine de l'agriculture concernant le Web de données liées. Pour cela, nous étudierons dans un premier temps un besoin concret d'après l'élaboration d'un projet de recherche, besoin auquel le projet Vespa souhaite répondre. Nous étudierons ensuite un échantillon des types de sources de données disponibles dans ce domaine.

### 2.1. Présentation du domaine

Lors du plan Ecophyto [23] et du Grenelle de l'environnement, le gouvernement a engagé l'initiative d'une transition vers une agriculture durable. Cette agriculture préconise l'utilisation de produits phytosanitaires uniquement lorsque cela est nécessaire et non plus de manière systématique. Cette agriculture durable doit permettre la réduction de l'utilisation de produits phytosanitaires dans les cultures agricoles. En effet, un usage trop intensif de produits phytosanitaires peut amener à retrouver des résidus de ces produits dans les récoltes, polluer les eaux ou encore avoir une influence négative sur l'écosystème.

Une des méthodes préconisées pour favoriser la transition vers une agriculture durable est le renforcement de l'épidémiosurveillance. En effet, pour n'utiliser des produits phytosanitaires que lorsque cela est nécessaire, il est indispensable de déterminer les risques d'attaques sur les cultures. En France, des agents ont pour tâche d'observer l'état des cultures et les différentes attaques survenues. Le résultat de ces observations est

23. http://agriculture.gouv.fr/Ecophyto-PSPE

retranscrit dans un document textuel qui paraît plusieurs fois par mois pour chaque région de France : les Bulletins de Santé du Végétal. Néanmoins, ces documents sont textuels et ne permettent donc pas une analyse globale de la situation des attaques de bio-agresseurs en France. Cette analyse pourrait être particulièrement pertinente pour déterminer le développement d'une maladie et donc en déduire son évolution future.

C'est dans ce cadre que se positionne le projet Vespa [24]. Ce projet, dirigé par l'INRA, est consacré en partie à l'archivage et à l'exploitation des BSV dans un système de recherche d'informations. Ce système doit permettre de faciliter l'épidémiosurveillance, notamment à travers un outil : PestObserver.

Nous allons donc tout d'abord étudier ce qu'est un BSV puis nous présenterons l'outil PestObserver. Nous verrons ensuite en quoi l'étude présentée dans ce manuscrit permet l'amélioration de l'outil.

## 2.2. Les Bulletins de Santé du Végétal (BSV)

Une des méthodes de surveillance existant à l'heure actuelle est la rédaction de Bulletins de Santé du Végétal (BSV) [25] par un représentant régional du Ministère de l'agriculture. Ces BSV sont des documents d'information techniques et réglementaires qui doivent être mis à la disposition du public, notamment par l'intermédiaire d'un site Web. Néanmoins, ces BSV sont stockés sur des sites Web différents (un par région). Ils ont pour vocation, entre autres, d'alerter les personnes concernées lors d'apparitions d'agresseurs dans les cultures. Il est donc intéressant pour les agriculteurs ou les conseillers agricoles d'une région d'avoir connaissance des alertes agricoles (AA) des régions limitrophes ou encore de l'évolution des agressions. Il n'existe pas encore, à notre connaissance, de système permettant l'interrogation unifiée des BSV de France.

La figure 6 présente un exemple de BSV. Ce bulletin présente l'observation de plusieurs agresseurs (Sésamie et Pyrale) sur les cultures de maïs en Midi-Pyrénées au 8 juillet 2010. Nous pouvons observer qu'il y a une évaluation des risques de ces agressions mais aucune recommandation sur les techniques de lutte. Cette absence de recommandation est voulue pour ne pas influencer les agriculteurs sur l'utilisation d'un produit plutôt qu'un autre. Les auteurs de ces BSV varient suivant la région ou le type de culture. Cette variabilité des auteurs apporte une diversité sur la présentation des BSV. De plus le rythme des publications suit le développement des cultures et n'est pas régulier. Le rythme de publication d'un BSV pour un type de culture dans une région donnée peut aller d'une parution hebdomadaire à une parution mensuelle. Enfin, les BSV ne sont pas une agrégation automatique de données mesurées mais une synthèse analytique humaine sur des observations.

Ces contraintes apportent une grande diversité sur la forme, le rythme de parution et le vocabulaire employé. Ces diversités en font des documents difficilement exploitables automatiquement.

24. Valeur et optimisation des dispositifs d'épidémiosurveillance dans une stratégie durable de protection des cultures

25. http://agriculture.gouv.fr/Bulletins-de-sante-du-vegetal

CHAMBRE D'AGRICULTURE MIDI-PYRÉNÉES

# BULLETIN DE SANTE DU VEGETAL
## MIDI-PYRENEES

Grandes Cultures - n°28 — 8 juillet 2010

### A retenir

**MAÏS :**

**Pyrale:**

- 1ère génération : vol très étalé avec un pic enregistré entre le 13 juin et le 1er juillet selon les secteurs.
- 2ème génération: pic de vol prévu au cours de la 1ère décade d'août.

**Sésamie :**

2ème génération : le début du vol devrait intervenir entre le 20 et le 25 juillet, avec un pic prévu entre le 26 juillet et le 03 août.

## Maïs

**• Stades phénologiques et état des cultures**

Les températures de ces derniers jours ont fortement accéléré le développement du maïs. Aujourd'hui, le stade moyen est situé entre 14 feuilles à sortie panicule, la floraison est proche pour la majorité des parcelles. Les derniers semis arrivent à 10 feuilles.

**• Sésamie**

Le vol de première génération est terminé. L'observation de pieds de ponte se poursuit. Toutefois ces attaques ne dépassent jamais 1 à 2 % de pieds touchés dans la parcelle. Les piégeages sont peu nombreux pour l'instant.

D'après le modèle, le pic de vol de deuxième génération devrait se situer entre le 26 juillet et le 03 août.

**Évaluation du risque** :

Le risque faible en premier génération se confirme. Toutefois, si le climat reste sec et chaud, les attaques de deuxième génération pourraient être significatives dans les secteurs les plus touchés en 2009.

**• Pyrale**

L'étalement du vol de première génération se confirme. Les piégeages restent significatifs sur la majorité des secteurs. Le pic de vol est dépassé dans l'ensemble de la région; du 13 juin au 1er juillet selon les secteurs.

Cependant, les conditions optimales pour les pontes et leur survie se situaient autour du 15 juin, ce qui devrait donner un maximum de vol de la deuxième génération lors de la première décade d'août.

Le début de vol devrait être significatif au 20 juillet dans les secteurs les plus chauds et au 25 juillet pour les secteurs les plus frais.

Directeur de publication :
Jean-Louis CAZAUBON
Président de la Chambre Régionale d'Agriculture de Midi-Pyrénées
BP 22107 – 31321 CASTANET TOLOSAN Cx
Tel 05.61.75.26.00 – Fax 05.61.73.16.66

Dépôt légal : à parution
ISSN en cours



ARVALIS Institut du végétal — CETIOM — Fredec Midi-Pyrénées

Figure 6 – Exemple d'un BSV sur le maïs en Midi-Pyrénées

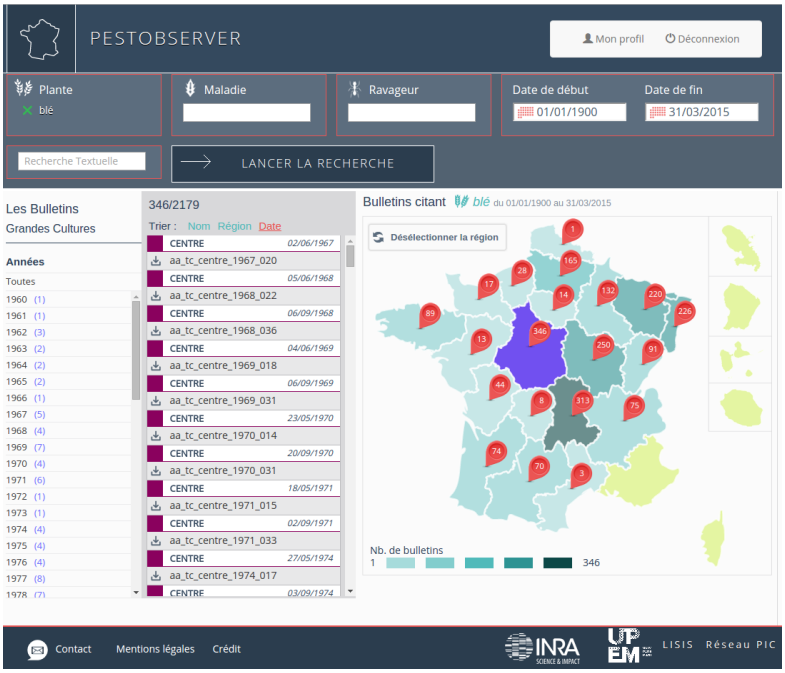


FIGURE 7 – Exemple de l'utilisation de l'outil PestObserver [Turenne et al., 2015]

## 2.3. PestObserver

Dans le cadre du projet Vespa dirigé par l'INRA, le laboratoire Irstea de Clermont-Ferrand est un contractuel pour l'archivage des BSV. Cet archivage doit permettre d'extraire un ensemble d'informations sur les cultures et les niveaux d'attaques de ces cultures au cours du temps.

Dans ce projet, est développé par le laboratoire LISIS [26] de Paris un outil permettant l'interrogation des Bulletins d'alertes agricoles archivés dont les BSV font partie : PestObserver [27] [Turenne et al., 2015]. Cet outil permet d'effectuer des recherches par mots-clefs sur les BSV archivés. Pour cela, il est possible de spécifier le type de plante à rechercher, le type de ravageur (l'agresseur est dans ce cas un insecte ou un animal) ou la maladie (l'agresseur est dans ce cas généralement un champignon ou une bactérie). Il est aussi possible de filtrer les résultats de la recherche sur une période donnée. Le résultat obtenu est un ensemble de bulletins répondant aux critères définis, représentés sur une carte pour savoir quelles régions sont concernées.

La figure 7 présente un exemple de l'utilisation de cet outil. Nous avons fait une recherche sur la plante de type "blé". Nous pouvons observer sur la carte présentée au centre que 346 bulletins concernent la région Centre sur la période entre 01/01/1977 et 31/03/2015. Il est ensuite possible de trier les résultats par date ou par région ou encore de limiter les résultats suivant une date ou une région donnée.

26. Laboratoire Interdisciplinaire Sciences Innovations Sociétés
27. http://pestobserver.eu/

Ce système est assimilable à un système de recherche d'information en texte intégral. Les bulletins numérisés ont été indexés. L'outil PestObserver interroge cet index pour générer les résultats. Néanmoins, l'indexation de document se fait en fonction du contenu textuel du document. En d'autres termes, notre requête sur les bulletins traitant du "blé" ne retournera que ceux contenant le terme exact "blé" dans leur contenu. Il ne traitera pas les bulletins contenant le terme "Triticum" (terme scientifique pour définir le blé), ni ceux contenant le terme "Triticum Durum" qui est le blé dur, espèce du genre Triticum. De la même façon, si nous cherchons une plante et une maladie, alors le résultat sera un ensemble de bulletins contenant les termes associés. Les auteurs de l'outil PestObserver ont résolu le problème en proposant un dictionnaire des termes et leurs synonymes. Néanmoins, le dictionnaire étant construit manuellement, il n'est pas raisonnable de le faire pour tous les termes pouvant apparaître dans les BSV. De plus, la gestion des synonymes n'est pas suffisante puisque nous l'avons vu, l'indexation des termes plus spécifiques est aussi nécessaire.

C'est dans ce cadre que l'Irstea de Clermont-Ferrand souhaite, en plus de l'archivage, effectuer une annotation sémantique des BSV à partir d'une base de connaissances [Roussey and Bernard, 2015]. Une telle annotation repose sur les technologies du Web sémantique. Elle permettrait d'interroger les textes non seulement à partir des chaînes de caractères qui les composent mais aussi à partir des sens associés aux ressources ontologiques (classes, propriétés, individus, relations) qu'ils abordent. Pour effectuer une telle annotation, il est nécessaire d'avoir une base de connaissances regroupant toutes les connaissances pouvant être utilisées pour l'annotation. De la même manière qu'il n'était pas envisageable de construire manuellement un dictionnaire exhaustif, construire une base de connaissances exhaustive du domaine de l'observation d'attaques dans les cultures ne l'est pas plus. Néanmoins, l'avantage de ce domaine est qu'il existe un grand nombre de sources non-ontologiques pouvant être exploitées. Il nous paraît donc possible de tendre vers une certaine exhaustivité.

En plus de pouvoir être utilisée dans l'outil PestObserver, une telle source peut avoir un intérêt majeur dans l'évolution des pratiques agricoles. En effet, l'annotation sémantique des BSV peut permettre de mettre à disposition les données présentes dans les BSV sous un format adapté au Web de données liées. Il devient alors possible d'effectuer des analyses croisées avec d'autres sources présentes sur ce Web de données liées. Ce décloisonnement des observations des attaques peut donc être un avantage non négligeable pour réduire l'usage des produits phytosanitaires. De plus, il est possible d'utiliser des outils du Web sémantique pour interroger ces données [Pradel, 2013].

## 2.4. Types de sources agricoles

Afin de générer la base de connaissances qui pourra permettre l'annotation sémantique des BSV, il est nécessaire d'analyser les différentes sources disponibles. C'est l'objectif de cette section qui nous permettra d'avoir une vue globale sur les sources existantes et leur format.

En s'appuyant sur les travaux présentés dans [Villazon-Terrazas et al., 2010], nous analysons les sources selon trois caractéristiques :

**Le type de source** représente l'objectif de la modélisation de la source,

**Le modèle** représente l'organisation utilisée pour représenter la source,

**L'implémentation** est le format de représentation concret dans un fichier.

Ces niveaux permettent de faire la distinction entre l'objectif de la modélisation (le type de source), la façon dont la modélisation est effectuée (le modèle) et la façon dont les données sont stockées (l'implémentation). Les données publiées sur le Web de données liées, et donc repréentées en RDF, sont plus facilement accessibles.

Dans ce cadre, nous définissons un ensemble de types de sources existant pour le domaine de l'agriculture. Ces sources sont présentées selon leur type.

### 2.4.1. Taxonomies

Le terme "taxonomie" provient du mot grecque "taxinomia" qui est composé du terme "taxis" (classement) et du terme "nomos" (loi). Le terme est apparu en 1863 dans l'ouvrage "Théorie élémentaire de la botanique ou exposition des principes de la classification naturelle et de l'art de décrire et d'étudier les végétaux" [de Candolle, 1813] écrit par Augustin Pyrame de Candolle. Initialement, une taxonomie était consacrée au recensement et à la catégorisation des organismes vivants. De nos jours, une taxonomie peut permettre la catégorisation d'autres domaines.

Dans ce manuscrit, nous considérerons qu'une taxonomie est le classement des êtres vivants. Elle permet de représenter chaque catégorie d'être vivant sous la forme de taxons avec des liens hiérarchiques entre taxons. Un taxon représente une catégorie d'être vivant plus ou moins spécifique. Ce taxon peut être désigné par plusieurs termes.

#### NCBI Taxonomy

Un exemple de taxonomie utilisée dans le domaine de l'agrnomie est la NCBI Taxonomy [28]. Cette taxonomie n'est pas uniquement consacrée à la catégorisation des types de cultures mais aussi à la catégorisation de tous les êtres vivants associés à des séquençages d'ADN. Elle contient un très grand nombre de taxons (457.110). Historiquement, le NCBI avait pour mission de récolter et de référencer le séquençage de fragments d'ADN. Les outils de séquençage de fragments d'ADN se sont particulièrement développés au début des années 1990. En raison de la quantité de plus en plus grande de résultats de séquençage à stocker, des bases de données ont été mises en place pour centraliser ces séquençages. NCBI est l'organisme qui a été en charge du développement et du maintien de la base de données sur le continent nord américain. L'équivalent a été mis en place en Europe par l'EBI [29], mais l'EBI ne maintient pas de taxonomie. Afin de catégoriser les êtres vivants associés aux différents séquençages d'ADN, le NCBI a réutilisé sa base de données des génomes (GenBank [30]) pour créer et maintenir à jour sa taxonomie des organismes vivants. La taxonomie du NCBI est devenue incontournable dans le domaine de la biologie. Bien que le NCBI précise que leur taxonomie n'a pas vocation à devenir un référentiel dans le

28. The National Center for Biotechnology Information - http://www.ncbi.nlm.nih.gov/taxonomy
29. European Bioinformatics Institute - https://www.ebi.ac.uk
30. http://www.ncbi.nlm.nih.gov/genbank/

| | NCBI Taxonomy |
|---|---|
| **Type de source** | Taxonomie |
| **Modèle** | Base de données |
| **Implémentation** | TSV |

TABLE 1 – Caractéristiques de NCBI Taxonomie

domaine du vivant, dans la pratique, les experts du domaine font souvent référence à cette taxonomie, à défaut d'en avoir une qui soit exhaustive.

De par son lien avec GenBank, NCBI Taxonomy repose sur une modélisation que l'on appelle APG-III [APG, 2009]. Cette classification repose sur une catégorisation des taxons par similitudes génétiques. Elle est par conséquent à différencier de la classification Cronquist [Cronquist, 1981]. Cette dernière est fondée sur une catégorisation par similitude phénotypique. Ces classifications sont, toutes deux, utilisées dans le domaine de la biologie.

L'accès à la NCBI Taxonomy peut se faire de deux façons. La première est l'utilisation d'une interface Web permettant la navigation dans la taxonomie suivant la hiérarchie de taxons. La navigation peut être assistée par une interface d'interrogation par mots-clefs (nommée ici Entrez)[Sayers, 2009]. Cette interface est utile d'un point de vue utilisateur, puisqu'elle facilite l'accès à la taxonomie, même pour des utilisateurs non-informaticiens. Par exemple ici, `http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=4567` nous accédons à la page associée à "Triticum Durum", le blé dur. Néanmoins, un des aspects qui nous intéresse est la ré-exploitation de cette source. Pour cela, il existe aussi un format d'exportation de la taxonomie permettant de la manipuler directement. Le NCBI propose un accès FTP à ses fichiers sources contenant la taxonomie dans son intégralité [31].

Le modèle utilisé par cette source est une base de données. Chaque fichier représentant une table, des identifiants sont utilisés dans ces fichiers pour faire le lien entre les fichiers. Le format d'implémentation utilisé ici ressemble à du CSV [32][Shafranovich, 2005], à ceci près que la virgule est remplacée par la suite de caractère "tabulation barre verticale tabulation". La description détaillée de l'organisation des fichiers de la taxonomie est accessible à cette adresse `ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump_readme.txt`. Nous pouvons remarquer que beaucoup d'informations sont en relation avec le séquençage génétique des organismes vivants. De plus, les termes utilisés pour désigner un organisme sont séparés dans un deuxième fichier (names.dmp), le lien se faisant par un id unique. Il est possible de découvrir les liens hiérarchiques entre taxons en utilisant l'attribut "parent tax_id" et il est également possible d'identifier le rang taxonomique grâce à l'attribut "rank". La division permet directement de filtrer les organismes vivants de type plante. De cette manière, nous pouvons récupérer toute la taxonomie des plantes provenant de la NCBI Taxonomy.

31. `ftp://ftp.ncbi.nih.gov/pub/taxonomy`
32. Coma Separated Values

### TaxRef

Contrairement à la NCBI Taxonomy, TaxRef est une taxonomie qui a vocation à devenir un référentiel national[Gargominy et al., 2014]. Alimentée et administrée par le MNHN [33], l'objectif de cette taxonomie est de répertorier tous les noms scientifiques et synonymes pouvant permettre l'identification d'un organisme vivant en France. Après avoir observé une augmentation du nombre d'organismes vivants et la prolifération des référentiels existants, le MNHN a décidé de créer un référentiel national, en respectant la nomenclature des différents référentiels internationaux déjà existants. De cette manière, le partage des données concernant la taxonomie des organismes vivants devient plus simple, non seulement en France mais aussi dans le monde entier. Pour alimenter cette taxonomie, le MNHN réutilise différentes bases de données validées et officielles (WoRMS [34], base Nadeaud [35], ...) mais aussi les publications scientifiques. Pour garantir leur statut de référence et donc avoir une qualité optimum, ces réutilisations se font manuellement par un ensemble d'experts. Comme cette source est une référence, seuls les taxons définis dans la litérature scientifique sont considérés. Bien qu'un terme puisse ne pas être considéré comme le taxon scientifique de l'organisme vivant, il se peut que, dans la littérature scientifique, il soit couramment utilisé. De cette manière, il est renseigné dans la taxonomie TaxRef. C'est pour cela que cette taxonomie comporte aussi bien des noms vernaculaires [36] que des noms scientifiques.

Le MNHN propose une interface Web pour parcourir la taxonomie à partir de requêtes par mots-clefs. De cette manière, nous accédons à une page associée à l'organisme vivant proposant un certain nombre d'informations, telles que les localisations de cultures ou d'élevage, les différents termes associés et d'autres. Par exemple `http://inpn.mnhn.fr/espece/cd_nom/141978` la page associée au "Triticum Durum". Nous retrouvons notamment sur cette page le nom vernaculaire associé à cette espèce : le blé d'Afrique. De la même manière que pour la NCBI Taxonomy, il est intéressant de pouvoir manipuler directement la taxonomie. C'est pour cette raison que, après une inscription et une justification de l'utilisation de ce référentiel, un fichier est proposé au téléchargement contenant la taxonomie [37].

Là encore, pour cette source, le modèle utilisé est la base de données. Pour simplifier l'exploitation de cette taxonomie, le choix a été fait de renseigner tous les taxons dans une seule table et de définir uniquement les valeurs possibles dans des tables associées (c.f. figure 8). L'implémentation de ce fichier est effectuée dans un format similaire au CSV, mais séparant les valeurs par des caractères "|".

33. Muséum National d'Histoire Naturelle
34. World Register of Marins Species - `http://marinespecies.org/`
35. `http://inpn.mnhn.fr/espece/inventaire/I220`
36. "Nom vulgaire d'animal ou de végétal, par opposition aux noms qui suivent les règles de la nomenclature scientifique" - `http://www.cnrtl.fr/definition/vernaculaire`
37. `http://inpn.mnhn.fr/telechargement/referentielEspece/referentielTaxo`

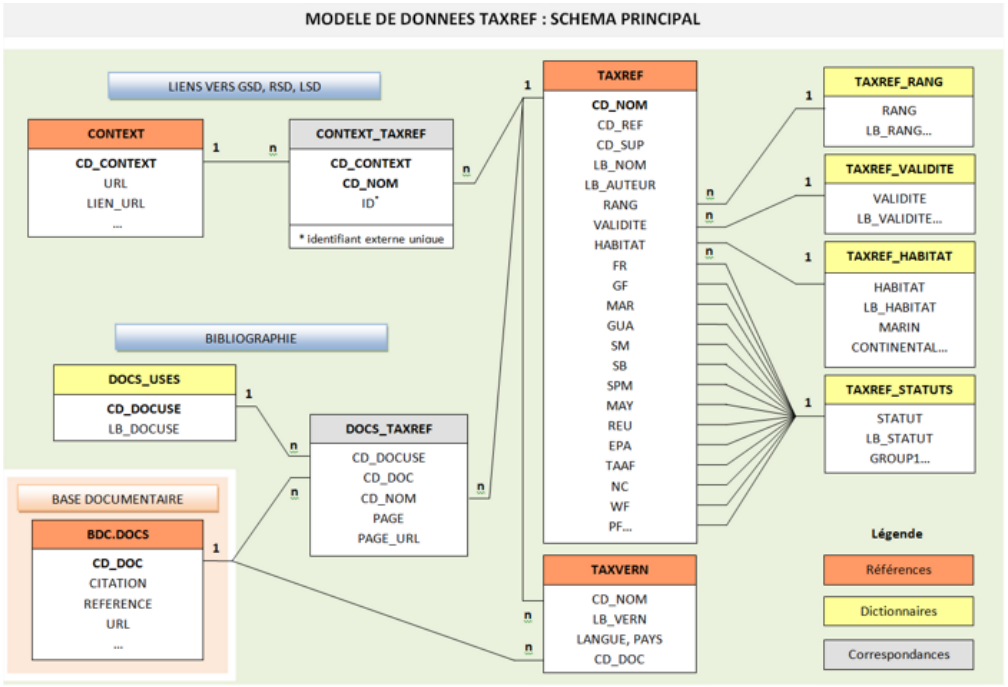


FIGURE 8 – Modèle des données de la taxonomie TaxRef

| | **TaxRef** |
|---|---|
| **Type de source** | Taxonomie |
| **Modèle** | Base de données |
| **Implémentation** | Valeurs séparées par "\|" |

TABLE 2 – Caractéristiques de TaxRef

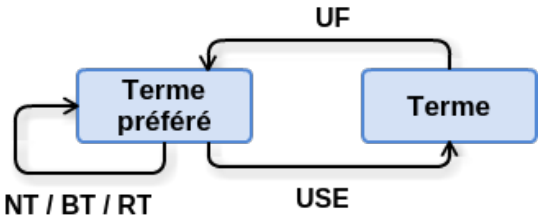


FIGURE 9 – Structure d'un thésaurus ISO 2788

### 2.4.2. Thésaurus

Un thésaurus est un vocabulaire contrôlé qui permet de faciliter l'indexation de documents. Bien que le terme de thésaurus soit apparu en 1532 avec l'ouvrage "Thesaurus linguæ latinæ" de Robert Estienne, les thésaurus se développent considérablement au début des années 1990 avec l'avènement des outils informatiques. Vue la quantité de documents à parmi lesquels rechercher, l'usage d'un vocabulaire contrôlé s'est imposé pour améliorer les resutats des recherches. D'après [Roussey, 2001], nous pouvons définir un thésaurus de la manière suivante : "Lexique des termes d'indexation du langage documentaire avec leurs relations structurant le domaine. Un thésaurus organise le vocabulaire d'un langage documentaire contrôlé pour que des relations entre regroupement de termes (par exemple terme générique et terme spécifique) soient explicitées." Nous pouvons observer dans cette définition la notion de regroupement de termes. Les termes regroupés sont des synonymes, ce qui permet de faciliter la désambiguïsation lors de l'indexation d'un document. De plus, ces regroupements de termes sont liés entre eux pour définir, par exemple, des relations hiérarchiques.

Il existe principalement deux modèles pour représenter les thésaurus. Le premier, correspondant à la norme ISO 2788, ne représente pas explicitement les regroupements de termes. Le deuxième, correspondant au SKOS, les représente explicitement.

La première norme définissant un modèle représentant un thésaurus est la norme ISO 2788 de 1986[ISO 2788, 1986]. Cette norme est fondée sur la représentation de relations entre termes. Un thésaurus suivant la norme ISO 2788 est donc assimilable à un graphe étiqueté et orienté comme présenté dans la figure 9.

Un thésaurus suivant la norme ISO 2788 contient donc des relations entre les termes, pouvant être étiquetées de cinq façon différentes :

**NT (Narrower Term) :** relation hiérarchique descendante

**BT (Broader Term) :** relation hiérarchique ascendante

**RT (Related Term) :** relation d'association

**UF (Used For) :** relation d'équivalence entre termes indiquant le terme préféré

**USE (Use) :** relation d'équivalence entre termes

Les relations font apparaître des regroupements de termes. En effet, USE étant utilisé pour désigner les synonymes d'un terme, tous les synonymes sont regroupés autour d'un terme préféré qui est représentatif du regroupement de terme. Ce terme préféré

est appelé un descripteur. Ces regroupements de termes peuvent être mis en relation par l'intermédiaire de trois relations différentes (NT, BT et RT). Les relations NT et BT représentent une notion de hiérarchie. Bien qu'un ordre hiérarchique entre les regroupements de termes existe, ces relations restent ambiguës car le sens exact qui y est associé n'est pas défini. En effet, les thésaurus ayant pour but, à cette époque, de n'être utilisés que par des humains (et non des machines), une certaine liberté a été laissée dans la modélisation de ces relations. Une relation NT (respectivement BT) peut représenter une relation de méronymie (respectivement métonymie), une relation d'hyperonymie (respectivement hyponymie) ou autre. Les connaissances de la personne manipulant le thésaurus permettront de désambiguïser ces relations. La relation RT est d'autant plus ambiguë qu'elle ne spécifie aucune autre information qu' "une relation existe entre ces termes". Cette relation RT peut donc avoir de nombreuses interprétations.

Dans le but de promouvoir les thésaurus à l'échelle du Web de données liées (c.f. section 1.3 du chapitre I), le W3C [38] a mis en place une recommandation proposant un modèle pour les thésaurus en exploitant la syntaxe RDF : SKOS [39][Miles and Bechhofer, 2009]. Ce modèle de thésaurus propose une représentation explicite des regroupements de termes des thésaurus sous la forme de concepts SKOS. La recommandation SKOS permet d'exprimer les mêmes relations hiérarchiques que la nome ISO 2788 mais entre concepts SKOS. De plus, elle propose la possibilité de définir des relations spécifiques du domaine pouvant lier deux concepts SKOS du thésaurus. Ainsi, les ambiguités liées à l'usage de la relation RT sont levées. Cette ambiguïté est aussi réduite pour la définition des synonymes qui peuvent être définis comme terme préféré ou terme alternatif. Ils peuvent aussi être représentés en plusieurs langues de façon explicite. De plus, cette recommandation permet de rendre les thésaurus adaptés au Web de données liées. De cette manière, il est possible de réutiliser des ressources issues d'autres sources de données publiées sur ce Web de données liées.

Nous pouvons noter qu'il existe aussi la norme ISO 25964 [25964-1, 2011] comme modèle possible pour la représentation de thésaurus. Néanmoins, ce modèle est particulièrement proche de la recommandation SKOS. De plus, SKOS proposant une intégration au Web de données liées, il est souvent privilégié par rapport à la norme ISO 25964.

#### Agrovoc

Dans le domaine agricole, un thésaurus particulièrement réputé est le thésaurus Agrovoc [40]. Ce thésaurus est implémenté en utilisant le format SKOS présenté précédemment.

Le développement du thésaurus Agrovoc a commencé, tout d'abord sous un format papier, dans le début des années 1980 par la FAO [41]. Le but de ce thésaurus était de proposer un vocabulaire pour l'indexation de ressources dans le domaine de l'agriculture et de l'alimentation. Au début des années 2000, la FAO arrêta la version papier pour passer à une version numérique sous forme d'une base de données spécifique. En 2004, ils

38. World Wide Web Consortium - http://www.w3.org/
39. Simple Knowledge Organization System - http://www.w3.org/TR/skos-reference/
40. http://aims.fao.org/aos/agrovoc/
41. Food and Agriculture Organization of the United Nations - http://www.fao.org/home/fr/

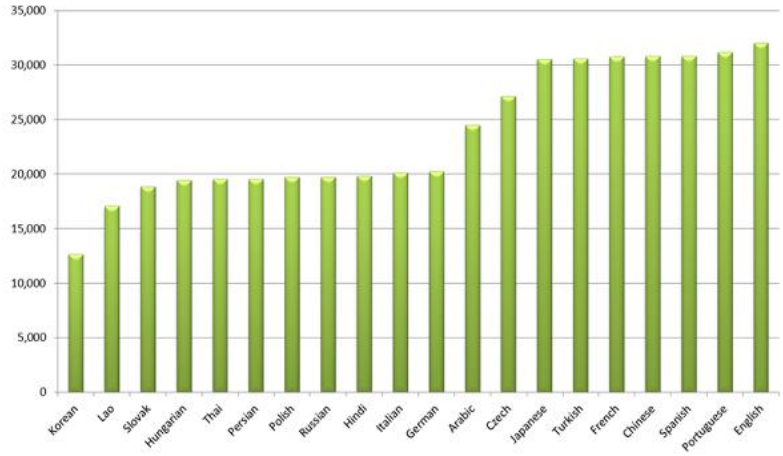


FIGURE 10 – Agrovoc : nombre de termes pour chaque langue

expérimentèrent le passage à une version OWL (c.f. section 1.2.3 du chapitre I) qu'ils abandonnèrent pour passer à une version SKOS en 2009, le niveau de d'expressivité de SKOS étant suffisant puisque l'objectif était simplement d'avoir un vocabulaire d'indexation.

Agrovoc utilise une spécialisation du modèle SKOS présenté précédemment : le SKOS-XL [42]. Ce modèle permet l'ajout de relations entre les termes eux-mêmes (et non nécessairement par l'intermédiaire de concepts SKOS). De plus, la FAO a étendu le vocabulaire disponible en utilisant l'ontologie Agrontology [43], elle-même développée par la FAO. Cette ontologie définit un certain nombre d'éléments supplémentaires pour enrichir le vocabulaire. Nous trouvons notamment la relation "pestOf" [44] qui permet d'établir un lien entre deux organismes vivants représentant le fait qu'un organisme peut potentiellement en attaquer un autre. Le format d'implémentation est le RDF.

La force du thésaurus Agrovoc est le nombre de concepts répertoriés (plus de 40.000), mais surtout le nombre de langues différentes dans lesquelles les étiquettes des concepts sont disponibles. Nous pouvons observer sur la figure 10 le nombre de termes par langue.

La maintenance du thésaurus Agrovoc se fait tout d'abord par les experts de la FAO, mais aussi grâce à une communauté d'utilisateurs qui proposent des ajouts et autre modifications. De plus, ce thésaurus est lié sur le Web de données liées à 16 sources différentes [45].

42. http://www.w3.org/TR/skos-reference/skos-xl.html
43. http://aims.fao.org/sites/default/files/uploads/file/aos/agrontology/index.htm
44. http://aims.fao.org/sites/default/files/uploads/file/aos/agrontology/index.htm#d4e1950
45. http://aims.fao.org/standards/agrovoc/linked-open-data

| | Agrovoc |
|---|---|
| **Type de source** | Thésaurus |
| **Modèle** | SKOS |
| **Implémentation** | RDF |

TABLE 3 – Caractéristiques d'Agrovoc

### 2.4.3. Bases de données relationnelles

La notion de bases de données relationnelles (BDR) est apparue en 1970 [Codd, 1970]. Une BDR est une base de données exploitant la théorie de l'algèbre relationnelle. Dans ce modèle une table est une "relation" (au sens de l'algèbre relationnelle). Les données sont organisées en différentes tables (ou relations) reliées entre-elles par des relations.

Nous considérons l'étude d'une BDR comme type de source. Néanmoins, les BDR peuvent aussi être un modèle pour un autre type de sources. En d'autres termes, une BDR peut être utilisée pour représenter un autre type de source comme un thésaurus par exemple. Mais elle peut aussi, dans certains cas, être un type de source à part entière. C'est notamment le cas lorsque nous souhaitons exploiter la modélisation d'une BDR.

#### BDR Arvalis

Nous avons évoqué précédemment les Bulletins de Santé du Végétal (BSV) dans lesquels sont présentées des alertes agricoles à des fins de protection des cultures en France. Ces BSV s'appuient sur la base de données Arvalis. Le type de cette source est donc une base de données, le modèle utilisé est la base de données relationnelle (BDR) et le format d'implémentation est CSV. Cette BDR contient toutes les observations sur les cultures. Ces observations contiennent non seulement la culture attaquée, l'agresseur et la date de l'agression, mais aussi le stade de développement de la culture, l'identité de l'observateur, le niveau de risques, etc. Nous pouvons voir sur la figure 11 l'ensemble des caractéristiques enregistrées pour chaque observation.

Un des intérêts de cette base de données (autre que le regroupement des données concernant les observations d'attaques) est l'absence de recommandations pour permettre d'éviter les traitements dits automatiques. Les alertes agricoles ne sont là que pour avertir l'agriculteur d'un risque d'attaque de bio-agresseurs dans sa culture, mais ne donnent aucune indication sur la façon de lutter contre ces agresseurs. Précédemment, les observations étaient faites par des producteurs de produits phytosanitaires préconisant systématiquement leurs produits. Grâce à la base de données Arvalis, et aux BSV de manière générale, les agriculteurs peuvent être informés des risques sans être orientés automatiquement sur un produit phytosanitaire.

### 2.4.4. Base de connaissances

Comme vu précédemment, Agrovoc a fait le choix d'utiliser SKOS comme formalisme pour se placer sur le Web de données liées. Pour étendre le vocabulaire utilisé afin de

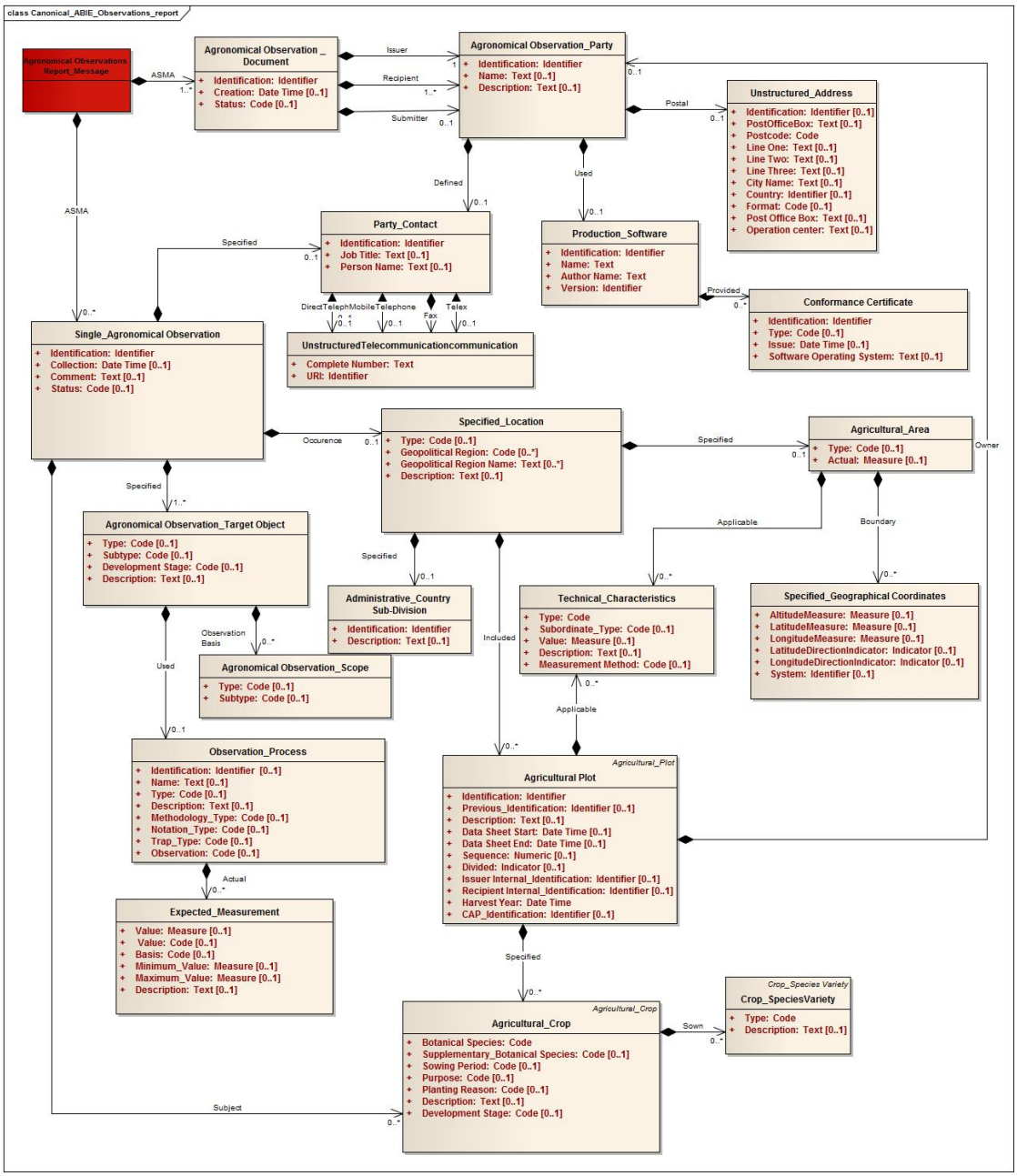


FIGURE 11 – Base de données relationnelle Arvalis

| | Arvalis |
|---|---|
| **Type de source** | Base de données |
| **Modèle** | BDR |
| **Implémentation** | CSV |

TABLE 4 – Caractéristiques de la BDR Arvalis

désambiguïser certaines relations, une ontologie a été définie. La définition d'une ontologie (Cf. section 1.2.3 du chapitre I) pose le vocabulaire qui sera utilisé pour décrire des faits formant une base de connaissances.

### AgroPortal

Les bases de connaissances présentes sur le Web de données liées ne sont pas indexées par un moteur de recherche classique. Il est donc plus difficile de trouver une base de connaissances adaptée à nos besoins que de trouver un document indexé. Néanmoins, certains travaux essaient de pallier ce problème. Nous pouvons citer en particulier AgroPortal [46] [Jonquet et al., 2015a] qui propose une plate-forme permettant d'entreposer des bases de connaissances en rapport avec le domaine de l'agronomie.

Cette plate-forme, qui utilise la technologie de la plate-forme BioPortal [Noy et al., 2009, Jonquet et al., 2015b], permet non seulement de stocker des bases de connaissances mais aussi tous jeux de données au format RDF. Cet entrepôt permet un stockage pérenne, ainsi qu'une annotation de ces jeux de données afin de pouvoir les retrouver à partir d'une requête par mots-clefs.

Nous avons alors utilisé ce portail pour chercher des bases de connaissances adaptées à nos besoins, deux se sont révélées pertinentes.

### Plant Ontology

La Plant Ontology (PO) [Avraham et al., 2008] a pour objectif de fixer un vocabulaire commun concernant la génétique des plantes. Cette ontologie est donc orientée sur la formalisation de la connaissance des gènes des plantes, mais aussi des différents stades de développement et des caractéristiques phénotypiques [47] de celles-ci.

La PO est en fait un regroupement de plusieurs ontologies décrivant des caractéristiques différentes des plantes (génotypiques ou phénotypiques, par exemple). Historiquement, la PO a été formalisée sous un format OBO [48]. Ce format est proche de celui utilisé pour les ontologies telles que nous les définissons dans ce manuscrit (Cf. section 1.2.3 du chapitre I). Ce format OBO est utilisé dans le domaine de la biologie et de l'agronomie pour représenter des connaissances. Il est néanmoins de moins en moins utilisé au profit du format OWL. La PO est disponible avec le modèle OBO.

46. http://agroportal.lirmm.fr/
47. d'apparence physique
48. The Open Biological and Biomedical Ontologies - http://www.obofoundry.org/

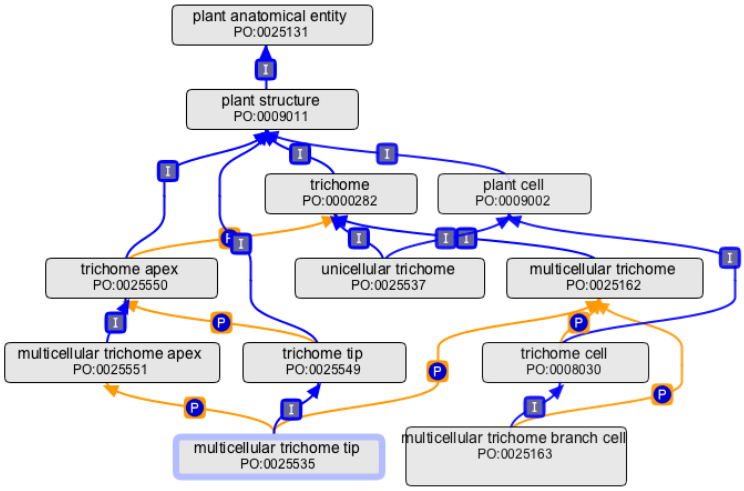


FIGURE 12 – Sous parti de The Plant Ontology (relation "I" : is_a, relation "P" : part_of)

| | **Plant Ontology** |
|---|---|
| **Type de source** | Base de connaissances |
| **Modèle** | OBO |
| **Implémentation** | OBO Flat File [49] |

TABLE 5 – Caractéristiques de la Plant Ontology

Le développement de cette ontologie a débuté en 2002 en réutilisant des bases de données existantes concernant les détails phénotypiques du riz et du maïs. La Plant Ontology Consortium (POC) a ensuite mis en place un système de mise à jour communautaire pour cette ontologie. Les utilisateurs peuvent proposer des améliorations ou des modifications de l'ontologie, qui sont ensuite validées par le consortium avant d'être ajoutées dans l'ontologie. De cette manière, l'ontologie a évolué pour intégrer des connaissances pour d'autres genres de plantes.

La figure 12 montre une sous-partie de la PO qui présente la modélisation pour la structure d'une plante.

### Crop Ontology

Le développement de la Crop Ontology (CO)[Shrestha et al., 2010] a débuté en 2010 par la Generation Challenge Programme [50]. L'objectif de ce programme est de permettre l'amélioration des conditions de culture pour les agriculteurs en milieu difficile comme par exemple les regions arides. La CO participe à cet objectif en proposant un vocabulaire

50. http://www.generationcp.org/

| | Crop Ontology |
|---|---|
| **Type de source** | Base de connaissances |
| **Modèle** | OBO |
| **Implémentation** | OBO Flat File |

TABLE 6 – Caractéristiques de la Crop Ontology

commun entre les différents acteurs de ce programme, afin de partager des informations concernant les cultures. Il est par exemple possible de catégoriser des mesures dans les cultures et de croiser ces mesures avec d'autres.

La CO se concentre sur la représentation des connaissances concernant le phénotype et l'anatomie des plantes. Elle permet aussi de faire le lien avec des données de mesures effectuées dans les cultures. Le génotype des cultures est aussi présent dans la CO afin de pouvoir faire le lien entre le phénotype, une observation et le gène impliqué. Le modèle utilisé dans cette source est OBO avec RDF comme format d'implémentation.

L'avantage de cette ontologie est son accès simplifié par un des Web services disponibles pour rechercher des éléments [51]. Cette API permet d'imaginer de nombreuses applications possibles pour la réutilisation de cette ontologie afin d'améliorer l'analyse des cultures.

Toujours dans l'objectif de faciliter la manipulation, une interface Web est aussi proposée afin d'annoter automatiquement des documents au format CSV [52] avec la CO.

### DBPedia

DBPedia [53][Auer et al., 2007] est une base de connaissances fondée sur l'exportation de données extraites de Wikipedia [54]. Partant du constat que Wikipedia est une source contenant un nombre important d'informations à jour, le projet DBPedia a pour objectif de transformer ces informations en une base de connaissances. Pour cela, les auteurs de DBPedia exploitent le contenu de ce que l'on appelle des infobox, qui sont les tableaux situés à droite des pages Wikipedia contenant des données. De cette manière, l'extraction de triplets est simplifiée et la base de connaissances DBPedia peut être enrichie facilement. Le modèle choisi pour représenter cette base de connaissances est OWL et le format d'implémentation est RDF.

Nous pouvons observer sur la figure 13 une partie de l'infobox de la page "Triticum Durum" [55] de Wikipedia. La hiérarchie taxonomique liée au blé dur peut donc être extraite à partir de cette infobox sous une forme de triplets pouvant venir enrichir DBPedia. Le résultat est disponible à cette adresse : `http://dbpedia.org/page/Durum`.

La base de connaissances DBPedia est, à l'heure actuelle, largement utilisée comme référence pour l'alignement. Toutes les bases de connaissances essaient d'être alignées

51. `http://www.cropontology.org/api`
52. Coma Separated Value - `https://fr.wikipedia.org/wiki/Comma-separated_values`
53. `http://dbpedia.org/`
54. `https://fr.wikipedia.org/`
55. `https://fr.wikipedia.org/wiki/Bl%C3%A9_dur`

| | DBPedia |
|---|---|
| **Type de source** | Base de connaissances |
| **Modèle** | OWL |
| **Implémentation** | RDF |

TABLE 7 – Caractéristiques de DBPedia

avec DBPedia afin de l'utiliser comme pont entre les différentes sources. L'avantage de cette source est qu'elle contient énormément d'individus sur un très grand nombre de domaines. Néanmoins, la politique de Wikipedia suivant le principe participatif et l'auto-correction par la communauté, un certain nombre d'erreurs et d'approximations peuvent apparaître.

## 2.5. Bilan sur les sources agricoles

Comme nous venons de le voir, la transition vers une agriculture durable apporte un certain nombre de problématiques et notamment des besoins d'analyse des données. Afin de pouvoir réduire l'utilisation de produits phytosanitaires, il est nécessaire de pouvoir analyser les données concernant les observations d'attaques de bio-agresseurs dans les cultures et de les croiser avec d'autres. Il est par exemple intéressant de pouvoir croiser une observation d'attaque agricole avec différentes techniques de lutte possibles contre cette attaque. Pour cela, le projet VESPA s'intéresse au stockage et à l'indexation des BSV pour faciliter l'analyse des attaques survenant dans les cultures françaises. L'indexation des BSV se faisant par un dictionnaire maintenu manuellement, il est difficile de pouvoir couvrir l'intégralité du vocabulaire présent dans les BSV. Il est intéressant de pouvoir créer un vocabulaire d'indexation avec les différentes sources présentes dans ce domaine. La création de ce vocabulaire sous la forme d'une base de connaissances pouvant intégrer le Web de données liées agricole est un atout maître pour permettre le décloisonnement de ces données.

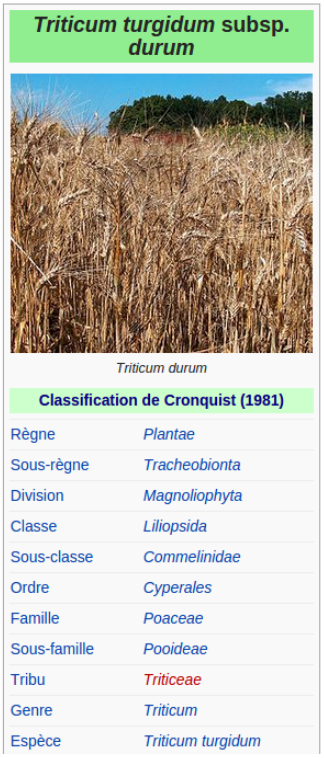


FIGURE 13 – Info-box Blé Dur Wikipedia

Après avoir étudié les différents types de sources disponibles, nous avons pu noter deux points importants. Le premier est l'hétérogénéité des types de sources disponibles dans ce domaine. En effet, nous avons pu observer qu'il existe des thésaurus, des taxonomies, des bases de données ou encore des bases de connaissances

pouvant être considérés dans la création de ce vocabulaire. Le deuxième aspect est la complémentarité des sources, en particulier au niveau de leur qualité. Certaines sources peuvent être particulièrement fournies mais ne garantissent pas la qualité des éléments représentées, alors que d'autres sources garantissent une certaine qualité mais sont moins exhaustives. Nous pouvons prendre l'exemple d'Agrovoc qui propose un grand nombre de labels mais qui peut aussi contenir un certains nombres d'erreurs [Soergel et al., 2004], alors que TaxRef est une référence en matière de taxonomie des organismes vivants en France mais contient bien moins de labels.

Il est alors particulièrement pertinent, dans ce cas, de pouvoir considérer plusieurs sources dans un processus de construction d'une base de connaissances. De cette manière, il serait possible d'exploiter la quantité et la complémentarité des sources présentes dans le domaine de l'agriculture.

## 3. Méthodologie de construction de base de connaissances : NeOn

Afin de pouvoir construire une base de connaissances à partir de sources non-ontologiques, il est nécessaire de suivre une méthodologie définissant les différentes étapes nécessaires de ce processus. Une telle méthodologie considère tout le processus de création, de la définition des besoins jusqu'à la validation du résultat et la mise à disposition.

Afin de déterminer la méthodologie à utiliser, nous avons défini trois critères importants que doit permettre la méthodologie. Le premier critère est la possibilité de réutiliser des sources non-ontologiques dans le processus de création. Nous souhaitons pouvoir exploiter la quantité de sources disponibles dans le domaine de l'agriculture, comme nous l'avons observé dans la section 2.4 du chapitre I.

Le deuxième critère est la réutilisation de patrons de conception dans le processus afin d'améliorer la qualité de la modélisation. Il est courant de considérer, dans le domaine du génie logiciel, la réutilisation de patrons de conception. Il en est de même pour l'ingénierie des connaissances. La réutilisation de patrons de conception dans la modélisation d'une ontologie permet de faciliter le travail de modélisation en s'assurant d'une certaine qualité. Ces patrons de conception permettent aussi de faciliter la mise en correspondance des ressources de différentes bases de connaissances. Il devient alors plus facile de lier deux ontologies modélisées suivant les même patrons de conceptions [Gangemi and Presutti, 2009]. Il existe un entrepôt de patrons de conceptions pouvant être réutilisés sur le site Ontology Design Pattern [56]. Ce critère est d'autant plus intéressant que la FAO [57] a proposé des patrons de conception pour le domaine de l'agriculture.

Enfin, notre dernier critère est la possibilité de créer des ontologies modulaires. Pour faciliter la création d'une ontologie de grande taille [Rector, 2003, d'Aquin et al., 2006], elle est découpée en plusieurs modules, chacun étudiant un aspect spécifique de cette ontologie [Grau et al., 2007]. Tous les modules sont ensuite reliés pour créer l'ontologie

56. http://ontologydesignpatterns.org/
57. Food and Agriculture Organization of the United Nations - http://www.fao.org

finale. Cette modularisation des ontologies est assimilable à la décomposition en différents paquetages d'un système d'information en ingénierie logicielle.

Bien que plusieurs méthodologies aient été définies dans la littérature, à notre connaissance une seule permet de répondre à nos trois critères : la méthodologie NeOn. Cette méthodologie se décline en neuf scénarios, chacun proposant une méthode différente de construction d'ontologies suivant les objectifs souhaités. Chaque scénario est constitué de différents processus pour la construction collaborative d'ontologies et de réseaux d'ontologies. La figure 14 présente ces scénarios. [Suárez-Figueroa, 2010]

1. **«Des spécifications à l'implémentations» :** Le premier scénario permet la construction d'une ontologie ex nihilo, c'est à dire qu'aucune autre source n'est utilisée dans cette méthode.
2. **«Réutilisation de sources non ontologiques» :** Ce scénario permet le développement d'une ontologie à partir de sources non-ontologiques telles que des thésaurus, des bases de données ou autres.
3. **"Réutilisation de sources ontologiques" :** Ce scénario propose une méthode pour la construction d'une ontologie en réutilisant d'autres ontologies existantes, ou par réutilisation de parties d'ontologies existantes.
4. **"Réutilisation et reingénierie de sources ontologiques" :** Ce scénario est similaire au précédent, si ce n'est que les ontologies considérées dans cette méthodologie peuvent être modifiées si nécessaire.
5. **"Réutilisation et fusion de sources ontologiques" :** Ce scénario permet la réutilisation de sources ontologiques et leur fusion pour créer l'ontologie finale.
6. **"Réutilisation, réingénierie et fusion de sources non ontologiques" :** Ce scénario est similaire au précédent en incluant de possibles modifications aux ontologies sélectionnées avant la fusion.
7. **"Réutilisation de patrons de conceptions ontologiques (ODP)" :** Ce scénario propose de sélectionner des patrons de conception ontologiques qui permettront la construction d'une ontologie.
8. **"Restructuration de sources ontologiques" :** Ce scénario modifie les ontologies pour les intégrer dans un réseau d'ontologies.
9. **"Localiser des sources ontologiques" :** Ce scénario traduit une ontologie dans une langue donnée.

Grâce à la diversité de ces scénarios, la méthodologie NeOn permet de couvrir un grand nombre de situations. Ces scénarios ne sont pas incompatibles et un certain nombre de processus sont communs à tous les scénarios. Nous pouvons le remarquer sur la figure 14, car tous les scénarios impliquent l'application du scénario 1. Ce scénario est donc central dans cette méthodologie et dresse les étapes principales de la création d'une base de connaissances, quel que soit le scénario choisi. Ce scénario définit 5 étapes :

1. **Spécifications des besoins :** Durant cette étape, l'équipe en charge du développement d'une base de connaissances doit définir les besoins d'une telle base de connaissances comme le domaine, la langue ou encore les objectifs que la base de

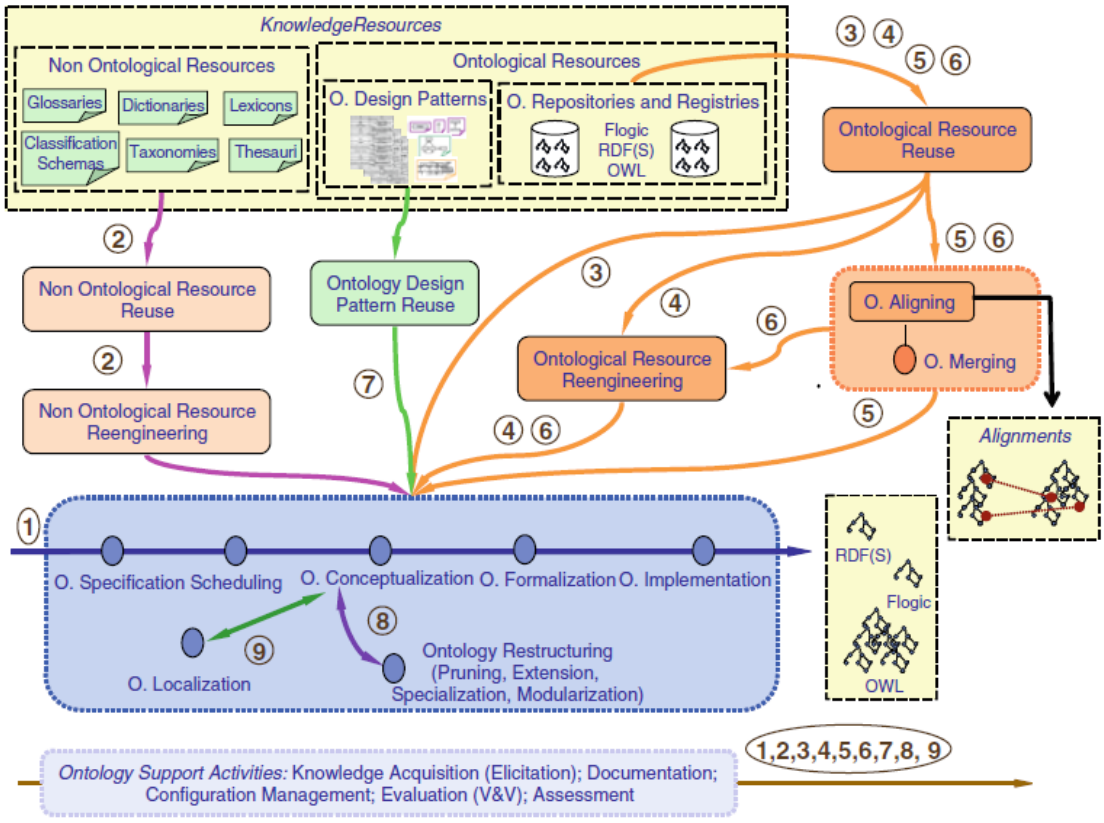


FIGURE 14 – Méthodologie NeOn

connaissances doit respecter. Ces objectifs sont représentés sous forme de questions auxquelles la base de connaissances doit pouvoir répondre.

2. **Plannification :** Cette étape de planification permet de définir un calendrier des objectifs à atteindre et les ressources humaines qui seront nécessaires pour ces objectifs.
3. **Modélisation :** L'étape de la modélisation de la base de connaissances permet la structuration et l'organisation des différentes connaissances à représenter.
4. **Formalisation :** Cette étape permet la formalisation des connaissances en suivant un vocabulaire prédéfini.
5. **Implémentation :** Cette dernière étape permet la représentation des connaissances sous un format d'implémentation manipulable par une machine (par exemple en OWL).

Le résultat de ce scénario est une base de connaissances implémentée. Les autres scénarios permettent d'obtenir des éléments pouvant aider à l'application du scénario 1. Le scénario 3 permet par exemple de trouver d'autres ontologies pouvant aider à la modélisation de notre base de connaissances.

Cette méthodologie prenant en compte tous les critères énumérés ci-dessus, nous avons décidé de l'utiliser comme structure pour la proposition que nous présentons dans ce manuscrit.

## 4. Conclusion

La première partie présente notre vision du Web sémantique et plus particulièrement du Web de données liées. Nous avons mis en évidence deux verrous du Web sémantique à l'heure actuelle. Le premier est l'apparition de plus en plus de données sur le Web de données liées sans considération de la confiance portée à ces données. Il devient alors difficile de réutiliser ces données sans pouvoir garantir leur qualité. Le deuxième verrou provient du manque de réutilisation de sources déjà existantes pour fournir le Web de données liées. En effet, un grand nombre de sources sont disponibles mais ne participent pas du Web de données liées. Il est donc particulièrement intéressant de pouvoir exploiter la richesse de ces sources.

C'est notamment le cas dans le domaine de l'agriculture comme nous l'avons vu dans la deuxième section. Un grand nombre de sources existent pour ce domaine mais peu sont présentes sur le Web de données liées alors que cela pourrait permettre de faciliter la transition vers une agriculture durable.

Enfin, nous avons présenté la méthodologie NeOn qui propose différents scénarios de création d'ontologies. Ces scénarios permettent la prise en compte des différents critères que nous avons définis pour répondre à nos besoins. Nous considérerons cette méthodologie pour structurer la proposition présentée dans ce manuscrit.

Il est intéressant de pouvoir proposer une méthodologie qui permette d'exploiter plusieurs sources simultanément. De cette manière, il devient possible d'exploiter la complémentarité des sources. Cette exploitation de la complémentarité des sources

entraîne la nécessité de définir un niveau de confiance. Comme nous venons de le voir, une ressource provenant d'une source de bonne qualité a une confiance supérieure à une autre ressource provenant d'une source de moins bonne qualité. Nous posons alors l'hypothèse suivante :

> La confiance d'une ressource est fonction de deux criteres : (1) le nombre de sources dont elle est issue et (2) la qualité de ces sources.

Notre proposition présentée dans ce manuscrit sera construite en fonction de cette hypothèse.

# Deuxième partie .
# Etat de l'art

Nous souhaitons réutiliser les différents types de sources présentés précédemment pour obtenir une base de connaissances. L'originalité de notre approche réside dans la transformation simultanée des sources. Nous étudierons les différents travaux concernant la transformation de sources non-ontologiques. Cette étude nous permettra de déterminer s'il existe des méthodes de transformation considérant les sources de manière simultanée et quelles sont les étapes particulièrement complexes d'une transformation. Nous verrons ensuite les techniques d'alignement pour découvrir les éléments en commun entre les sources dans le but de les fusionner. Ces correspondances peuvent être exploitées de différentes manières pour obtenir une fusion de bases de connaissances. Nous étudierons ici les différentes façon de fusionner et particulièrement la façon dont sont traitées les incompatibilités lors de cette fusion. Lorsque nous considérons la fusion d'éléments, il est important de tenir compte de la qualité des sources, afin de représenter un intérêt variable suivant la provenance des éléments fusionnés. Nous finirons alors notre étude sur les travaux concernant la définition de la qualité d'une source.

## 1. Transformation de sources non-ontologiques

Notre objectif est de pouvoir extraire des connaissances provenant de plusieurs sources. Pour cela, il est nécessaire de pouvoir transformer des sources non-ontologiques en bases de connaissances. Nous avons vu précédemment que ces sources pouvaient être de natures diverses et reposer sur des niveaux de formalisme différents. Nous allons ici nous intéresser à deux types de sources en particulier, qui sont les thésaurus et les bases de données.

### 1.1. Transformation de Thésaurus

L'étude de la transformation des thésaurus est primordiale dans nos travaux puisque, dans le domaine que nous étudions, ce type de sources est très répandu. La modélisation utilisée dans un thésaurus est particulièrement proche du formalisme utilisé pour définir une base de connaissances. Néanmoins, le niveau de formalisation des thésaurus est inférieur à celui des bases de connaissances. La transformation vers un formalisme plus élevé impose de lever des ambiguïtés. C'est sur ce point que repose toute la difficulté de la transformation de thésaurus en bases de connaissances.

#### 1.1.1. Présentation des travaux

Nous allons nous attacher, dans la suite, à mettre en évidence un certain nombre de critères permettant de comparer les différents travaux étudiés concernant la transformation de thésaurus en bases de connaissances :

| | Usa. | Domaine | Thés. | Obj. |
|---|---|---|---|---|
| (Wielinga et al., 2001) | RI | Art et architecture | AAT | CO |
| (Hahn, 2003) | ID | Médecine | UMLS | CO + PO |
| (Van Assem et al., 2004) | RI | Médecine, générique | MESH (en), WordNet (en) | CO + PO |
| (Soergel et al., 2004) | RI | Agriculture | AGROVOC (multi) | CO + PO |
| (Hepp et De Bruijn, 2007) | RI | Produits et services | UNPCS (en), eCl@ss (multi) | CO |
| (Chrisment et al., 2008) | RI | Astronomie | IAU (multi) | CO |
| (Villazón-Terrazas et al., 2010) | N | Générique | ASFA (multi)/ ETT (multi) | CO + PO |
| (Li et Li, 2012) | ID | Agriculture | MESH | CO + PO |
| (Kless et al., 2012) | ID | Agriculture | AGROVOC | CO |
| (Charlet et al., 2012) | RI | Urgences médicales | SNOMED (fr), UMLS (multi) | EO + PO |

TABLE 8 – Présentations des travaux

— *Usage (Usa.)* présente le but de l'ontologie générée, parmi trois possibilités : Recherche d'Information (RI), Intégration de Données (ID), Non défini (N) ;
— *Domaine* indique le domaine dans lequel est appliquée la méthode ;
— *Thésaurus utilisé(s) (Thés.)* précise le ou les thésaurus sur lesquels la méthode a été appliquée ;
— *Objectif (Obj.)* détermine l'objectif global de la méthode parmi :
  — Construction d'Ontologies (CO) : création de classes et de propriétés à partir de rien,
  — Enrichissement d'Ontologies (EO) : ajout de classes et de propriétés à une ontologie existante,
  — Peuplement d'Ontologies (PO) : création d'individus et de relations entre individus en reprenant une ontologie existante.

Nous pouvons tout d'abord observer dans le Tableau 8 que la plupart des méthodes étudiées [Charlet et al., 2012, Chrisment et al., 2008, Hepp and De Bruijn, 2007, Soergel et al., 2004, Van Assem et al., 2004, Wielinga et al., 2001] transforment les thésaurus en vue d'une utilisation dans un système de recherche d'information. Seuls [Hahn, 2003, Kless et al., 2012, Li and Li, 2012] le font dans un but d'intégration de données. Cette observation résulte du fait que les thésaurus sont souvent utilisés pour

la recherche d'information. Par conséquent, les projets de transformation de thésaurus ont pour but d'intégrer de la sémantique dans les systèmes de recherche d'information. Les domaines sur lesquels sont appliquées les méthodes sont divers, ce qui montre l'intérêt d'outils de transformation aussi génériques que possibles. La plupart des méthodes étudiées ont été appliquées sur un seul thésaurus, voire sur une partie du thésaurus pour les méthodes fastidieuses. Certaines méthodes ont été appliquées sur plusieurs thésaurus, mais seul [Charlet et al., 2012] utilise plusieurs sources pour enrichir une seule et même ontologie. Ces travaux considèrent plusieurs types de sources non-ontologiques lors de la création de l'ontologie en plus d'un thésaurus. Néanmoins, chaque source est transformée avec un processus manuel très spécifique. La dernière colonne montre que [Charlet et al., 2012] est la seule méthode d'enrichissement, alors que les autres sont des méthodes de création d'ontologies. Nous pouvons remarquer que seules certaines méthodes vont jusqu'au peuplement de l'ontologie générée afin d'obtenir une base de connaissances. [Charlet et al., 2012, Hahn, 2003, Li and Li, 2012, Soergel et al., 2004, Van Assem et al., 2004, Villazon-Terrazas et al., 2010].

### 1.1.2. Utilisation de la terminologie

L'étape de transformation de la terminologie consiste à étudier les termes et les concepts d'un thésaurus afin de générer les classes dans une ontologie. Pour cela, deux familles de méthodes se démarquent dans les travaux précédemment cités.

— La plupart des travaux transforment uniquement les concepts du thésaurus en classes OWL.
— Certains travaux [Hepp and De Bruijn, 2007, Villazon-Terrazas et al., 2010] génèrent une classe de l'ontologie pour chaque terme du thésaurus.

Les travaux générant une classe par terme utilisent ensuite les relations de synonymie pour définir des relations "sameAs"(Cf. section 1.3 du chapitre I) entre les classes créées. De cette manière, il n'existe pas d'ambiguïté. Cela surcharge néanmoins considérablement le nombre de classes de l'ontologie. La génération des classes est le plus souvent validée manuellement par un expert du domaine.

Concernant nos travaux, la transformation des concepts semble être l'approche la plus intéressante. Une validation manuelle semble pourtant peu réaliste, vu la taille importante de certains thésaurus (Cf. section 2.4.2 du chapitre I). Il est donc interessant de considérer la redondance des éléments transformés à partir de plusieurs sources. De cette façon, la phase de validation manuelle peut être simplifiée.

### 1.1.3. Utilisation de la hiérarchie

Une fois les classes de l'ontologie extraites, les relations hiérarchiques du thésaurus peuvent également être utilisées pour enrichir la transformation. Les relations "narrower" et "broader" témoignent d'une spécialisation ou d'une généralisation thématique et non d'une relation de subsomption entre classes. Cette distinction induit certaines confusions puisque, comme le montre [Soergel et al., 2004], les relations hiérarchiques d'un thésaurus sont parfois utilisées pour représenter des relations de composition (on

| | Sél. | Trait. | Types | Désambiguïsation | Val. |
|---|---|---|---|---|---|
| (Wielinga et al., 2001) | non | Auto. | HsCO | non | non |
| (Hahn, 2003) | oui | Auto. | HsCO - HC - HF | non | Auto. |
| (Van Assem et al., 2004) | non | Auto. | HsCO - HC - HF | non | non |
| (Soergel et al., 2004) | non | Auto. | HsCO - HC - HF | Patrons de transformations | non |
| (Hepp et De Bruijn, 2007) | non | Auto. | HsCO | non | non |
| (Chrisment et al., 2008) | non | Auto. | HsCO | non | Man. |
| (Villazón-Terrazas et al., 2010) | non | Auto. | HsCO - HC - HF | Source externe (non définie) | non |
| (Li et Li, 2012) | non | Auto. | HsCO - HC - HF | non | non |
| (Kless et al., 2012) | oui | Man. | HsCO - HC - HF | Man. | Man. |
| (Charlet et al., 2012) | oui | Man. | HsCO - HC - HF | Man. | Auto. |

TABLE 9 – Utilisation de la hiérarchie

trouve par exemple dans AGROVOC "Cow, narrower, Cow Milk" [58]). La transformation d'une relation hiérarchique en relation "subClassOf" ne peut donc pas être systématique.

Pour étudier le traitement de ces relations hiérarchiques, nous proposons différents critères. Le résultat de notre analyse est présenté dans le tableau 2. Les critères liés au traitement de la hiérarchie du thésaurus sont :

— *Sélection (Sél.)* indique si la méthode filtre les branches de la hiérarchie ;
— *Traitement (Trai.)* précise le niveau d'automatisation du traitement (automatique ou manuel) ;
— *Types* énumère les types de relations possibles dans l'ontologie (Hiérarchie "subClassOf" -> HsCO, Hiérarchie Compositionnelle -> HC, Hiérarchie Fonctionnelle -> HF) ;
— *Désambiguïsation* précise si la méthode applique un processus de désambiguïsation pour déterminer la nature de la relation dans l'ontologie et, si oui, quelle technique est utilisée ;
— *Validation (Val.)* indique si la méthode applique un processus de validation du résultat et, si oui, s'il s'agit d'une validation automatique ou manuelle.

Le filtrage des branches de la hiérarchie peut permettre le traitement spécifique de la hiérarchie du thésaurus suivant une branche spécifique du thésaurus. Un thésaurus pouvant couvrir un domaine large, cette hiérarchie peut donc être utilisée pour représenter différentes relations suivant les branches. La hiérarchie fonctionnelle représente les relations de généralisation et de spécialisation thématiques spécifiques au domaine.

Il apparaît dans le tableau 9 que seulement [Charlet et al., 2012][Hahn, 2003] [Kless et al., 2012] effectuent une sélection de la partie du thésaurus à traiter. Cet aspect est particulièrement intéressant puisqu'il permet de ne considérer qu'une sous-partie d'un thésaurus, l'intégralité d'un thésaurus n'étant pas forcément pertinente pour les besoins de la transformation. Les méthodes de traitement de la hiérarchie se décomposent en deux grandes familles :

— celles qui transforment toutes les relations hiérarchiques narrower/broader en un

58. Vache, plus spécifique, Lait de Vache

seul type de relation ontologique ;
— celles qui transforment les relations hiérarchiques en plusieurs types de relations ontologiques.

Dans la première famille, nous trouvons les travaux de [Hepp and De Bruijn, 2007, Chrisment et al., 2008, Wielinga et al., 2001, Van Assem et al., 2004]. Ils proposent un traitement automatique de la hiérarchie en la transformant directement en hiérarchie de subsomption ("subClassOf"). [Chrisment et al., 2008] précise que les relations narrower/broader du thésaurus considéré correspondent particulièrement à des relations "subClassOf". [Van Assem et al., 2004] propose de choisir la transformation de toute la hiérarchie et de déterminer si c'est une hiérarchie de subsomption, de méronymie ("partOf") ou autre (fonctionnelle). Mais toutes les relations de la hiérarchie du thésaurus seront traduites de la même façon. Les auteurs ne proposent pas de pouvoir changer de transformation suivant la branche du thésaurus car ils considèrent que la modélisation d'un thésaurus est similaire dans toutes les branches (toutes les relations hiérarchiques d'un thésaurus ont le même sens). Dans la deuxième famille nous trouvons les travaux de [Soergel et al., 2004, Villazon-Terrazas et al., 2010, Kless et al., 2012, Li and Li, 2012, Charlet et al., 2012]. Ces travaux proposent un traitement automatique de la hiérarchie tout en prenant en compte les éventuelles ambiguïtés qu'elle peut receler. [Soergel et al., 2004] propose une utilisation de règles de désambiguïsation qui permettent la désambiguïsation des relations hiérarchiques du thésaurus, mais aucune autre source n'est utilisée. [Villazon-Terrazas et al., 2010] propose la mise en place de patrons de transformation pour spécifier comment la transformation va être effectuée. Dans certains patrons définis dans [Villazon-Terrazas et al., 2010], il est proposé de désambiguïser les relations hiérarchiques par l'utilisation d'une source externe sans préciser laquelle ni de quelle façon elle est utilisée. Nous pouvons remarquer que les patrons de transformation proposés dans [Villazon-Terrazas et al., 2010] sont une généralisation des règles de désambiguïsation proposées dans [Soergel et al., 2004]. En effet, les règles de désambiguïsation peuvent être incluses dans un patron de transformation. Mais quand les règles de désambiguïsation ne permettent de désambiguïser que des relations, les patrons de transformation permettent la mise en place de tout un processus de transformation. Les travaux les plus récents [Charlet et al., 2012, Kless et al., 2012, Li and Li, 2012] transforment manuellement la hiérarchie des thésaurus. Il est intéressant de noter que [Hahn, 2003, Kless et al., 2012] sont les seuls auteurs qui utilisent des thésaurus dans lesquels sont définis plusieurs types de relations hiérarchiques (de subsomption, de méronymie ou autres). Les relations hiérarchiques dans ces thésaurus sont déjà désambiguïsées, ce qui facilite leur transformation. Par rapport à la validation de la hiérarchie de l'ontologie, nous pouvons remarquer que [Charlet et al., 2012] applique toujours une validation par patrons structurels et [Chrisment et al., 2008] une validation manuelle. [Hahn, 2003] utilise un moteur d'inférences pour détecter les incohérences à corriger. Tous les autres auteurs ne mentionnent pas de validation particulière.

Nous pouvons déduire de cette étude que l'utilisation des relations hiérarchiques reste une problématique majeure. En effet, la plupart des travaux utilisent une transformation simple (la hiérarchie transformée directement en "subClassOf"). Les seuls travaux mettant

| | Util. | Trait. | Désamb. | Val. |
|---|---|---|---|---|
| (Wielinga et al., 2001) | non | non | non | non |
| (Hahn, 2003) | oui | Auto. | Man. | non |
| (Van Assem et al., 2004) | non | non | non | non |
| (Soergel et al., 2004) | oui | Auto. | Patrons de transformations | non |
| (Hepp et De Bruijn, 2007) | non | non | non | non |
| (Chrisment et al., 2008) | oui | Semi-auto. | Corpus | Man. |
| (Villazón-Terrazas et al., 2010) | oui | Auto. | Source externe | non |
| (Li et Li, 2012) | oui | Man. | Man. | non |
| (Kless et al., 2012) | oui | Man. | Man. | non |
| (Charlet et al., 2012) | oui | Man. | Man. | Auto. |

TABLE 10 – Utilisation des associations

en place une méthode complexe sont ceux présentés dans [Soergel et al., 2004] qui proposent de définir des règles de désambiguïsation ou [Villazon-Terrazas et al., 2010] qui propose une désambiguïsation par l'utilisation de patrons de transformation et encourage l'utilisation d'une source externe. Pour nos travaux, une transformation directe n'est pas envisageable puisque nous souhaitons une représentation correcte du domaine. L'utilisation de patrons de transformation peut permettre de pallier ce problème. Néanmoins, un patron générique pour tout un thésaurus semble complexe à obtenir. De plus, une sous-partie du thésaurus peut être suffisante pour répondre aux besoins de la base de connaissances souhaitée. Il serait alors intéressant de pouvoir filtrer les éléments à transformer. Cette méthode peut pourtant introduire un certain nombre d'erreurs en raison de l'automatisation du processus. Il faut donc envisager une validation des désambiguïsations effectuées.

### 1.1.4. Utilisation des associations

Considérons maintenant le traitement des relations d'associations (related term) présentes dans un thésaurus. Ces relations permettent de définir une relation entre deux concepts du thésaurus sans en définir la nature exacte. Pour étudier leur traitement, nous allons, là-encore, définir plusieurs critères qui seront ensuite utilisés dans le tableau 3 :

— *Utilisation (Util.)* précise si la méthode prend en compte les relations d'associations (relations "related" dans les thésaurus) ;
— *Traitement (Trait.)* détermine le niveau d'automatisation du traitement des relations d'association (aucun traitement, manuel, semi-automatique, automatique) ;
— *Désambiguïsation (Désamb.)* détermine si ces relations sont désambiguïsées lors de la transformation, et précise la technique utilisée ;
— *Validation (Val.)* précise si la méthode applique un processus de validation (manuel ou automatique) concernant ces relations.

Les relations d'associations sont les plus difficiles à transformer car leur sens n'est pas précisé. Elles peuvent en effet représenter tout type de relations ontologiques.

C'est pour cela que, comme nous l'observons dans le tableau 10, certains travaux [Hepp and De Bruijn, 2007, Van Assem et al., 2004, Wielinga et al., 2001] ne prennent pas en compte ces relations du thésaurus. Les travaux présentés dans [Charlet et al., 2012, Kless et al., 2012, Li and Li, 2012] traitent ces relations de manière entièrement manuelle, ce qui permet par la même occasion de les désambiguïser. La méthode proposée par [Hahn, 2003] extrait automatiquement les relations ambiguës et un expert les désambiguïse manuellement. Nous retrouvons [Chrisment et al., 2008, Soergel et al., 2004, Villazon-Terrazas et al., 2010] qui traitent ces relations de manière automatique. Les travaux présentés dans [Soergel et al., 2004] utilisent le même procédé que pour les relations hiérarchiques, c'est-à-dire une utilisation de règles de désambiguïsation permettant d'identifier des relations entre les éléments. [Villazon-Terrazas et al., 2010] utilise, lui aussi, le même procédé que pour les relations hiérarchiques, autrement dit l'utilisation d'un patron de transformation et potentiellement une source externe pour désambiguïser cette relation (sans préciser la méthode exacte). La méthode qui effectue un traitement intéressant de ces relations est celle présentée par [Chrisment et al., 2008]. Elle extrait toutes les relations d'association du thésaurus pour poser des hypothèses de relations candidates entre les classes de l'ontologie. Ces relations sont ensuite désambiguïsées par un traitement automatique qui exploite un corpus de textes du domaine. Si plusieurs relations existent, le choix est laissé à l'utilisateur.

Ignorer les relations d'associations lors d'un processus de transformation d'un thésaurus en ontologie ne permettrait d'avoir qu'une hiérarchie de classe. Or il est particulièrement intéressant de pouvoir exporter les liens entre ces classes autre que hiérarchiques. Malheureusement, leur extraction reste complexe. Le traitement d'un corpus de textes, comme le préconise [Chrisment et al., 2008], est une méthode très intéressante pour permettre la désambiguïsation de relations en fonction des occurrences de ces relations dans le texte. Néanmoins, il nous faudrait disposer d'un corpus suffisamment important pour qu'il nous permette d'identifier tous les noms de relations entre classes de l'ontologie. Nous avons souhaité, dans ces travaux, nous concentrer sur le traitement de sources structurées. Nous souhaitons pouvoir transformer des thésaurus de grandes tailles en minimisant l'intervention humaine. En cela, l'utilisation de patrons de transformation spécifiques suivie d'une désambiguïsation par la comparaison des sources est la solution la plus envisageable. Elle permet de n'avoir à spécifier qu'un patron de transformation pour effectuer la transformation. Ce patron spécifique contient les règles de désambiguïsation des relations dans le thésaurus. Nous retrouvons aussi le problème de la validation des éléments extraits, car la désambiguïsation à l'aide de sources externes n'est pas totalement fiable, ces sources pouvant contenir des erreurs.

### 1.1.5. Analyse

**Terminologie :** Plusieurs aspects ressortent de l'étude de ces méthodes de transformation de thésaurus en ontologies. Tout d'abord, concernant l'identification des classes de l'ontologie à partir des termes du thésaurus, nous avons observé qu'il n'y a que très peu de validation des classes générées. Les seules validations qui existent sont manuelles [Chrisment et al., 2008, Charlet et al., 2012]. Les autres méthodes étudiées considèrent

donc les classes générées comme valides. Le manque de validation peut causer certains problèmes puisque, comme montré dans [Soergel et al., 2004], certaines relations de synonymie ne sont pas correctes et peuvent apporter des confusions lors du traitement automatique de thésaurus. Dans Agrovoc, par exemple, nous trouvons que "Hydrophilicity" et "Hydrophobicity" sont des labels du même concept.

**Relations hiérarchiques :** Le traitement de la hiérarchie du thésaurus est aussi, dans la plupart des méthodes, effectué de façon relativement simple. Nous avons vu que les auteurs interprètent généralement la hiérarchie du thésaurus comme une hiérarchie "subClassOf", ce qui n'est pas toujours pertinent [Soergel et al., 2004]. Les auteurs prenant en compte cette problématique sont [Soergel et al., 2004, Villazon-Terrazas et al., 2010]. [Soergel et al., 2004] propose une mise en place de règles de désambiguïsation permettant l'identification de la nature d'une relation du thésaurus, quelle qu'elle soit, relations hiérarchiques comprises. Néanmoins, ce système nécessite d'identifier spécifiquement dans ces règles toutes les relations du thésaurus qui représentent une relation de "subClassOf" dans l'ontologie. Ceci peut devenir contraignant dans le cas du traitement d'un gros thésaurus. [Villazon-Terrazas et al., 2010], en plus de l'utilisation de patrons de transformation, propose une désambiguïsation de ces relations en utilisant une source externe. En effet, il existe des sources telles que WordNet ou DBpedia qui définissent plusieurs relations hiérarchiques ("subClassOf" , compositionnelle). En utilisant ces sources, nous pouvons désambiguïser la relation hiérarchique du thésaurus. Néanmoins, ces sources n'étant pas totalement fiables, il est peu judicieux de s'appuyer entièrement sur leur contenu. [Gangemi et al., 2003] montre d'ailleurs que WordNet ne contient pas une modélisation qui peut être reprise totalement pour en générer une ontologie.

**Relations d'association :** Concernant le traitement des relations d'association, trois possibilités nous semblent envisageables.

— L'utilisation de règles de désambiguïsation comme le présente [Soergel et al., 2004], qui nécessite une définition de toutes les relations existantes et demande donc un travail important avant la transformation.

— La proposition d'hypothèses d'identification des relations candidates à partir de l'analyse d'un corpus de textes, comme le propose [Chrisment et al., 2008]. Une limite dans cette approche est que le traitement du corpus peut lui aussi générer des erreurs, ce qui peut amener à de mauvaises désambiguisations dans l'ontologie finale.

— La désambiguïsation des relations par sources externes, proposée dans les travaux [Villazon-Terrazas et al., 2010], est aussi possible, bien que les sources externes ne définissent généralement pas de relations spécifiques de domaine et puissent également entraîner certaines erreurs d'interprétation.

L'ensemble de ces travaux montre qu'il est nécessaire de désambiguïser toutes les relations des thésaurus. Nous pouvons remarquer que les travaux les plus récents [Charlet et al., 2012, Kless et al., 2012, Li and Li, 2012] s'orientent plutôt vers une transformation manuelle du thésaurus.

## 1.2. Transformation de bases de données

Après avoir étudié les méthodes de transformation de thésaurus, nous allons maintenant nous intéresser aux méthodes de transformation des bases de données. Notre volonté étant de réutiliser des sources non-ontologiques existant sur le Web, il nous est apparu important de considérer les bases de données au vu de leur place centrale dans les systèmes d'informations actuels. Cette section décrit et analyse les méthodes permettant la transformation d'une base de données (structure et contenu) en une base de connaissances.

L'étude de la littérature sur ce domaine a fait apparaître deux familles de techniques :

— Transformation par règles

— Transformation par un langage intermédiaire

### 1.2.1. Transformation par règles

Cette famille de méthodes propose de définir des règles permettant d'identifier les actions pour la transformation. Nous pouvons par exemple définir que toutes les tables deviennent des classes dans l'ontologie finale. Les travaux [Sequeda et al., 2011b, Astrova, 2004, Tirmizi et al., 2008, Cerbah, 2008, Arenas et al., 2012] appliquent ce principe de règles de transformation. D'après l'étude [Sequeda et al., 2012], les travaux les plus aboutis sont ceux présentés dans [Tirmizi et al., 2008] grâce à la complétude des règles définies. Cette complétude est étudiée en fonction des caractéristiques des bases de données prises en compte par les règles. Il a été montré dans [Sequeda et al., 2011b] que les travaux présentés dans [Tirmizi et al., 2008] respectent cette complétude, contrairement aux autres travaux de l'état de l'art.

Nous pouvons remarquer que cette famille de méthodes est similaire à la définition de patrons de transformation présentés dans [Villazon-Terrazas et al., 2010].

**Règles de transformations [Tirmizi et al., 2008]** Les travaux de [Tirmizi et al., 2008] définissent six règles pour transformer une base de données relationnelle. Ces six règles sont définies de la façon suivante :

**Régle 1 :** détecte une table binaire qui servira à la mise en place des règles suivantes. Ce type de table spécifie une relation entre deux autres tables. Elle ne contient donc que deux attributs (ou ensembles d'attributs) qui sont des clefs étrangères vers deux autres tables. Elle ne contient aucune autre clef étrangère ou autre attribut. Elle permet donc de spécifier les relations [0..n] entre deux tables.

**Règle 2 :** pose la condition selon laquelle si une table n'est pas une table binaire, alors une classe OWL est créée.

**Règle 3 :** définit la génération d'une propriété d'objet entre deux classes s'il existe une table binaire définissant une relation entre les deux tables qui ont servi à la création des deux classes.

**Règle 4 :** est composée de plusieurs sous-règles pour définir les propriétés (Object-Property) provenant de clefs étrangères mais sans tables binaires :

**Règle 4.a :** conduit à la définition d'une propriété fonctionnelle s'il existe une clef étrangère vers une table non binaire sans la contrainte NOT NULL, ni UNIQUE.

**Règle 4.b :** est la même que la *a*, seulement avec la contrainte NOT NULL sur la clef étrangère. Cette contrainte ajoute le fait que la cardinalité de la propriété est forcément égale à 1. Chaque instance de cette classe aura nécessairement cette relation.

**Règle 4.c :** est la même que la *a*, avec la contrainte UNIQUE. Cette contrainte ajoute le fait que la propriété inverse est aussi fonctionnelle.

**Règle 4.d :** est la même que la *a*, avec les contraintes NOT NULL et UNIQUE. Les cardinalités de la propriété générée et de son inverse sont égales à 1.

**Règle 5 :** définit les conditions pour la génération de propriétés de type de données avec plusieurs sous-règles :

**Règle 5.a :** indique que tout attribut qui n'est pas une clef étrangère sera traduit par une propriété de données. Cette propriété de données est fonctionnelle en raison de la structure des attributs dans les bases de données.

**Règle 5.b :** est la même que la *a* avec la condition NOT NULL. Celle-ci ajoute la cardinalité 1 à la propriété de données.

**Règle 5.c :** est la même que la *a* avec la condition CHECK. Cette condition ajoute le fait que la valeur de la propriété de données doit être dans la liste définie en base de données.

**Règle 6 :** définit les conditions nécessaires pour la génération d'une relation de type "subClassOf" (spécialisation) entre deux classes de l'ontologie. Il faut pour cela que deux tables soient reliées par une clef étrangère qui est aussi considérée comme clef primaire de la table source.

Les travaux présentés dans [Tirmizi et al., 2008] présentent quatre propriétés qui doivent être respectées pour que les règles définies permettent une transformation considérée comme valide :

**Conservation des informations :** Possibilité de transformer la base de connaissances générées pour retrouver la base de données initiale.

**Conservation des requêtes :** Toutes les requêtes possibles sur la base de données le sont aussi sur la base de connaissances.

**Monotonie :** Une modification sur la base de données peut être incluse dans la base de connaissances finale sans refaire tout le processus de transformation.

**Conservation sémantique :** Si la base de données est inconsistante, alors la base de connaissances le sera aussi.

Ces règles ont été étendues [Sequeda et al., 2011a] et proposées dans une recommandation du W3C [59] [Arenas et al., 2012].

59. http://www.w3.org/TR/rdb-direct-mapping/

### 1.2.2. Transformation par un langage intermédiaire

La deuxième famille de méthodes de transformation est la transformation par l'utilisation d'un langage intermédiaire. Pour cela, nous pouvons citer deux travaux représentatifs [Auer et al., 2009, Das et al., 2012].

Les méthodes de cette famille ne sont pas directement des méthodes de transformation mais plutôt une proposition d'implémentation des transformations. Néanmoins, certains travaux cherchent à transformer des bases de données en exploitant directement un langage intermédiaire défini. C'est notamment le cas de [Auer et al., 2009] qui a proposé des transformations de bases de données d'applications très utilisées sur le Web [60].

Le projet Triplify [Auer et al., 2009] propose d'étendre le langage SQL en ajoutant des informations concernant la transformation à effectuer. Nous pouvons prendre l'exemple suivant :

```
SELECT id,
price AS 'price^^xsd:decimal',
desc AS 'rdfs:label@en',
cat AS 'belongsToCategory->category'
FROM products
```

Cette requête permet de spécifier que l'attribut "price" de la table "products" est une propriété de données de type décimal, l'attribut "desc" est un label en anglais et l'attribut "cat" est une propriété nommée "belongsToCategory" vers une instance de "category".

Le projet R2RML [Das et al., 2012] est également une recommandation du W3C [61] sur l'utilisation de JSON comme langage intermédiaire pour définir la transformation à effectuer sur la base de données. Nous pouvons prendre l'exemple suivant :

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

Cet exemple prend en compte une table "EMP" qui permet la création de la classe "Employee". Cette table a l'attribut "EMPNO" qui est utilisé pour la génération de

60. Par exemple PhpBB - http://triplify.org/About
61. http://www.w3.org/TR/2012/REC-r2rml-20120927/

l'URI d'une instance. Elle contient aussi l'attribut "ENAME" qui est transformé en une propriété de données de "Employee" nommée "ex :name". Un exemple de résultat est :

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

### 1.2.3. Analyse

En plus de ces deux familles, il existe d'autres travaux qui présentent des approches dédiées à une base de données spécifique [Dhombres et al., 2012, Krivine et al., 2009]. Ces travaux montrent que les méthodes génériques ne sont pas forcément pertinentes dans un cas réel. Les méthodes présentées précédemment montrent toutes des limites dans le cas des bases de données complexes. C'est particulièrement le cas pour la gestion des relations de spécialisation entre classes (subClassOf) qui n'est pas suffisamment explicite dans les bases de données. Les approches à base de règles peuvent découvrir certaines de ces relations mais peuvent aussi apporter du bruit, alors que les méthodes utilisant un langage intermédiaire permettent de mettre en place la transformation directement au niveau des éléments de la base de données.

## 1.3. Analyse des méthodes de transformation

Cette étude des méthodes de transformation de thésaurus et de bases de données nous permet de déduire qu'il n'existe, ni pour les thésaurus, ni pour les bases de données, de méthodes génériques permettant une transformation automatique de bonne qualité. Les méthodes proposant une qualité suffisante nécessitent une intervention humaine, soit au départ pour définir la transformation, soit pour valider et enrichir le résultat obtenu. Il apparaît que lorsque nous souhaitons transformer une source non-ontologique, même autre qu'un thésaurus ou une base de données, il est nécessaire de définir les règles à utiliser pour transformer la source. Les travaux qui sont les plus adaptés et génériques pour prendre en compte tous les types de sources sont ceux présentés dans NOR2O [Villazon-Terrazas et al., 2010]. Ceux-ci permettent la définition d'un patron de transformation spécifique à la source en utilisant des patrons génériques déjà définis. De cette manière, il est possible d'utiliser des transformations génériques déjà définies (mais qui peuvent apporter du bruit) ou de spécialiser un patron de manière spécifique à la source étudiée. La faiblesse de cette méthode étant la désambiguïsation des relations, nous pouvons utiliser les travaux de [Soergel et al., 2004]. Ces travaux proposent des règles pour désambiguïser les relations. Nous devons alors trouver un moyen de définir les relations du domaine à utiliser.

# 2. Alignement d'ontologies

Les ontologies permettent de représenter formellement un domaine particulier. Nous avons également vu qu'il est intéressant de pouvoir lier ces ontologies afin de contribuer

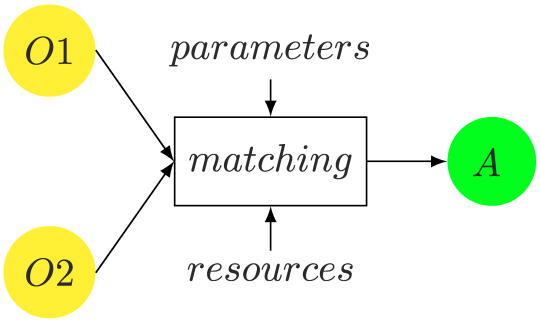


FIGURE 15 – Alignement d'ontologies [Euzenat and Shvaiko, 2007]

au Web de données liées. Ces liens représentent des équivalences sémantiques entre deux éléments d'ontologies distinctes.

Nous souhaitons pouvoir exploiter des éléments provenant de plusieurs sources. Il est de ce fait nécessaire de pouvoir détecter les éléments qui sont similaires sémantiquement entre les sources après les avoir transformés. Nous allons étudier ici le processus automatique de création de ces relations, les différentes approches de la littérature et les outils réalisant ce processus.

## 2.1. Définitions

La figure 15 présente le processus général de l'alignement d'ontologies. Nous pouvons observer sur cette figure les deux ontologies à aligner O1 et O2, qui sont deux entrées du processus de "matching" ou processus d'alignement. L'alignement A est le résultat de l'alignement entre O1 et O2. Le processus d'alignement nécessite la définition d'un certain nombre de paramètres qui permettent de déterminer des pondérations ou des seuils dans les différentes stratégies utilisées dans le processus. Ce processus peut aussi utiliser des ressources externes pour, entre autres, la désambiguïsation.

L'objectif est donc de créer un alignement qui est un ensemble de correspondances. D'après [Euzenat and Shvaiko, 2007], une correspondance est un quadruplet $< id, e1, e2, r >$. L'id est un identifiant pour la correspondance. Les deux composantes $e1$ et $e2$ représentent les deux éléments des ontologies en entrée mises en correspondance. La dernière composante $r$ est le type de correspondances, une correspondance pouvant être de différents types :

**Equivalence :** permet de définir une équivalence sémantique entre les deux éléments

**Plus général :** permet de définir une relation de généralisation sémantique entre les deux éléments

**Plus spécifique :** permet de définir une relation de spécialisation sémantique entre les deux éléments

**Disjonction :** permet de définir une relation de disjonction. Par exemple la classe "Plante" et la classe "Abribus" peuvent être déclarées disjointes. Cela signifie qu'un individu ne pourra pas être du type "Plante" et du type "Abribus".

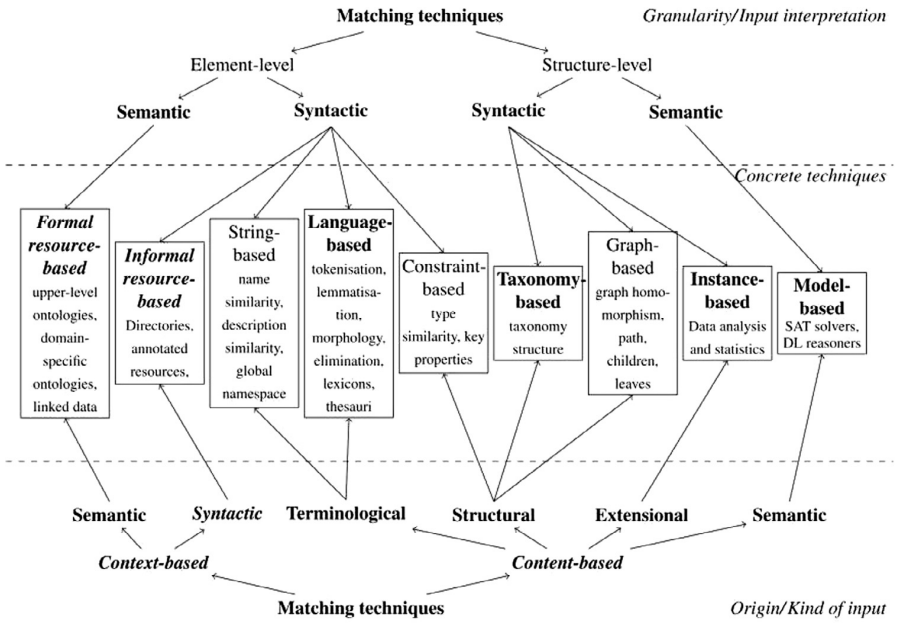


FIGURE 16 – Stratégies d'alignement d'ontologies [Shvaiko and Euzenat, 2013]

Ces correspondances sont aussi associées à un certain nombre de métadonnées dont la plus utilisée est le score de confiance. Chaque correspondance est associée à un score compris entre 0 et 1, représentant la confiance de la correspondance : plus ce score est proche de 1 et plus la correspondance est considérée comme fiable.

## 2.2. Stratégies

Afin de générer les correspondances pour créer l'alignement entre deux ontologies, il est possible de résumer les stratégies possibles en 9 stratégies différentes. Ces stratégies sont présentées dans la figure 16.

Ces stratégies sont regroupées en plusieurs catégories. En partant du haut du schéma, nous pouvons séparer les stratégies en deux catégories :

**Element-level :** étudie chaque élément des sources indépendamment les uns des autres

**Structure-level :** étudie la structure des ontologies pour déterminer des liens de correspondances en fonction de l'emplacement des éléments dans la structure de l'ontologie

Chacune de ces catégories peut être de nouveau séparée en deux sous-catégories :

**Semantic :** utilise la sémantique formelle des éléments

**Syntactic :** utilise la terminologie des éléments

Si nous étudions les stratégies d'alignement en partant du bas du schéma présenté figure 16, nous pouvons retrouver une nouvelle catégorisation.

**Context-based :** utilise des ressources externes pour justifier les correspondances

**Content-based :** utilise les éléments internes des ontologies pour justifier les correspondances

Nous retrouvons ensuite la catégorisation Semantic/Syntactic du côté des stratégies de type "context-based". Les stratégies de type "content-based" sont catégorisées en quatre sous-catégories :

**Terminological :** utilise la terminologie des éléments

**Structural :** utilise la structure des éléments des ontologies (classes, individus, relations)

**Extensional :** utilise les individus des classes

**Semantic :** utilise la sémantique formelle des éléments

D'après ces catégorisations, nous pouvons lister les différentes stratégies :

**Formal resource-based :** utilise des ressources externes dites formelles (ontologie de haut niveau ou ontologies de domaine)

**Informal resource-based :** utilise des ressources externes informelles

**String-based :** utilise des techniques de similarité de chaînes de caractères (Levenstein, TFIDF, ...)

**Language-based :** utilise des techniques de traitement du langage naturel (tokenisation, lemmatisation, ...)

**Constraint-based :** utilise les contraintes définies dans les ontologies (domain et range des propriétés, type des attributs, ...)

**Taxonomy-based :** utilise une taxonomie (par exemple WordNet) afin de désambiguïser les relations de spécialisations

**Grap-based :** considère les ontologies comme des graphes étiquetés et utilise des techniques d'homomorphisme de graphes

**Instance-based :** utilise les individus des classes afin de déterminer des correspondances entre les classes (techniques statistiques sur les attributs des individus)

**Model-based :** utilise la sémantique formelle des ontologies et les inférences possibles

D'après [Otero-Cerdeira et al., 2015], les différents projets de système d'alignement d'ontologies utilisent tous un sous-ensemble de ces stratégies. L'intérêt est donc de pouvoir définir des stratégies plus précisément et d'agréger le résultat des différentes stratégies en posant des poids sur chacune d'elles.

## 2.3. OAEI

Il existe un grand nombre de travaux proposant des systèmes d'alignements d'ontologies [Otero-Cerdeira et al., 2015]. Afin de déterminer le système qui convient à nos besoins, nous avons étudié les résultats de la campagne d'évaluation OAEI [62]. Cette campagne propose un certain nombre de catégories afin d'évaluer les systèmes d'alignement suivant

62. Ontology Alignment Evaluation Initiative - http://oaei.ontologymatching.org

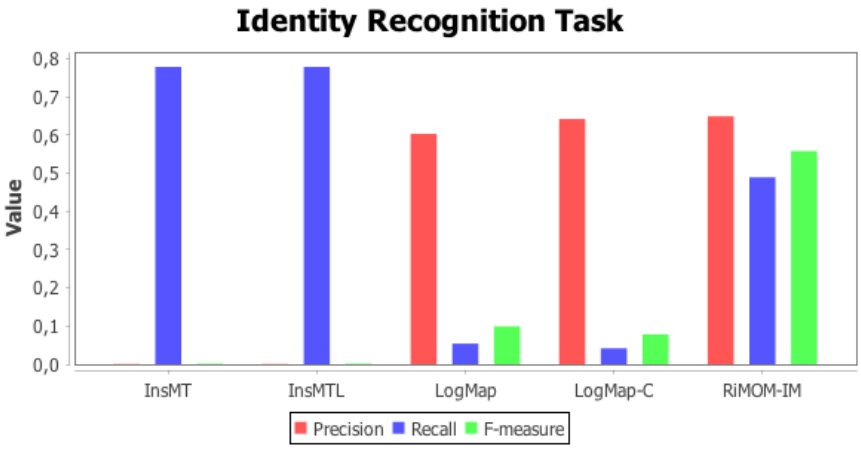


FIGURE 17 – Résultats Instance Matching - OAEI 2014 [Dragisic et al., 2014]

plusieurs critères. La campagne de 2014 [63] a proposé neuf catégories permettant de comparer des systèmes suivant différentes spécificités. La campagne "Multifarm" est par exemple spécifique pour comparer les systèmes spécialisés sur les alignements multi-langues.

Notre objectif étant de réutiliser diverses sources afin de générer une base de connaissances, il est nécessaire de pouvoir aligner des individus. Nous avons donc ciblé notre étude sur les travaux permettant l'alignement de concepts, de relations et d'individus. C'est pour cela que nous avons étudié la tâche de l'OAEI permettant la comparaison des approches sur l'alignement d'individus en plus de l'alignement de classes : "Instance Matching" [64].

En observant les résultats obtenus lors de la campagne 2014 de la catégorie Instance Matching de l'OAEI [Dragisic et al., 2014] (Cf. figure 17), nous pouvons observer des résultats intéressants sur le projet RiMOM [Li et al., 2009] et le projet LogMap [Jiménez-Ruiz and Grau, 2011]. Bien que les résultats soient plus intéressants pour le projet RiMOM que pour le projet LogMap, nous avons fait le choix de l'utilisation de LogMap car il est disponible en libre accès [65], ce qui n'est pas le cas de RiMOM.

## 2.4. LogMap

Le projet LogMap est un système permettant la génération de correspondances entre deux bases de connaissances. Le processus global de ce projet est présenté dans la figure 18.

Ce processus est constitué de cinq étapes :

**Indexation terminologique et structurelle (Lexical and Structural Indexation) :** La première étape est l'indexation terminologique des éléments constituant les ontologies en entrée. Elle indexe non seulement les chaînes de caractères associées aux

63. http://oaei.ontologymatching.org/2014/
64. http://islab.di.unimi.it/im_oaei_2014/index.html
65. https://www.cs.ox.ac.uk/isg/tools/LogMap/

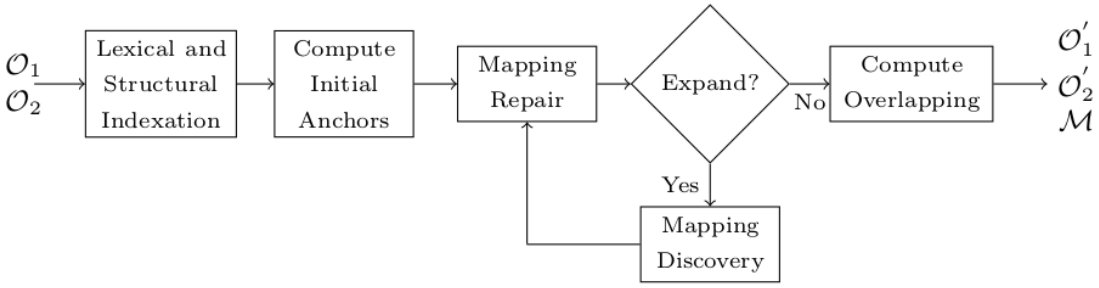


FIGURE 18 – Processus LogMap [Jiménez-Ruiz and Grau, 2011]

éléments mais aussi les éléments hiérarchiquement au-dessus et en-dessous de l'élément. Une représentation de type "interval labelling schema" [Agrawal et al., 1989] permet de repérer facilement et rapidement si un élément est une spécialisation ou une généralisation d'un autre.

**Génération des ancres initiales (Compute Initial Anchors) :** En utilisant une technique de comparaison de chaînes de caractères, des ancres sont générées entre les deux ontologies. La comparaison utilisée ici est la similarité exacte des chaînes de caractères. De cette manière, des correspondances avec une forte fiabilité sont générées et serviront de point d'entrée pour les prochaines étapes.

**Réparation des correspondances (Mapping Repair) :** En utilisant un raisonneur, LogMap essaie de déterminer les correspondances apportant un incohérence (particulièrement en utilisant les classes disjointes). De cette manière, des correspondances sont supprimées pour éliminer ces incohérences.

**Découverte des correspondances (Mapping Discovery) :** S'il existe encore des voisins non étudiés (condition "Expand"), alors chaque paire de voisins des éléments déjà étudiés est étudiée à son tour afin de déterminer s'il existe une correspondance entre-eux. Cette correspondance est définie à partir d'une comparaison terminologique des éléments, mais aussi grâce à une comparaison structurelle en utilisant la hiérarchie de généralisation et de spécialisation associée aux éléments. S'il existe une correspondance entre deux nouveaux éléments, alors cette correspondance est ajoutée dans la base des correspondances et les éléments sont ajoutés dans les éléments à étudier. Un score de fiabilité est associé à chacune des correspondances suivant les similarités terminologique et structurelle. L'étape de réparation des correspondances est de nouveau effectuée avec cette nouvelle base de correspondances.

**Calcul de l'intersection (Compute Overlapping) :** Lorsqu'il n'existe plus de paire d'éléments à étudier dans le voisinage des éléments actifs, alors deux sous-ensembles (un pour chaque ontologie) sont générés. Ces sous-ensembles représentent les éléments qui n'ont pas été mis en correspondance dans les étapes précédentes. De cette manière, un expert peut intervenir uniquement sur ces sous-ensembles pour

ajouter des correspondances.

L'outil LogMap permet de répondre aux besoins de notre processus. En effet, cet outil permet non seulement de mettre en correspondance des classes de l'ontologie mais également des individus dans la partie assertionnelle. De plus, la participation de cet outil à la campagne d'évaluation OAEI a permis de mettre en évidence ses bons résultats. Après exprimentation de l'outil, nous avons pu observer qu'un seul type de correspondances était détecté : les équivalences. Nous considérerons donc dans la suite de ce manuscrit que les correspondances obtenues par l'outil LogMap ne sont que des équivalences.

Ces correspondances nous permettent de détecter les éléments communs à plusieurs sources. Nous cherchons alors à fusionner ces éléments pour obtenir l'élement qui sera présent dans la base de connaissances finale à représenter.

# 3. Fusion de bases de connaissances

## 3.1. Fusion d'ontologies

Nous considérerons dans ce manuscrit la définition de *fusion* telle qu'elle est définie dans les travaux [Pottinger and Bernstein, 2003] :

> En considérant deux modèles A et B et un ensemble de correspondances $Map_{AB}$, le processus de fusion génère un troisième modèle représentant l'union sans doublon des modèles de A et B conformément aux correspondances de $Map_{AB}$.

Cette définition est suffisamment générique pour considérer comme modèle plusieurs types de sources, telles que des bases de données, des diagrammes UML ou encore des ontologies. La notion de "union sans doublon" est particulièrement importante dans cette définition. Eliminer les doublons lors d'une fusion de deux modèles peut être un processus complexe. Prenons un exemple simple de définition du nom d'une personne. Si le modèle A définit le nom comme un seul attribut alors que le modèle B définit le nom de famille et le prénom, la fusion naïve de ces attributs peut amener à avoir les trois attributs dans le modèle final.

Afin d'étudier les travaux traitant de cette notion de fusion de bases de connaissances, nous avons défini quatre critères :

**Symétrique :** La notion de fusion symétrique implique que les deux modèles à fusionner ont la même importance. Il est possible d'utiliser une technique de fusion asymétrique pour privilégier un modèle plutôt qu'un autre lors de choix à faire dans la modélisation du modèle résultat.

**Processus d'alignement inclus (Align.) :** Certains travaux impliquent le processus d'alignement des modèles dans leur processus alors que d'autres considèrent les correspondances comme une entrée du système.

**Conflits :** Certains travaux prennent en compte la notion de conflits pouvant exister, un conflit étant une fusion impliquant une incohérence dans le modèle.

| Projet | Symétrique | Align. | Conflits | Confiance |
|---|---|---|---|---|
| Vanilla [Pottinger and Bernstein, 2003] | oui | non | oui | non |
| Prompt [Noy and Musen, 2003] | oui | oui | oui | non |
| Atom [Raunich and Rahm, 2014] | non | non | oui | non |

TABLE 11 – Travaux sur la fusion

**Confiance :** Suivant le processus de fusion appliqué, une confiance peut être associée aux éléments fusionnés.

Nous pouvons remarquer sur le tableau 11 que seul le projet Atom présenté dans [Raunich and Rahm, 2014] propose un processus asymétrique lors de la fusion. Ce processus asymétrique permet de définir un modèle prioritaire par rapport à l'autre. De cette façon, certaines ambiguïtés susceptibles d'apparaître lors de la fusion peuvent être résolues automatiquement. C'est par exemple le cas pour notre exemple de définition du nom d'une personne. Si le modèle B est privilégié par rapport au modèle A, alors le modèle final aura deux attributs "prénom" et "nom de famille". Nous remarquons également sur le tableau 11 que seul le projet Prompt propose un système d'alignement intégré au processus de fusion. Comme nous l'avons vu dans la section précédente, il existe de nombreux systèmes d'alignement permettant d'obtenir des correspondances entre ontologies. Il n'est ici pas nécessaire de proposer un nouveau processus d'alignement mais plutôt de réutiliser des travaux existants et éprouvés.

Un point important figurant sur ce tableau est que tous les travaux proposent une gestion des conflits. Les travaux de [Pottinger and Bernstein, 2003] proposent une catégorisation des types de conflits pouvant apparaître lors du processus de fusion :

**Représentation :** un conflit de représentation apparaît quand le même élément du monde réel est modélisé sous deux formes différentes dans les deux modèles à fusionner (c'est par exemple le cas de la représentation du nom d'une personne dans notre exemple précédent)

**Modèle :** un conflit lié au modèle apparaît lorsque la fusion apporte une incohérence liée au modèle utilisé.

**Fondamentale :** un conflit fondamental apparaît lorsque la fusion apporte une incohérence par rapport à la représentation des données. Par exemple un élément ne peut être que d'un seul type fondamental (entier, chaîne de caractères, ...).

Les travaux sur Prompt [Noy and Musen, 2003] définissent les conflits de la même manière et proposent une interface spécifique pour résoudre ces conflits. Les travaux sur Vanilla [Pottinger and Bernstein, 2003] s'intéressent particulièrement à la dernière catégorie. Pour cela, les auteurs proposent un ensemble de règles permettant de résoudre ces conflits de manière générique. Ils partent de l'hypothèse que ce type de conflits est présent quel que soit le type de modèle utilisé (bases de données, ontologies ou autres).

Les travaux sur Atom [Raunich and Rahm, 2014] traitent spécifiquement des conflits du premier type : représentation. Pour cela, les auteurs utilisent un processus asymétrique afin de privilégier une modélisation plutôt qu'une autre lors du processus de fusion d'ontologies.

Nous remarquons enfin que les processus de fusion présentés ici ne traitent pas de la notion de confiance. La problématique de fusion étant toujours considérée entre deux modèles, la confiance ne peut pas apparaître puisqu'il n'y a qu'une alternative possible. La notion d'asymétrie est donc suffisante dans ce cas-là. Néanmoins, lorsque nous considérons la notion de fusion avec plus de deux sources à fusionner, nous pouvons généraliser la notion d'asymétrie en considérant l'importance des sources dans ce processus. Apparaît dans ce cas la notion de confiance d'un élément en fonction des sources impliquées et particulièrement de leurs qualités respectives.

## 3.2. Conflits lors de l'alignement

Les travaux de fusion présentés précédemment font tous la même hypothèse de départ : une correspondance est forcément de type 1 :1, c'est à dire qu'un élément d'une source ne peut être mis en correspondance qu'avec un autre élément d'une autre source. Or les systèmes d'alignement actuels permettent d'obtenir des correspondances de type 1 :n. Ces correspondances, considérées comme "complexes" d'après la définition donnée dans [Gillet et al., 2013], peuvent apporter un nouveau type de conflit. En effet, si une correspondance 1 :n apparaît entre un élément a1 de l'ontologie A et n éléments b1 à bn de l'ontologie B, alors plusieurs interprétations sont possibles :

— Erreur du système d'alignement
— Relation de subsomption avec un élément de B qui subsume b1 à bn
— Relation de méronymie avec n relations de méronymie entre a1 et les n élément de B
— Autre

Il est particulièrement difficile de déterminer la nature de la relation dans le cas de correspondances complexes de type 1 :n. Nous posons l'hypothèse simplificatrice dans nos travaux de ne pas considérer de correspondances complexes. De ce fait, nous ne chercherons pas à désambiguïser ces relations 1 :n. Nous considérerons dans ce manuscrit qu'une relation d'équivalence de type 1 :n implique n-1 relations d'équivalence fausses. D'après cette hypothèse, ces correspondances correspondent à des erreurs du système d'alignement. Les éléments fusionnés avec ces correspondances à désambiguïser ne peuvent pas apparaître ensemble dans la base de connaissances finale. Ceci nous amène à étudier les travaux sur les conflits lors de la fusion de bases de connaissances.

Les travaux connexes concernant la fusion de bases de connaissances et plus particulièrement la gestion des conflits sont assez récents. La plupart des travaux [Trojahn et al., 2011, Abbas and Berio, 2013, Raunich and Rahm, 2014] cherchent à détecter des conflits entre les différentes correspondances établies entre deux sources par une ou plusieurs approches d'alignement. L'idée sous-jacente est de déterminer quelles correspondances peuvent être ignorées dans le but de lever le conflit. Dans ces travaux, un conflit est détecté lorsque les correspondances établies rendent la base de connaissances inconsistante d'un

point de vue logique. Certains travaux [Abbas and Berio, 2013, Trojahn et al., 2011] considèrent également les préférences des deux agents utilisant chacun des ontologies pour établir ces conflits. Le traitement des conflits se fait soit par l'utilisation de règles [Raunich and Rahm, 2014], soit en cherchant des sous-ensembles compatibles, notamment en utilisant la théorie de l'argumentation [Trojahn et al., 2011].

Bien que certaines approches permettent un traitement de conflits entre correspondances, aucune ne propose le traitement de conflits entre éléments générés à partir des correspondances dans un processus de fusion. Comme nous l'avons vu dans [Pottinger and Bernstein, 2003], les processus d'alignement et de fusion ayant été séparés dans les différents travaux pour se concentrer sur l'un ou sur l'autre, les conflits sont gérés là-aussi de manière indépendante. Nous considérons néanmoins qu'il existe une notion de conflit lié à la génération de correspondances complexes par les systèmes d'alignements. La gestion d'une correspondance de type 1 :n apparaissant dans les systèmes d'alignements n'est pas prise en compte par les système de fusion actuels.

## 4. Qualité d'une source

Comme vu dans le paragraphe précédent, la notion de qualité d'une source est importante dans un processus de fusion multi-sources. Nous pouvons définir la qualité d'une source (ou la qualité des données provenant d'une source) comme proposé dans [Jayawardene et al., 2013] par :

> Des données de bonne qualité résultent d'une bonne représentation du monde réel.

La notion de "bonne représentation" est intentionnellement subjective, puisqu'une bonne représentation est une représentation adaptée à l'usage que l'on souhaite faire des données. De cette manière, il est évident qu'une source sera de qualité si elle permet de remplir un objectif donné.

Dans un processus de fusion multi-sources, la notion de qualité implique la notion de confiance d'un élément fusionné. En effet, si un élément est généré à partir de la fusion d'éléments provenant de sources de qualités variables, alors cet élément a une confiance elle aussi variable. Plusieurs définitions de la notion de confiance en informatique et sur le Web sémantique sont présentées dans [Artz and Gil, 2007]. Celle qui correspond le mieux à notre propos est :

> La confiance d'une partie A à une partie B pour un service X est la croyance mesurable de A dans B pour une période déterminée dans un contexte spécifique (par rapport au service X).

Considérons $A$ comme étant l'utilisateur qui veut créer une base de connaissances, $B$ étant une source et $X$ le processus d'extraction. Dans cette définition, la confiance concerne la source $B$, avec le processus d'extraction $X$, qui génère des éléments fusionnés avec un score de confiance associé.

En étudiant la littérature, nous avons noté que plusieurs domaines de recherche ont étudié la notion de qualité d'une source. Le domaine qui s'est particulièrement intéressé

à cette problématique est la communauté des bases de données [Strong et al., 1997]. En définissant des patrons de qualité, les travaux de [Strong et al., 1997] dressent un portrait de ce qui peut être considéré comme une base de données de qualité. Dans ces travaux, la qualité d'une base de données dépend directement des problèmes rencontrés lors de l'utilisation de ces données. En énumérant les différents problèmes pouvant subvenir, ils mettent en évidence les critères définissant des données de qualités. Ces travaux permettent de donner un cadre grâce aux patrons définis pour représenter les problèmes rencontrés, sans chercher l'exhaustivité de cette représentation.

Les taxonomies comme définies dans la section 2.4.1 du chapitre I, et particulièrement celles modélisées avec le vocabulaire SKOS, ont aussi été étudiées afin de déterminer la qualité de ce type de source [Suominen and Hyvönen, 2012]. Le projet SKOSIFY [Suominen and Hyvönen, 2012] prend particulièrement en compte l'analyse de la structure de la source pour établir sa qualité. En effet, certaines erreurs de conception peuvent générer des ambiguïtés lors de l'interprétation des sources SKOS. Le projet SKOSIFY propose un outil pour corriger automatiquement un ensemble d'erreurs de conception pour en améliorer la qualité. Nous pouvons prendre l'exemple de la contrainte présente dans SKOS définissant l'unicité d'un label préféré par langue. Le projet SKOSIFY propose dans ce cas de remplacer les labels préférés d'une même langue en labels alternatifs.

La problématique de l'accessibilité des données sur le Web a aussi été considérée comme étant une façon de représenter la qualité d'une source. Les travaux de [Gil and Artz, 2007] présentent une liste de critères pouvant intervenir dans la définition de la qualité de ce type de source. Les auteurs de ces travaux se sont particulièrement intéressés à l'impact des retours des utilisateurs de ces données pour déterminer la qualité d'une source.

Plus récemment est apparue l'étude de critères de qualité définissant la qualité d'une base de connaissances, ou du moins d'une source présente sur le Web de données liées [Artz and Gil, 2007]. De la même manière que SKOSIFY, un outil a été proposé pour détecter les erreurs de modélisation apparaissant dans une base de connaissances [Poveda-Villalon et al., 2012]. Le projet OOPS ! dresse une liste d'erreurs pouvant apparaître dans la conception d'une base de connaissances. Ce projet propose également une résolution automatique de certaines de ces erreurs et permet de mettre en évidence celles qui ne peuvent être corrigées automatiquement.

Ce qui apparaît dans cette étude est la diversité et le nombre conséquent de critères pouvant être utilisés. Chaque type de sources ayant des attributs spécifiques, il est difficile de répondre à la question : "qu'est ce qu'une source de qualité ?". De plus, bien que les différents travaux présentent de longues listes de critères, les auteurs précisent que ces listes ne sont ni exhaustives, ni suffisantes. Comme nous l'avons vu précédemment, la définition d'une source de qualité étant particulièrement dépendante de l'usage que l'on souhaite en faire, il est complexe de définir la liste de tous les critères pour tous les usages de réutilisation.

Les travaux présentés dans [Jayawardene et al., 2013] essaient de faire l'état des différents aspects pouvant intervenir dans la définition de la qualité d'une source quelle qu'elle soit. Ces travaux tentent de définir des critères se recoupant entre les différents domaines utilisant la qualité d'une source. Là-encore, l'exhaustivité n'est pas réaliste.

Conscient de la difficulté de dresser une liste exhaustive, nous présentons dans le tableau 12 les critères que nous retenons lorsqu'il s'agit d'étudier la qualité de sources pouvant être exploitées pour générer une base de connaissances.

| | | |
|---|---|---|
| Conception | Origine | Provenance |
| | | Expertise |
| | | Nécessité de qualité |
| | Accessibilité | Données ouvertes |
| | | Format |
| | | Qualité de la présentation/documentation |
| Utilisation | Popularité | Beaucoup utilisé |
| | | Données liées |
| | | Source de référence |
| | | Réputation |
| Contenu | Fraîcheur | Dernière MAJ |
| | | Fréquence MAJ |
| | | Nécessité MAJ |
| | Validité | Syntaxe |
| | | Contraintes |
| | Précision | Granularité |
| | | Ambiguïté |
| | Couverture | Couverture du domaine |

TABLE 12 – Critères de qualité d'une source

Ces critères sont regroupés en trois catégories : conception, utilisation et contenu. La catégorie concernant la conception de la source s'intéresse à l'origine des données présentes dans la source, c'est à dire la provenance du contenu de la source etl'expertise des auteurs sur le domaine traité. De plus, il est important de considérer la nécessité de réutiliser des sources de qualité. Cette nécessité peut être variable suivant l'objectif poursuivi lors de la création de cette source. Il se peut que la base de connaissances à générer n'ait pas besoin d'être d'une qualité irréprochable, mais qu'elle soit exhaustive. C'est par exemple le cas de DBPedia, comme nous l'avons vu dans la section 2.4.4 du chapitre I. L'accessibilité est un autre aspect lié à la conception de la source. Des données qui ne sont pas disponibles ne sont pas des données réutilisables facilement. Il est donc important de s'intéresser à l'ouverture des données, au format utilisé (particulièrement si c'est un format ouvert) et à la qualité de la documentation. Ce dernier point permet de savoir s'il sera possible de réutiliser la source facilement ou non.

L'aspect utilisation de la source est aussi un bon indicateur de la qualité de celle-ci. Si une source a été plusieurs fois réutilisée dans d'autres projets, cela donne une bonne indication sur sa qualité. Il en est de même si cette source est liée à de nombreuses autres sources (Cf. section 1.3 du chapitre I). La réputation de la source est également un critère important. Ce que l'on appelle réputation est l'avis d'une communauté sur la qualité de la source. Le fait que la source soit de référence signifie qu'une instance officielle a définit

cette source comme étant celle qui fait autorité dans un domaine pour une problématique donnée.

Le contenu de la source est la catégorie la plus importante pour définir sa qualité. Il est tout d'abord important de savoir si la source est à jour et si elle le sera dans le futur, ou encore si avoir des données à jour est un objectif important pour les auteurs de la source. La validité de la source est équivalente aux outils "OOPS!" et "SKOSIFY" puisqu'ici, on s'intéresse à la validité syntaxique du langage utilisé pour représenter les données mais aussi au respect des différentes contraintes. Ces contraintes peuvent être sur le format d'implémentation (Cf. section 2.4 du chapitre I) ou sur le modèle. La granularité des données permet de savoir si le niveau de spécialisation de la source est suffisamment détaillé. L'ambiguïté des données fait référence aux ambiguïtés pouvant apparaître dans la modélisation des sources. Par exemple les relations "broader" d'un thésaurus sont ambiguës. Enfin, le dernier critère étant la couverture puisque nous souhaitons réutiliser cette source pour une problématique donnée, il est important qu'elle corresponde au domaine que la base de connaissance à générer doit couvrir.

L'évaluation de ces critères est particulièrement subjective. En effet, cette évaluation dépend non seulement de l'objectif souhaité, mais aussi d'un avis d'expert. Seul le critère sur la syntaxe peut être évalué automatiquement, tous les autres nécessitant un avis d'expert pour déterminer la valeur du critère.

## 5. Conclusion

Nous avons pu analyser les différentes méthodes de transformation de thésaurus et de bases de données. Les méthodes totalement automatiques ne permettent pas d'obtenir un résultat suffisant puisqu'elles ne prennent pas en compte les spécificités des modélisations des sources. Les méthodes totalement manuelles sont, elles, trop fastidieuses pour les sources de grande taille. Les travaux proposés par [Villazon-Terrazas et al., 2010] présentent un bon compromis en proposant la définition d'un patron de transformation qui sera utilisé pour effectuer la transformation automatiquement. Il est néanmoins nécessaire de pouvoir définir ce patron et principalement les éléments à désambiguïser. Cette définition doit aussi permettre de cibler les éléments à transformer pour ne garder que ce qui correspond aux besoins de la base de connaissances à générer.

Comme nous avons pu l'observer lors de l'étude des travaux portant sur la fusion d'ontologies, il est nécessaire, là aussi, de spécifier un modèle qui sera nécessaire pour orienter la fusion afin d'éviter les incohérences et les doublons. Les différents travaux traitant des incompatibilités lors d'un processus de fusion ne s'intéressent qu'aux incompatibilités entre correspondances et non entre éléments fusionnés.

Afin de prendre en compte la qualité des sources lors de la définition de la confiance d'un élément fusionné, il est nécessaire de dresser une liste de critères pouvant intervenir dans cette définition de la qualité. Néanmoins, il n'existe pas, à l'heure actuelle, de liste exhaustive de critères permettant de définir objectivement ce qu'est une source de qualité. La difficulté pour dresser une telle liste est que les critères sont très spécifiques aux besoins pour un usage donné.

# Troisième partie .
# Contributions scientifiques

La section contexte (Cf. section I) nous a permis de mettre en évidence la nécessité de créer une base de connaissances exploitant la quantité de sources disponibles dans un domaine donné. La section état de l'art a montré qu'aucune méthode n'a été proposée pour aider à la construction de bases de connaissances en considérant simultanément plusieurs sources existantes. Or l'intérêt d'une telle approche serait de pouvoir comparer les éléments présents dans les différentes sources. Nous faisons l'hypothèse que des éléments communs à plusieurs sources font partie d'un consensus sur le domaine et que leur présence dans la base de connaissances finale est souhaitable.

Nous présentons dans ce chapitre une méthodologie visant à construire une base de connaissances en nous fondant sur cette hypothèse. Nous présenterons dans un premier temps notre processus général, qui est le résultat de la fusion de deux scénarios de la méthodologie NeOn. Cette méthodologie se décompose en trois étapes : (1) l'analyse des sources, (2) la transformation des sources et (3) la fusion de bases de connaissances. Nous présenterons chacune de ces trois étapes dans des sections distinctes. Nous présenterons finalement la manière d'exporter le résultat obtenu pour une mise à disposition sur le Web de données liées (LOD).

## 1. Processus général

Comme nous l'avons vu dans la partie consacrée à la présentation du contexte de ces travaux (Cf. section 3 du chapitre I), nous situons notre proposition dans le cadre de la méthodologie NeOn [Suárez-Figueroa et al., 2012]. Les scénarios qui nous ont particulièrement intéressés sont le scénario 7, qui permet la construction d'ontologie à partir de patrons de conceptions, et le scénario 2, qui permet la transformation de sources non-ontologiques. La démarche permettant de les fusionner est décrite dans la section 1.1. Sur la base de ces principes de fusion, nous avons défini notre méthodologie, décrite dans la section 1.2.

### 1.1. Fusion de scénario NeOn

Comme présenté dans le contexte (Cf. section 3 du chapitre I), nous avons fait le choix de définir notre méthodologie en fonction de la méthodologie NeOn. Notre choix s'est plus spécifiquement porté sur les scénarios 7 et 2 que nous présenterons ici. Nous présenterons ensuite la façon dont nous les avons fusionnés pour obtenir la méthodologie appliquée dans notre approche.

### 1.1.1. Scénario 7 : Réutilisation de patrons de conception ontologiques (ODP)

Ce scénario permet la construction d'une base de connaissances en réutilisant des patrons de conception existants. Cette réutilisation est particulièrement bénéfique puisqu'elle permet de construire une base de connaissances en réutilisant des efforts de réflexions faits par d'autres, tout en augmentant la qualité de la modélisation. En effet, certains aspects de modélisation peuvent se retrouver dans plusieurs bases de connaissances. Ceci est par exemple le cas avec un patron de conception concernant la définition de la propriété modélisant une attaque d'un bio-agresseur. Pour cela, la FAO a défini le patron de conception : "hasPest" [66]. Il est donc intéressant de regarder si cet aspect n'a pas été déjà modélisé dans un patron de conception, afin de ne pas refaire le travail.

C'est ce que propose de faire ce scénario [Poveda Villalon et al., 2010] 7. Il est divisé en plusieurs tâches :

**Tâche 1 :** Identifier les besoins fonctionnels

**Tâche 2 :** Identifier les patrons de conception disponibles

**Tâche 3 :** Diviser et transformer le problème en sous-problèmes

**Tâche 4 :** Associer les sous-problèmes aux patrons de conception

**Tâche 5 :** Sélectionner les patrons de conception

**Tâche 6 :** Appliquer les patrons de conception

La tâche 1 permet de dresser une liste des besoins que la base de connaissances finale devra respecter. Plus précisément, les besoins fonctionnels représentent les questions auxquelles la base de connaissances devra répondre. Ces besoins fonctionnels sont représentés sous forme de questions, comme par exemple "Quels sont les sous-rangs taxonomiques de Triticum ?". Un besoin non-fonctionnel serait par exemple "la base de connaissances doit être modulaire". Les besoins non-fonctionnels seront définis lors de l'exécution du scénario 1 après ce scénario (Cf. section 3). Une fois que ces questions sont définies, la tâche 2 permet de repérer les patrons de conception pouvant intervenir pour définir la modélisation ou le vocabulaire qui permettra de représenter les réponses à ces questions. Pour cela, plusieurs entrepôts de patrons de conception peuvent être utilisés, comme par exemple :

— Ontology Design Patterns (ODP) Portal [67]
— W3C Semantic Web Best Practices and Deployment (SWBPD) [68]

La tâche 3 permet de représenter la base de connaissances à modéliser comme plusieurs "sous-ontologies". Si nous avons par exemple dans les besoins la question "Quels sont les sous-rangs taxonomiques de Triticum ?", alors nous avons un sous-problème qui est "la représentation de la taxonomie des Triticum", et si nous avons la questions "Quelle est la couleur de la fleur de colza ?", alors nous pouvons définir le sous-problème "phénotype des plantes". Pour chacun de ces sous-problèmes, des patrons de conception sont identifiés comme aidant à la modélisation de ce problème (tâche 4). La tâche 5

66. http://ontologydesignpatterns.org/wiki/Submissions:HasPest
67. http://ontologydesignpatterns.org/
68. http://www.w3.org/2001/sw/BestPractices/

permet de sélectionner les patrons de conception qui sont réellement pertinents dans la modélisation du sous-problème et éviter les doublons entre plusieurs sous-problèmes (un même patron de conception pouvant correspondre à plusieurs sous-problèmes). La dernière tâche permet de fusionner et de modéliser ces patrons de conception pour obtenir la base de connaissances souhaitée.

### 1.1.2. Scénario 2 : Réutilisation d'une source non-ontologique

Ce scénario permet la réutilisation d'une source non-ontologique (RNO). Ce scénario repose sur des patrons de transformation. Il n'est pas réaliste de proposer un patron générique pour tous les types de RNO. Néanmoins, certaines sources ont des similarités, que ce soit en termes d'implémentation ou de modélisation. Nous pouvons par exemple citer les thésaurus et les taxonomies qui sont deux types de sources très proches. Il est alors possible de définir des patrons de transformation pouvant être réutilisés en exploitant ces similarités. C'est ce que propose ce scénario à travers 2 activités :

**Activité 1 :** Ingénierie inversée de RNO

**Activité 2 :** Transformation de RNO

La première activité permet d'analyser la RNO à réutiliser et de représenter cette RNO suivant différents niveaux d'abstraction. En d'autres termes, cette activité vise à identifier ce qui concerne la modélisation des données, la structuration de l'implémentation ou de la syntaxe d'implémentation, comme vu dans la section 2.4 du chapitre I. Pour cette première activité, 3 tâches sont définies :

**Tâche 1.1 :** Récupération des données

**Tâche 1.2 :** Abstraction conceptuelle

**Tâche 1.3 :** Exploration de l'information

La tâche 1.1 permet de récupérer toutes les données et la documentation qui est en lien avec la RNO. La tâche 1.2 permet de déterminer le modèle qui a été utilisé pour représenter les données. Ce modèle peut être précisé dans la documentation, sinon, elle devra être reconstruite manuellement. Par exemple pour la taxonomie NCBI présentée précédemment, il est nécessaire de découvrir que le modèle est une base de données avec les taxons regroupés dans le fichier "nodes.dmp" et des labels dans "names.dmp". De la même manière, le format d'implémentation des données doit être défini dans la tâche 1.3, à partir de la documentation ou reconstruite manuellement. Toujours pour la source NCBI, ce format d'implémentation est le TSV.

L'activité 2 (Transformation de RNO) permet la transformation de la RNO en fonction des éléments déterminés dans l'activité 1. Cette transformation se fera en fonction des patrons de transformation déjà existants si c'est possible. Pour cela, 3 activités sont définies :

**Tâche 2.1 :** Recherche d'un patron de transformation adapté

**Tâche 2.2a :** Application de la transformation si un patron a été trouvé

**Tâche 2.2b :** Définition d'une méthode de transformation ad-hoc et généralisation de cette méthode afin de créer un nouveau patron si aucun patron adapté n'a été trouvé

**Tâche 2.3 :** Corrections manuelles

La tâche 2.1 permet de chercher un patron de transformation pouvant s'appliquer à notre RNO. Pour cela, il est possible d'effectuer cette recherche dans l'ODP Portal [69]. Ce portail regroupe des patrons de conception comme l'indique son nom, mais aussi des patrons de transformation. Ces patrons sont spécifiques à la modélisation des données, au format de l'implémentation et à l'approche générale de la transformation.

Par exemple, dans le cas où la RNO est un thésaurus suivant la norme (ISO 2788)2.4.2 du chapitre I, le patron de transformation suivant peut être utilisé :
https://tinyurl.com/phss8kj Ce patron propose un processus de transformation sous la forme d'un algorithme à appliquer sur le thésaurus pour obtenir la base de connaissances finale. Cet algorithme est le suivant :

**1.** Récupérer tous les regroupements de termes qui n'ont pas de relation "broader term"

**2.** Pour chacun de ces regroupements ti, faire :

**2.1** Créer la classe, Ci, dans la base de connaissances correspondant à ce regroupement de termes, si elle n'a pas déjà été créée

**2.2** Récupérer tous les regroupements de termes qui ont la relation "narrower term" vers ti

**2.3** Pour chacun de ces regroupements, tj, faire :

**2.3.1** Créer la classe Cj correspondante si elle n'a pas déjà été créée

**2.3.2** Utiliser une source externe pour déterminer la relation entre Cj et Ci

**2.3.3** Revenir à 2.3.1 tant qu'il y a des tj

**2.4** Récupérer tous les regroupements de termes qui ont la relation "related term" vers ti

**2.5** Pour chacun de ces regroupements, tr, faire :

**2.5.1** Créer la classe Cr correspondante, si elle n'est pas déjà créée

**2.5.2** Utiliser une source externe pour déterminer la relation entre Cr et Ci

**2.5.3** Revenir à 2.5.1 tant qu'il y a des tr

**2.6** Récupérer tous les regroupements de termes qui ont la relation "equivalent term" vers ti

**2.7** Pour chacun de ces regroupements, tq, faire :

**2.7.1** Créer la classe Cq correspondante, si elle n'est pas déjà créée

**2.7.2** Utiliser une source externe pour déterminer la relation entre Cq et Ci

**2.7.3** Revenir à 2.5.1 tant qu'il y a des tq

69. Ontology Design Patterns (ODP) Portal - http://ontologydesignpatterns.org/

S'il existe un patron de transformation adapté à la source, alors nous l'appliquons avec la tâche 2.2a. S'il n'existe pas de patron de transformation adapté, alors nous appliquons la tâche 2.2b. Cette tâche permet de créer une méthode de transformation ad-hoc, que nous essaierons ensuite de généraliser pour créer un nouveau patron de transformation qu'il sera possible de soumettre sur le site ODP. La tâche 2.3 permet de reprendre la transformation effectuée pour corriger manuellement les différentes erreurs qui peuvent apparaître. Nous retrouvons par exemple dans Agrovoc la relation "narrower term" entre "Cows" [70] et "Cow milk" [71]. Si cette relation est traduite par une relation "rdfs :subClassOf", une erreur apparaît puisque "Cow milk" n'est pas une spécialisation de "Cows".

Ce scénario présente le processus à suivre pour transformer une RNO en base de connaissances. Les patrons de transformation permettent de spécifier une transformation pour un type de source et une modélisation donnée. Ces patrons sont souvent trop génériques pour être appliqués directement sur une source. Il est alors nécessaire de les spécialiser pour les adapter aux besoins de la transformation à effectuer.

### 1.1.3. Fusion des scénarios 2 et 7

Le scénario 7 de NeOn propose une méthodologie pour la construction d'une base de connaissances en réutilisant des patrons de conception. D'après [Alexander, 1979], un patron de conception est un ensemble de lignes directrices pouvant aider à la résolution d'un problème de modélisation. Cette notion de patrons de conception a ensuite été adaptée à la conception d'ontologies [Gangemi, 2005, Svatek, 2004]. Il est intéressant de réutiliser des patrons de conception pour la modélisation de la taxonomie des plantes puisque la FAO [72] a proposé des patrons de conception liés à cette problématique. Ces patrons ont été proposés lors d'une étude sur la création d'une base de connaissances fondée sur Agrovoc : Agrontology [Caracciolo et al., 2013]. Ces patrons de conception sont disponibles à l'adresse suivante : `http://ontologydesignpatterns.org/wiki/Submissions:AOS_AGROVOC_Concept_Server_fundation_ontology_model`.

Le scénario 2 de la méthodologie NeOn propose d'exploiter des sources non-ontologiques pour générer une base de connaissances, ces transformations se fondant sur des patrons de transformation pour détecter la méthode la plus adaptée[Villazon-Terrazas et al., 2010] (Cf. section 1 du chapitre II).

L'utilisation de sources non-ontologiques est ici un pré-requis pour nos travaux, et l'utilisation de patrons de conception est considérée comme une bonne pratique dans le développement d'une ontologie[Gangemi and Presutti, 2009]. C'est pour cela que nous proposons une fusion des deux scénarios NeOn présentés dans la section 3 du chapitre I.

La figure 19 présente notre proposition de fusion des scénarios 2 et 7 de NeOn. Les modifications que nous avons apportées apparaissent sur cette figure en pointillés. Le scénario 7 n'a été que très peu modifié. Il propose d'abord de définir le projet ainsi que son périmètre et de rechercher ensuite des patrons de conception qui peuvent être

70. Vâches
71. Lait de vâche
72. Food and Agriculture Organization of the United Nations - `www.fao.org`

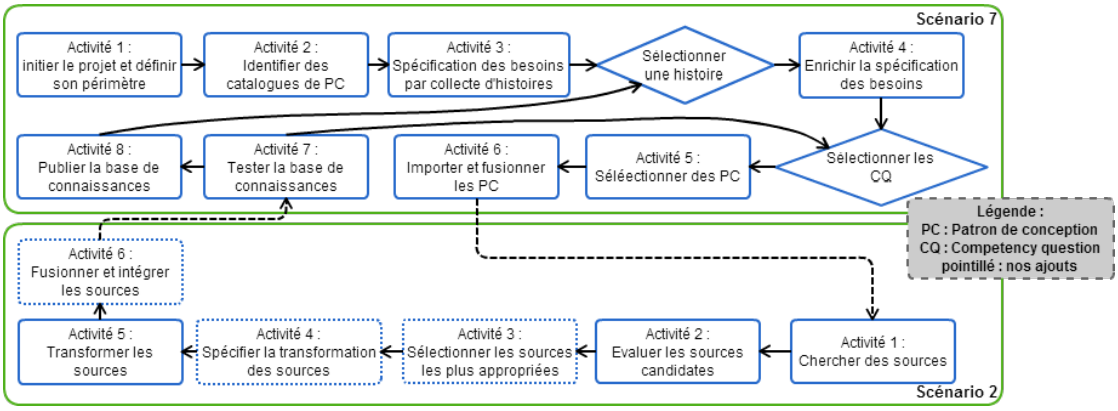


FIGURE 19 – Fusion du scénario 2 et du scénario 7 de NeOn

appliqués au projet. Par la collecte d'histoires, un ensemble de "competency questions" sont écrites pour définir les besoins de la base de connaissances à créer. Une histoire est la représentation sous forme textuelle d'un besoin fonctionnel. À partir d'une histoire est créée une question ("competency question") qui servira de cahier des charges pour la construction de la base de connaissances et pour vérifier, à la fin du processus, que tous les besoins initiaux ont trouvé une réponse. Nous modifions le scénario à cet endroit de façon à inclure le scénario 2 de NeOn directement dans le processus du scénario 7. De cette manière, nous pouvons intégrer l'enrichissement de la base de connaissances créée par réutilisation de sources.

Le scénario 2 propose de sélectionner des sources contenant les informations que nous souhaitons réutiliser pour créer notre base de connaissances, de les évaluer puis de trouver une méthode de transformation en utilisant des patrons de transformation [Villazon-Terrazas et al., 2010]. Nous avons modifié ce scénario en ajoutant certaines activités. L'activité 3 permet de faire une sélection des sources. En effet, le scénario proposé par NeOn prend en compte le fait qu'il n'existe qu'une source à transformer. Sont disponibles sur le Web plusieurs sources potentielles, comme observé dans la section 2.4 du chapitre I. Cette activité permet de déterminer les sources pertinentes et celles qui ne le sont pas. L'activité 4 permet de spécifier la transformation de la source en fonction de la base de connaissances créée dans le scénario 7. L'activité 6 est ajoutée pour fusionner les différentes sources sélectionnées. Cette phase de fusion propose également une intégration des sources à la base de connaissances créée dans le scénario 7, l'idée étant d'enrichir l'ontologie créée dans le scénario 7 avec la transformation des sources non-ontologiques. Une fois cette étape effectuée, le processus retourne sur le scénario 7 pour finir de tester l'ontologie et vérifier qu'elle réponde bien aux besoins du projet, autrement dit de vérifier que les "competency questions" ont toutes les réponses attendues.


FIGURE 20 – Processus général

### 1.2. Description du processus

D'après la fusion des scénarios présentée dans le paragraphe précédent, nous proposons un processus général. Nous avons réutilisé les différentes étapes de cette fusion afin de définir notre méthodologie. La fusion des deux scénarios peut être représentée sous la forme d'un processus général reprenant les activités importantes. Ce processus général, présenté sur la figure 20, est composé de trois étapes :

- **1 Analyse de sources :** Pendant ce processus, l'expert du domaine sélectionne les sources les plus appropriées pour la construction de la base de connaissances. Il inspecte chaque source potentielle afin d'évaluer sa couverture, mais également afin d'évaluer la faisabilité de la transformation automatique en base de connaissances. Cette étape correspond aux activités 1, 2 et 3 du scénario 2 de NeOn.
- **2 Transformation de sources :** Ce processus permet de spécifier la transformation à appliquer sur chaque source pour obtenir une base de connaissances au format OWL. Une fois ces spécifications définies, la transformation est appliquée et nous obtenons ce que nous appelons une base de connaissances source. Cette étape correspond aux activités 4 et 5 du scénario 2 de NeOn.
- **3 Fusion de bases de connaissances :** Ce processus construit la base de connaissances finale en se fondant sur toutes les bases de connaissances extraites à partir des sources. À notre connaissance, ce processus n'est proposé dans aucune des méthodes d'ingénierie d'ontologies. Généralement, les méthodes d'ingénierie d'ontologies utilisent plusieurs sources séparément afin d'enrichir la base de connaissances de manière séquentielle. Le processus de fusion utilise plusieurs bases de connaissances simultanément, afin d'extraire des connaissances communes aux différentes sources. Cette étape correspond à l'activité 6 du scénario 2 de NeOn.

Le scénario 7 intervient dans la création de la partie haute d'un module ontologique que nous présenterons dans la section 3.1.

Nous allons, dans la suite de ce chapitre, présenter plus en détails chacune des étapes.

## 2. Analyse des sources

Cette phase préliminaire du processus permet de dresser la liste des sources qui seront considérées dans la suite. Cette étape reprend les activités 1, 2 et 3 du scénario 2 présenté

dans la figure 19. Afin de pouvoir sélectionner les sources les plus adaptées, il est important de spécifier pourquoi une source est adaptée et quels sont les critères à observer sur une source pour la sélectionner.

Ce que nous appelons ici une source adaptée est une source qui correspond aux besoins de l'objectif de la transformation. En d'autres termes, une source sera considérée comme pouvant être une source du processus si elle correspond au niveau de qualité et aux objectifs souhaités dans la base de connaissances finale.

À cette fin, nous avons défini quatre critères principaux que nous utilisons pour interagir avec l'expert. Ces critères n'englobent pas tous les critères cités dans le tableau 12 mais représentent ceux que nous considérons les plus pertinents par rapport à notre méthodologie.

— **La réputation de la source** est généralement fondée sur la réputation de ses auteurs. Si l'auteur est une institution gouvernementale ou internationale, la source sera plus probablement acceptée par une large communauté d'utilisateurs et sera réutilisée. La réputation de la source peut également être fondée sur le nombre de ses utilisateurs.

— **La fraîcheur de la source** se fonde sur la dernière date de mise à jour de la source, ainsi que sur le fait que la source est souvent mise à jour ou pas.

— **L'adéquation de la source à la cible** représente la similarité entre la source et la base de connaissances souhaitée. Elle prend en compte la couverture de la source par rapport à la base de connaissances finale.

— **La clarté du modèle de la source** évalue le fait que le modèle conceptuel de la source peut être facilement trouvé. Par exemple, si différents patrons sont utilisés pour stocker le même type d'informations, alors le modèle est ambigu.

Nous nous limitons à ces seuls critères pour guider l'expert dans la sélection des sources car, comme observé dans la section 4 du chapitre II, il est très difficile d'obtenir une liste exhaustive de critères pouvant intervenir dans la caractérisation d'une source de qualité. Il n'est, de plus, pas envisageable de demander à un expert d'évaluer une longue liste de critères pour chaque source. Néanmoins, les quatre critères cités précédemment permettent d'observer de manière synthétique les éléments importants entrant en compte dans la description d'une source de qualité dans notre processus.

Notre objectif dans ce processus est de trouver des connaissances communes à plusieurs sources. Les différentes sources ne présentent pourtant pas nécessairement le même intérêt en fonction de leur qualité. De ce fait, une connaissance provenant d'une source de bonne qualité aura plus de poids qu'une autre provenant d'une source de moins bonne qualité. Nous considérerons ici la définition d'une source de bonne qualité donnée dans [Dong et al., 2013]. Cette définition précise qu'une source est d'autant meilleure qualité qu'elle contient des éléments qui sont vrais. Nous considérons un élément modélisé dans la source comme étant vrai s'il correspond à un élément du monde réel. Si par exemple dans une source est modélisé le fait qu'une plante a des bras, cet élément ne correspond pas à un élément du monde réel ; il est donc considéré comme faux. Nous pouvons alors

définir la qualité d'une source S notée $Q(S)$ de la manière suivante [Dong et al., 2013] :

$$Q(S) = \frac{|e \in S, \text{e est vrai}|}{|S|} \quad (1)$$

Cette fonction $Q(S)$ permet de définir la qualité d'une source S. Cette qualité est le ratio du nombre d'éléments $e$ de la source $S$ qui sont vrais sur le nombre total d'éléments dans $S$. Ce ratio permet d'obtenir une valeur comprise en 0 et 1 définissant le niveau de qualité d'une source.

Cette définition permet d'avoir l'intuition de ce que nous considérons comme étant la qualité d'une source. Néanmoins, toute la difficulté de notre approche est de déterminer si $e$ est vrai ou non. Nous ne pouvons pas utiliser cette fonction telle qu'elle est définie ici. Nous demandons donc à un expert de donner une valeur représentant cette qualité pour chaque source considérée. L'expert a à sa disposition les critères présentés précédemment pour l'aider à définir cette qualité.

Le critère "adéquation de la source à la cible" est particulièrement complexe à déterminer à cause de la notion de "base de connaissances souhaitée". Il est nécessaire de définir les spécifications de ce que nous souhaitons voir apparaître dans la base de connaissances finale afin de déterminer l'adéquation entre la source et cette base.

# 3. Transformation de sources

## 3.1. Module ontologique

Un des aspects à prendre en considération lors de la conception d'une base de connaissances est la réutilisabilité de celle-ci [Gangemi and Presutti, 2009]. De façon analogue aux pratiques dans le domaine de l'ingénierie logicielle, la simplification de la réutilisation d'une base de connaissances passe par la création de sous-parties de l'ontologie répondant à un sous-ensemble des spécifications attendues de l'ontologie globale. Afin de couvrir toutes les spécifications, plusieurs sous-parties sont créées et sont liées entre-elles. De cette manière il est possible de se concentrer sur un aspect spécifique de la modélisation en ne s'intéressant qu'à une sous-partie particulière. Ce procédé permet aussi la réutilisation d'une sous partie de la base de connaissances globale (ontologie regroupant toutes les sous-parties) et non de toute la base de connaissances.

Il est possible de remarquer dans la pratique des différents acteurs du Web sémantique que la réutilisation d'ontologies de petite taille est privilégiée par rapport à l'utilisation d'ontologies plus larges mais moins facilement manipulables. C'est particulièrement le cas si nous comparons la réutilisation d'une ontologie de petite taille telle que FOAF [73] [Brickley and Miller, 2005] et la réutilisation d'une ontologie plus large telle que DOLCE [74] [Masolo et al., 2002].

De cette manière, nous souhaitons construire une ontologie générale fondée sur des modules ontologiques. Nous appelons module ontologique une sous-partie de l'ontolo-

73. Friend Of A Friend
74. a Descriptive Ontology for Linguistic and Cognitive Engineering

gie générale telle que définie précédemment. Nous nous fondons sur la définition de [Gangemi and Presutti, 2009]. Nous allons appliquer le processus global pour chaque module que nous souhaitons créer. Ces modules, une fois regroupés, permettront d'obtenir l'ontologie globale.

Chaque module représente un objectif particulier couvrant certaines spécifications de l'ontologie globale. Les différents modules permettent aussi d'avoir un objectif clairement identifié lors du processus de transformation. Pour ce qui nous concerne, notre objectif est de découvrir les connaissances communes entre plusieurs sources, et ce pour un module donné. Il faut donc avoir une définition des spécifications que nous souhaitons pour le module étudié. Nous demandons à un expert de conceptualiser ce que nous considérons comme étant la partie haute du module étudié. La partie haute d'un module englobe les classes les plus générales hiérarchiquement du module. Nous considérons qu'il n'existe pas d'autres classes au dessus hiérarchiquement que les classes de cette partie haute pour représenter le domaine qui nous intéresse. Nous représentons aussi dans cette partie haute du module ontologique toutes les propriétés que nous souhaitons utiliser.

Notre objectif est de pouvoir enrichir cette partie haute du module grâce aux éléments extraits des différentes sources. Pour cela nous extrayons uniquement les éléments qui spécialisent cette partie haute du module. En d'autres termes nous récupérons uniquement les classes qui sont, directement ou indirectement, une spécialisation d'une des classes de la partie haute du module. Nous extrayons aussi les individus qui instancient ces classes. Les relations que nous extrayons peuvent être étiquetées de quatre façons différentes :

**instanceOf :** Pour l'instanciation des individus suivant une des classes de la partie haute du module ou une spécialisation

**subClassOf :** Pour la spécialisation des classes de la partie haute du module

**rdfs :label :** Pour extraire les labels des éléments extraits

$\Sigma_{propO}$ **de la partie haute du module :** Pour représenter une relation étiquetée par une propriété définie dans la partie haute du module

De cette manière nous garantissons l'uniformité des étiquettes des relations.

L'utilisation de la partie haute du module telle que nous l'avons définie a deux avantages. Le premier est l'harmonisation de la modélisation que nous souhaitons obtenir. Effectivement chaque source ayant sa propre modélisation il n'est pas chose aisée que de fusionner ces différences. Nous l'avons vu dans l'état de l'art (c.f. section 3.1 du chapitre II) la fusion de plusieurs modèles peut apporter des incohérences et de la redondance. Pour éviter ce problème nous définissons un modèle de référence que nous utilisons pour harmoniser les différentes modélisations : la partie haute du module ontologique. Le deuxième avantage de l'utilisation de la partie haute d'un module ontologique est qu'elle permet la définition des bornes du domaine. Les éléments extraits des différentes sources doivent être en adéquation avec le module que nous cherchons à modéliser. Nous extrayons uniquement les éléments qui sont une spécialisation ou une instanciation d'une classe de la partie haute du module ontologique. De cette manière, si un élément extrait n'est pas une spécialisation ou une instanciation alors cet élément ne fait pas partie du domaine étudié.

Par conséquent, nous considérerons dans la suite de ce manuscrit, qu'un module ontologique est la partie haute de la sous-partie telle que définie dans les travaux [Gangemi and Presutti, 2009]. Afin de construire ce module, nous nous fondons, comme vu dans la section 1 du chapitre III, sur le scénario 7 de NeOn. Un module ontologique est donc le regroupement de patrons de conception adaptés et spécialisés pour le domaine étudié.

La figure 21 présente un exemple d'un module ontologique : AgronomicTaxon. Ce module ontologique [Roussey et al., 2013] permet de représenter la taxonomie des plantes. Sa conception a suivi les étapes définies dans le scénario 7 de NeOn. Il est possible de trouver plus de détails sur ce module à l'adresse suivante : `https://sites.google.com/site/agriontology/home/irstea/agronomictaxon`.

## 3.2. Transformation automatique d'une source

À ce stade du processus, nous considérons que nous avons une liste de sources sélectionnées et analysées et un module ontologique permettant de définir les classes et propriétés que nous allons chercher à spécialiser ou instancier. L'objectif est maintenant de proposer une représentation des sources considérées suivant la modélisation définie dans le module afin d'obtenir une base de connaissances source (SKB [75]).

75. Source Knowledge Base

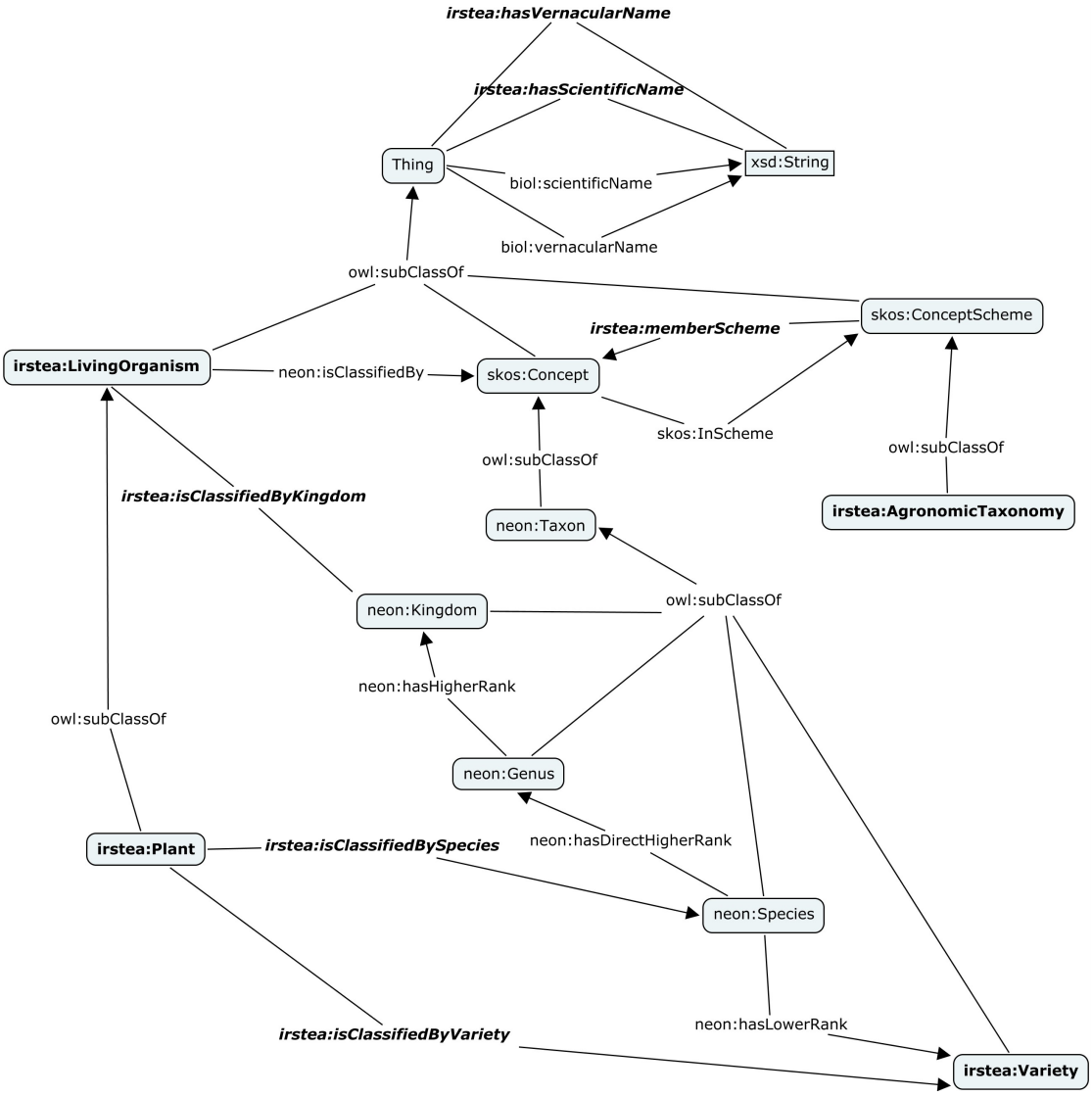


FIGURE 21 – Exemple de module ontologique : AgronomicTaxon

### 3.2.1. Base de connaissances source

Une base de connaissance source $SKB$ telle que :
$SKB = (V_{skb}, E_{skb}, \Sigma V_{skb}, \Sigma E_{skb}, etiqV_{skb}, etiqE_{skb}, sourceE_{skb}, cibleE_{skb})$
est une base de connaissance générée à partir d'une source grâce à une méthode automatique qui prend en entrée un module $KB_m$ tel que :
$KB_m = (V_m, E_m, \Sigma V_m, \Sigma E_m, etiqV_m, etiqE_m, sourceE_m, cibleE_m)$.
Le multigraphe du module est totalement inclus dans la base de connaissance source. Les composants de la SKB suivent les contraintes suivantes :

Base de connaissance source

| |
|---|
| $V_{skb}$ est l'ensemble des sommets de $SKB$. $V_m \subseteq V_{skb}$. |
| $\Sigma V_{skb}$ est l'ensemble des étiquettes des sommets de $SKB$. $\Sigma V_m \subseteq \Sigma V_{skb}$. |
| $etiqV_{skb} : V_{skb} \rightarrow \Sigma V_{skb}$ est l'application qui associe à chaque sommet son étiquette. $\forall v \in V_m, \exists l \in \Sigma V_m$ tel que $etiqV_{skb}(v) = etiqV_m(v) = l$. |
| $\Sigma E_{skb}$ est l'ensemble des étiquettes d'arcs de $SKB$. $\Sigma E_{skb} = \Sigma E_m$. Les arcs de $SKB$ ne peuvent avoir comme étiquette que les étiquettes d'arc définies dans le module. |
| $E_{skb}$ est l'ensemble des arcs de $SKB$. $E_m \subseteq E_{skb}$. |
| $etiqE_{skb} : E_{skb} \rightarrow \Sigma E_{skb}$ est l'application qui associe à chaque arc son étiquette. $\forall e \in E_m, \exists l \in \Sigma E_m$ tel que $etiqE_{skb}(e) = etiqE_m(e) = l$. |
| $sourceE_{skb} : E_{skb} \rightarrow V_{skb}$ est l'application qui associe à chaque arc son sommet initial. $\forall e \in E_m, \exists v \in V_m$ tel que $sourceE_{skb}(e) = sourceE_m(e) = v$. |
| $cibleE_{skb} : E_{skb} \rightarrow V_{skb}$ est l'application qui associe à chaque arc son sommet terminal. $\forall e \in E_m, \exists v \in V_m$ tel que $cibleE_{skb}(e) = cibleE_m(e) = v$. |

### 3.2.2. Processus de transformation

Afin d'obtenir une SKB à partir d'une source donnée, nous définissons le processus présenté dans la figure 22.

Ce processus comporte quatre étapes :

**Alignement :** Le module et la source sont alignés. De cette manière, il est possible de déterminer quels sont les éléments dans la source qui correspondent aux classes du module.

**Extraction :** En utilisant ces correspondances, tous les éléments de la source faisant partie du domaine étudié sont extraits.

**Transformation syntaxique :** Afin d'obtenir un format unique (ici RDF) une transformation syntaxique est opérée en suivant le patron de transformation sélectionné (Cf. chapitre suivant).

**Réingénierie :** Les éléments extraits et transformés sont ensuite associés au module ontologique afin de l'enrichir, en utilisant des relations étiquetées "subClassOf" ou "instanceOf". De cette manière, les éléments extraits sont représentés en exploitant la modélisation du module ontologique.

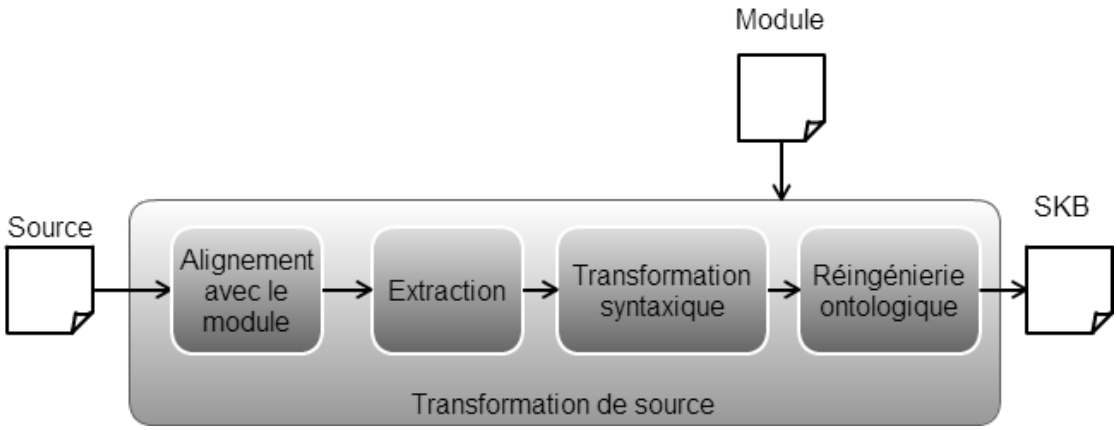


FIGURE 22 – Processus de transformation automatique d'une source

### 3.2.3. Patron de transformation

Afin de réaliser cette étape de transformation automatique des sources, nous étendons les travaux présentés dans [Villazon-Terrazas et al., 2010] qui correspondent au scénario 2 de la méthodologie NeOn présenté dans la section 1 du chapitre III. Ces travaux proposent de créer, pour chaque type de source à transformer, un patron de transformation permettant de donner les indications sur la façon de transformer la source. Comme vu dans la section 1 du chapitre II, il n'est pas réalisable de proposer une méthode de transformation générique. En effet chaque transformation de source est forcement orientée pour correspondre à un besoin. C'est pour cela que nous pensons que la réutilisation des patrons de transformation est la méthode qui correspond le mieux à nos besoins.

Afin d'appliquer cette méthodologie, nous demandons à un expert de définir un patron de transformation tel que défini dans [Villazon-Terrazas et al., 2010]. Pour cela nous suivons la méthodologie définie dans le scénario 2 présenté précédemment. La définition du patron de transformation se fait en s'appuyant sur les patrons déjà existants.

Nous étendons néanmoins ces patrons de transformation par l'implication du module ontologique dans le processus. Ce module ontologique peut être assimilable aux travaux de [Soergel et al., 2004]. Ces travaux proposent une désambiguïsation des éléments à transformer en réutilisant des règles de désambiguïsation adaptés au domaine. Nous proposons ici de représenter ces règles sous la forme d'un module ontologique modélisé par un expert et d'un patron de transformation de la méthodologie NeOn[Villazon-Terrazas et al., 2010]. Nous adaptons alors la création d'un patron de transformation en utilisant la désambiguïsation spécifique à un domaine [Soergel et al., 2004] par l'utilisation d'un module.

La création du patron de transformation se fait donc en réutilisant des patrons de transformations déjà existants et le module ontologique. Les patrons de transformations déjà existants sont génériques et non spécifique à un domaine en particulier. Nous proposons d'étendre cette définition de patrons de transformation pour les rendre spécifique à un

domaine. Pour cela nous incluons le module ontologique dans le processus qui permettra la définition du domaine à représenter.

Un patron de transformation se présente sous la forme d'un algorithme de transformation de la source. Cet algorithme de transformation est assimilable à un ensemble de règles à appliquer dans cet algorithme pour obtenir la transformation souhaitée. Par exemple, dans le patron de transformation spécifique au thésaurus (https://tinyurl.com/phss8kj) l'algorithme propose de la règle suivante : tous les regroupements de termes sont transformés en classe de l'ontologie. L'utilisation du module dans la définition de ces règles apporte la nécessité de définir des règles plus spécifiques. Par exemple que tous les regroupements de termes du thésaurus dans une certaine branche du thésaurus sont transformés en classe avec la relation "subClassOf" d'une classe du module.

De cette manière, il est possible d'obtenir un processus automatique de transformation de la source en suivant les règles définies. Les alignements permettent d'orienter la modélisation attendue et d'extraire de la source uniquement ce qui est considéré dans le domaine du module défini.

Si nous prenons l'exemple du thésaurus Agrovoc, il est possible de définir un ensemble de règles pour obtenir la SKB présentée sur la figure 23.

Dans cet exemple, un alignement a été effectué entre la source Agrovoc et le module AgronomicTaxon. Les correspondances sont représentées par des doubles flèches pointillées sur ce schéma. Pour obtenir le résultat présenté dans cet exemple nous avons défini les règles suivantes, en nous inspirant du patron de transformation sur les thésaurus [76] :

- Nous récupérons tous les "skos :Concept" qui ont la relation (directement ou indirectement [77]) "skos :broader" vers "agrovoc :Triticum" (nous récupérons tous les taxons de rang inférieur à Triticum dans la taxonomie)
- Les relations "agrovoc :hasTaxonomicLevel" dont le sujet est un taxon extrait précédemment deviennent des relations "instanceOf" vers la classe correspondante dans le module (nous créons une nouvelle classe s'il n'existe pas de classe correspondante dans le module)
- Les relations "skos :broader" entre deux taxons extraits avec la première règle deviennent des relations "agronomicTaxon :hasHigherRank" entre ces deux taxons

En utilisant ces règles, nous obtenons la SKB présentée en bas de la figure 23. Chaque taxon est donc une instanciation de la classe correspondant au concept associé par la relation "hasTaxonomicLevel" dans Agrovoc. S'il n'y a pas de classe correspondante, alors nous créons une nouvelle classe. Nous utilisons la propriété "hasHigherRank" pour représenter la hiérarchie entre les taxons existant dans Agrovoc. Afin de respecter le domaine souhaité nous posons la contrainte de ne récupérer que les taxons qui sont reliés (directement ou indirectement) par la relation "skos :broader" à "Triticum". Si nous changeons cette limite au taxon "Plantae", qui correspond au règne de toutes les plantes, il est alors possible d'extraire toute la taxonomie des plantes. Nous remarquons que "Karnal Bunt" n'est pas relié à "Triticum" par une ou plusieurs relations "skos :broader",

76. https://tinyurl.com/phss8kj
77. par une chaîne de relation "skos :borader"

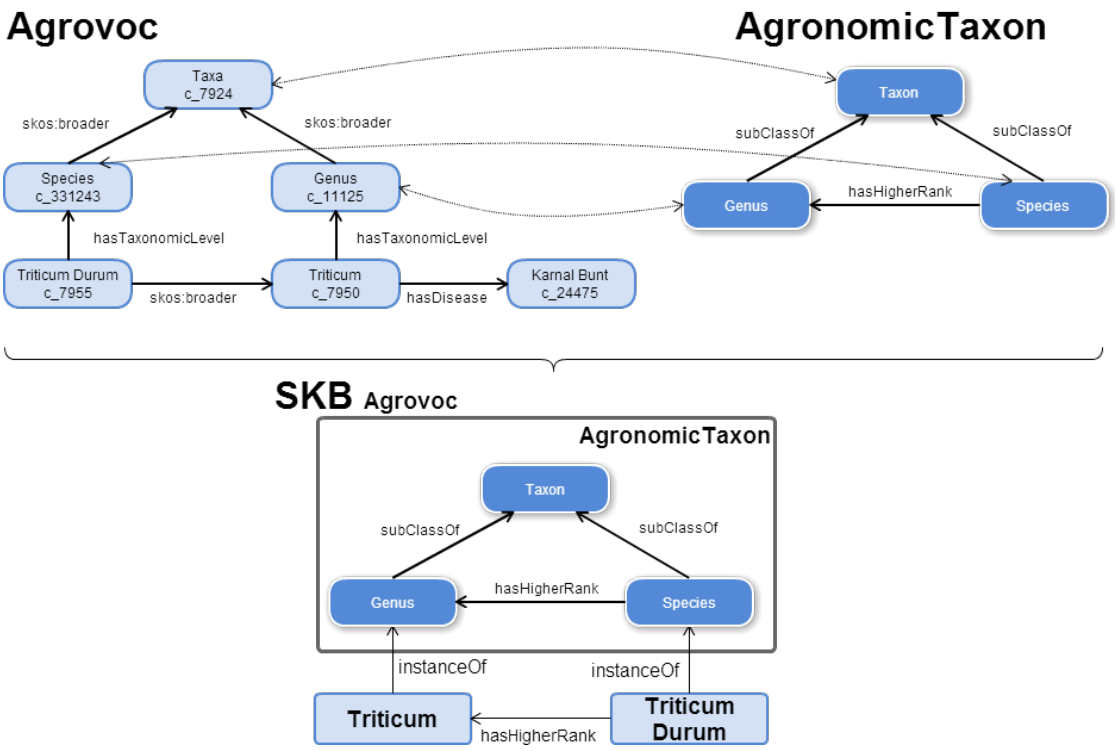


FIGURE 23 – Exemple de transformation de la source Agrovoc par une méthode automatique (les double flêches pointillées représentent des correspondances entre la source et le module)

ni par une propriété considérée dans ces règles ; par conséquent, il n'apparaît pas dans la SKB.

Dans cet exemple il n'y a pas de création de nouvelle classe. Néanmoins si nous avions considéré toute la classification des plantes dans Agrovoc, il existe des taxons catégorisés comme "SubSpecies". Ce rang n'étant pas défini dans le module, nous avons ajouté une nouvelle classe extraite d'Agrovoc.

Nous posons la contrainte de ne pas extraire de nouvelles propriétés car nous considérons que toutes les propriétés nécessaires à la définition du domaine sont présentes dans le module. Les règles de transformations mettent en évidence les transformations à effectuer pour obtenir l'équivalent avec les propriétés du module (par exemple les relations "agrovoc :hasTaxonomicLevel" deviennent des relations "instanceOf"). La relation "hasDisease" n'est pas représentée dans la SKB qui ne fait pas partie des règles de transformation car il n'y a pas d'équivalent dans le module. Cette relation est donc ignorée car considérée comme ne faisant pas partie des spécifications attendues pour le module.

La concrétisation de ces règles de transformation permet d'aboutir à un algorithme de transformation de la source en une SKB en fonction du module étudié. Cet algorithme

est implémenté et appliqué afin d'effectuer la transformation. Il contient les différentes étapes de la phase de transformation de la source (extraction, transformation syntaxique et réingénierie ontologique) en fonction des alignements avec le module.

Il est possible de définir un algorithme générique en fonction des règles définies plus haut. Cet algorithme générique est présenté dans l'algorithme 1. Nous avons considéré que la variable "classMappings" contient les correspondances entre classes du module et la source (par exemple ici le concept SKOS "agrovoc :Genus" correspond à la classe "agronomicTaxon :Genus" du module) et la variable "objPropMappings" contient les correspondances entre propriétés (par exemple ici les relations "skos :broader" deviennent des relations "agronomicTaxon :hasHigherRank"). Les fonctions "addClass" et "addIndividual" permettent d'ajouter respectivement une nouvelle classe et un nouvel individu. Ces fonctions ajoutent aussi automatiquement les labels présents dans la source et la relation de subsomption pour "addClass" et d'instanciation pour "addIndividual".

Nous considérons dans l'algorithme 1 que la source est au format RDF. Néanmoins il est tout à fait possible de réutiliser cet algorithme pour une base de données, une taxonomie (SKOS ou autre) ou toute autre source. Comme c'est par exemple le cas dans notre exemple de transformation de lu thésaurus SKOS Agrovoc présenté sur la figure 23. Pour ce faire, il est nécessaire d'effectuer en premier lieu la transformation syntaxique. La fonction "getAllSubClasses" a la particularité de récupérer toutes les sous-classes de "myClass" passé en paramètre, qu'elles soient sous-classes directes ou indirectes. C'est pour cela que nous récupérons la super-classe de la source. Cet algorithme se déroule en deux étapes. La première (de la ligne 7 à 25) permet de créer les nouvelles classes et tous les individus associés. La deuxième étape (de la ligne 26 à 41) récupère toutes les relations entre les individus déjà extraits. Il est nécessaire de procéder en deux étapes car il faut pouvoir connaître tous les individus présents dans la SKB avant de pouvoir en extraire les relations entre eux.

L'expert définit de cette manière un patron de transformation pour chaque source en s'adaptant, comme présenté précédemment, au module ontologique utilisé. De cette manière, une base de connaissances source (SKB) est générée pour chaque source considérée. Ces SKB représentent les sources transformées suivant l'objectif identifié par le module. Cette transformation étant automatique, il se peut que des erreurs apparaissent dans le résultat des SKB comme, par exemple, une erreur initialement présente dans la source, qui apparaîtra alors dans la SKB, ou encore un aspect spécifique de la modélisation de la source qui n'aurait pas été pris en compte dans le patron de transformation. Si nous reprenons l'exemple présenté sur la figure 23, il est possible que dans une branche spécifique de la classification taxonomique des plantes provenant de la source Agrovoc, la relation "skos :broader" ne soit pas nécessairement assimilable à la relation "hasHigherRank", comme défini dans le patron de transformation. De cette manière, une erreur va apparaître dans la SKB. L'hypothèse de base de nos travaux selon laquelle un élément extrait d'une source est d'autant plus digne de confiance qu'il apparaît dans plusieurs sources nous permet ici de mettre en évidence ces erreurs. En effet, ces erreurs apparaissant dans les SKB ne seront que peu partagées avec les autres sources.

```
classMappings = ensemble de correspondances entre les classes du module
                et la source
objPropMappings = ensemble de correspondances entre les propriétés du
                  module et la source
SKB = {Module}
Source = La source considérée
for(myClass in classMappings)
{
    for(ind in Source.getIndividuals(myClass))
    {
        SKB.addIndividual(ind, myClass);
    }
    for(subClass in Source.getAllSubClasses(myClass))
    {
        if(!classMappings.contains(subClass))
        {
            superClass = Source.getSuperClassOf(subCLass);
            SKB.addClass(subClass, superClass);
        }
        for(ind in Source.getIndividual(subClass))
        {
            SKB.addIndividual(ind, subCLass);
        }
    }
}
for(ind1 in SKB.getIndividuals())
{
    for(ind2 in SKB.getIndividuals())
    {
        if(ind1 != ind2)
        {
            for((propSource, propModule) in objPropMappings)
            {
                if(Source.containsRel(ind1, propSource, ind2))
                {
                    SKB.addRel(ind1, propModule, ind2);
                }
            }
        }
    }
}
```

**Algorithm 1:** Algorithme générique de patron de transformation

# 4. Fusion de bases de connaissances

Comme présenté dans le processus général, une fois les bases de connaissances sources obtenues, il faut les fusionner pour obtenir la base de connaissances finale.

## 4.1. Processus de fusion de bases de connaissances

Cette phase de fusion des SKB est décomposée en plusieurs étapes présentées dans la figure 24. En entrée de ce processus sont présentes les différentes SKB que nous considérons à fusionner. Nous souhaitons obtenir en sortie la base de connaissances finale qui est le résultat de la fusion des différentes SKB. Pour cela, quatre étapes sont présentes dans ce processus :

**Alignement des bases de connaissances sources :** Cette étape permet de créer des correspondances entre tous les couples de SKB considérée.

**Génération de candidats :** À partir des correspondances présentes entre les SKB, des candidats d'éléments ontologiques ou de relations sont générés. Ces candidats sont des éléments potentiels qui pourront faire partie de la base de connaissances finale.

**Calcul de la confiance :** Un score de confiance est associé à chaque candidat représentant sa confiance consensuelle. Ce score est d'autant plus élevé que l'élément a été trouvé dans plusieurs sources.

**Découverte de l'extension optimale :** Nous considérons une notion d'incompatibilité entre candidats lorsqu'ils partagent des éléments ontologiques communs. À partir de ces incompatibilités, nous définissons une extension comme un sous-ensemble de candidats non-incompatibles. Cette étape permet la découverte de l'extension optimale, qui est l'extension maximisant la confiance des candidats.

## 4.2. Alignements des bases de connaissances sources

La première étape du processus de fusion est l'alignement entre les différentes SKB. Conformément au fonctionnement des outils d'alignements, nous effectuons cet alignement

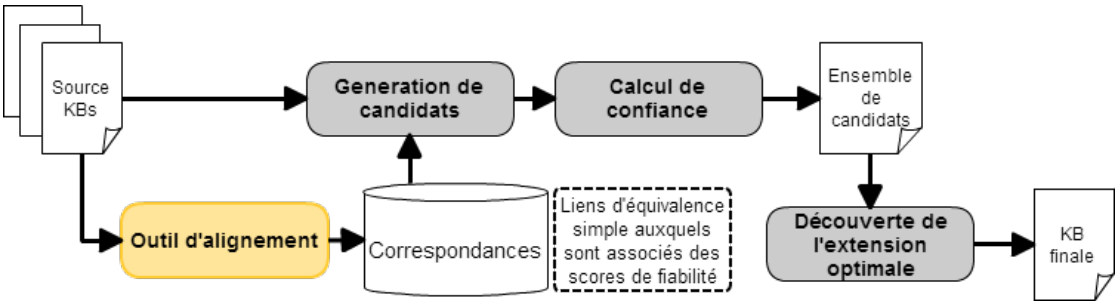


FIGURE 24 – Processus de fusion des bases de connaissances

entre chaque paire de SKB. Pour chaque paire, nous obtenons un alignement qui est un ensemble de correspondances. Une correspondance est une relation d'équivalence entre deux éléments appartenant respectivement à chacune des deux sources alignées. L'objectif est de détecter les éléments communs entre les différentes sources.

### 4.2.1. Correspondance

Un élément ontologique *oe* est un des sommets d'une base de connaissances. En d'autres termes, un élément ontologique est une ressource identifiée par une URI. Par exemple un individu est un élément ontologique mais une relation n'en est pas une. En effet, une relation est étiquetée par l'URI d'une propriété et relie deux ressources elles aussi identifiées par des URI, mais la relation elle-même n'est pas identifiée par une URI.

Elément ontologique

| |
|---|
| $oe \in C \cup I$ |

On définit aussi une fonction $nature(oe)$ qui permet d'obtenir le type de l'élément ontologique associé :

— $nature(oe)$ = "classe" si $oe \in C$,
— $nature(oe)$ = "individu" si $oe \in I$,
— $nature(oe)$ = "null" sinon.

Nous ne considérons pas les sommets propriétés : subClassOf, subPropertyOf, domain, range, instanceOf comme pouvant être des éléments ontologiques. Ces propriétés sont déjà définies dans le vocabulaire RDFS, et nous ne travaillerons que sur les arcs étiquetés avec ces propriétés. Nous ne cherchons pas non plus à définir de nouvelles propriétés d'objets ($PropO$) ou de propriétés de type de données ($PropDT$) car nous considérons que les propriétés nécessaires seront déjà définies, comme nous le verrons dans la section 3.1 du chapitre III. Nous nous limiterons, là aussi, à ces seules propriétés. Les seuls éléments ontologiques que nous considérons ici sont de nature "classe" ou "individu". Nous cherchons, dans un premier temps, à extraire uniquement ce type d'éléments ontologiques pour les ajouter dans la base de connaissances finale.

Soient deux bases de connaissances sources $SKB_i$ et $SKB_j$ construites grâce au processus de transformation de sources présenté précédemment, nous définissons une correspondance $m$ comme étant une arête entre une paire de sommets $\{oe_i, oe_j\}$, avec $oe_i \in V_{skbi}, oe_j \in V_{skbj}$. Une correspondance représente une équivalence entre deux éléments ontologiques, détectée à l'aide d'outils d'alignement. Les correspondances sont définies de la manière suivante :

Correspondance

— $V_{skbi} \neq V_{skbj}$ une correspondance est toujours établie entre deux sommets appartenant à des ensembles de sommets de SKB différentes. $\nexists m = \{oe_i, oe_k\}$ tel que $oe_i \in V_{skbi}$ et $oe_k \in V_{skbi}$.
— $nature(oe_i) = nature(oe_j) \neq$"null". Une correspondance est toujours établie entre deux éléments ontologiques de même nature.
— $valueE : (C \cup I) \times (C \cup I) \rightarrow ]0, 1]$ est une application qui, à toute arête définie comme correspondance, associe un unique degré de fiabilité compris entre 0 et 1 tel que $valueE(oe_i, oe_j) = valueE(oe_j, oe_i)$.

Dans nos travaux, nous utilisons l'outil d'alignement LogMap [78] car ce système a obtenu de bons résultats lors de l'évaluation OAEI 2014 [Dragisic et al., 2014] et que, de plus, il permet de mettre en correspondance des individus et pas seulement des classes. [Jiménez-Ruiz and Grau, 2011]

Dans les travaux de [Euzenat and Shvaiko, 2007], une correspondance est définie par un quadruplet. En plus des deux éléments ontologiques, une correspondance est composée d'un identifiant et d'un type de correspondance. En effet, dans ces travaux, une correspondance peut être une spécialisation, une généralisation, une équivalence ou une composition. Nous considérons pour notre part uniquement l'équivalence, car les systèmes d'alignement ne proposent, pour l'instant, que ce type de correspondance dans leurs résultats. De par le fait que nous ne considérons qu'un type de correspondance, nous avons donc l'unicité d'une paire d'éléments ontologiques pour un alignement donné. L'identifiant de la correspondance n'est donc pas nécessaire.

Les systèmes d'alignement permettent d'obtenir des correspondances de type $1 : n$, c'est à dire qu'un élément d'une source peut être jugé équivalent à n éléments de la deuxième source. Une relation d'équivalence $1 : n$ entre un élément de la source A et $n$ éléments de la source B peut être interprétée comme :

1. L'élément de la source A est en relation avec une agrégation des $n$ éléments de la source B. Cette relation peut être de la méronymie, subsomption ou une relation spécifique au domaine. Ce type de relation est associé à de l'alignement complexe, qui n'est pas encore traité par les systèmes d'alignements actuels [Dragisic et al., 2014]. Il n'existe pas de désambiguïsation de la relation existante dans les sorties des systèmes d'alignement.
2. L'élément de la source A est en relation d'équivalence avec un des $n$ éléments de la source B, mais l'outil d'alignement n'a pas pu faire un choix définitif et a proposé un ensemble d'éléments de B possibles.

Pour traiter ces correspondances $1 : n$, nous avons fait l'hypothèse de ne considérer que la deuxième interprétation des correspondances $1 : n$ ; il nous faut donc déterminer, parmi les $n$ éléments de la source B, lequel est vraiment en relation d'équivalence avec l'élément de la source A.

78. http://www.cs.ox.ac.uk/isg/projects/LogMap/

### 4.2.2. Alignement

Un alignement $A(SKB_i, SKB_j) = (V_{aij}, E_{aij}, \Sigma V_{aij}, etiqV_{aij}, valueE_{aij})$ entre deux bases de connaissances distinctes
$SKB_i = (V_{skbi}, E_{skbi}, \Sigma V_{skbi}, \Sigma E_{skbi}, etiqV_{skbi}, etiqE_{skbi}, sourceE_{skbi}, cibleE_{skbi})$ et
$SKB_j = (V_{skbj}, E_{skbj}, \Sigma V_{skbj}, \Sigma E_{skbj}, etiqV_{skbj}, etiqE_{skbj}, sourceE_{skbj}, cibleE_{skbj})$
est un graphe biparti étiqueté non-orienté ayant pour arêtes uniquement des correspondances entre $SKB_i$ et $SKB_j$. Nous définissons les composants d'un alignement $A(SKB_i, SKB_j)$ de la manière suivante :

Alignement

$V_{aij} = U \cup W$. Les sommets de $A(SKB_i, SKB_j)$ composent deux ensembles de sommets disjoints $U \subset (C_{skbi} \cup I_{skbi})$ et $W \subset (C_{skbj} \cup I_{skbj})$ tels que chaque arête relie des sommets de $U$ à des sommets de $W$.
$\Sigma V_{aij} \subset \Sigma V_{skbi} \cup \Sigma V_{skbi}$. L'ensembles des étiquettes des sommets est inclus dans l'union des ensembles d'étiquettes des sommets de $SKB_i$ et de $SKB_j$.
$etiqV_{aij} : V_{aij} \rightarrow \Sigma V_{aij}$ est la fonction d'étiquetage des sommets telle que :
— $\forall v \in V_{skbi} \cap V_{aij}$ alors $etiqV_{aij}(v) = etiqV_{skbi}(v)$.
— $\forall v \in V_{skbj} \cap V_{aij}$ alors $etiqV_{aij}(v) = etiqV_{skbj}(v)$.
$E_{aij} \subset V_{skbi} \times V_{skbj}$ est l'ensemble des arêtes représentant des correspondances.
$\forall m_k \in E_{aij}$ avec $m_k = \{oe_i, oej\}$ et $oe_i \in V_{skbi}, oe_j \in V_{skbj}$ alors $\nexists m_l \in (E_{aij} - \{m_k\})$ tel que $m_l = m_k = \{oe_i, oej\}$. Il n'existe qu'une seule correspondance impliquant les mêmes sommets de $V_{skbi}$ et de $V_{skbj}$
$valueE_{aij} : V_{skbi} \times V_{skbj} \rightarrow ]0, 1]$ est la fonction d'étiquetage des arêtes précédemment définie, qui associe à chaque correspondance son degré de fiabilité.

### 4.2.3. Bases de connaissances sources alignées

Soit $SKB_i = (V_{skbi}, E_{skbi}, \Sigma V_s, \Sigma E_{skbi}, etiqV_{skbi}, etiqE_{skbi}, sourceE_{skbi}, cibleE_{skbi})$ un exemple de SKB. Soit $A(SKB_k, SKB_j) = (V_{akj}, E_{akj}, \Sigma V_{akj}, etiqV_{akj}, valueE_{akj})$ un exemple d'alignement.

Nous considérons que les SKB ont toutes été générées à partir du même module $KB_m$. $KB_m$ étant l'ontologie représentant le module ontologique selectionné. Sa définition est donc identique à celle d'une base de connaissances donnée dans la section 1.2.4 du chapitre I :
$KB_m = (V_{kb_m}, E_{kb_m}, \Sigma V_{kb_m}, \Sigma E_{kb_m}, etiqV_{kb_m}, etiqE_{kb_m}, sourceE_{kb_m}, cibleE_{kb_m})$

Dans la suite de ce mémoire, $N$ sera le nombre de SKB prises en considération et $T$ le nombre d'alignements entre ces $N$ SKB. Notons que nous ne pouvons avoir que $\frac{N(N-1)}{2}$ alignements possibles entre $N$ SKB. Il n'existe pas forcément un alignement entre deux SKB données. Par conséquent, $T \leq \frac{N(N-1)}{2}$. Nous supposons que les SKB sont ordonnées de 1 à $N$.

Nous définissons par $SA = (V_{sa}, E_{sa}, \Sigma V_{sa}, \Sigma E_{sa}, etiqV_{sa}, etiqE_{sa}, sourceE_{sa}, cibleE_{sa})$ le multigraphe résultat de l'union des multigraphes des N SKB et des graphes des

alignements entre ces N SKB.

$SA$ est défini de la manière suivante :

Bases de connaissances sources alignées

$V_{sa} = \bigcup_{i=1}^{N} V_{skbi}$. L'ensemble des sommets du multigraphe $SA$ est l'union des ensembles de sommets des $N$ SKB.
$\Sigma V_{sa} = \bigcup_{i=1}^{N} \Sigma V_{skbi}$. L'ensemble des étiquettes de sommet de $SA$ est l'union des ensembles d'étiquettes des sommets des $N$ SKB.
$etiqV_{sa} : V_{sa} \rightarrow \Sigma V_{sa}$ est le prolongement des applications $etiqV_{skbi} : V_{skbi} \rightarrow \Sigma V_{skbi}$ qui associe à tout sommet de $V_{sa}$ une étiquette de $\Sigma V_{sa}$ tel que $\forall v \in V_{skbi}$, $etiqV_{sa}(v) = etiqV_{skbi}(v)$.
$\Sigma E_{sa} = \bigcup_{i=1}^{N} \Sigma E_{skbi} \cup ]0..1] = \Sigma E_m \cup ]0..1]$. L'ensemble des étiquettes d'arcs de $SA$ est l'union des ensembles d'étiquettes d'arcs des SKB et des degrés de fiabilité des correspondances issues des alignements entre les SKB. Rappelons que, dans une SKB, les étiquettes d'arcs sont limitées aux étiquettes d'arcs du module utilisé pour la transformation de la source, comme nous l'avons vu dans la section 3.2.3.
$E_{sa} = \bigcup_{i=1}^{N} E_{skbi} \cup (\bigcup_{i=1}^{N-1} \bigcup_{j=i+1}^{N} E_{aij})$. L'ensemble des arcs de $SA$ est l'union disjointe de $N$ ensembles d'arcs des SKB et des $T$ ensembles d'arêtes des alignements.
$etiqE_{sa} : E_{sa} \rightarrow \Sigma E_{sa}$ est une application qui associe à chaque arc son étiquette et à chaque arête son degré de fiabilité. $etiqE_{sa}$ est le prolongement des applications :

- $etiqE_{skbi} : E_{skbi} \rightarrow \Sigma E_{skbi}$ qui associe à chaque arc de $SKB_i$ son étiquette de $\Sigma V_{skbi}$ tel que $\forall e \in E_{skbi}$ $etiqE_{sa}(e) = etiqE_{skbi}(e)$
- $valueE_{akj} : E_{akj} \rightarrow ]0..1]$ qui associe à chaque arête de $A(SKB_k, SKB_j)$ son degré de fiabilité tel que $\forall e \in E_{akj}$ $etiqE_{sa}(e) = valueE_{akj}(e)$

$sourceE_{sa} : E_{sa} \rightarrow V_{sa}$ est une application qui associe à chaque arc d'une SKB son sommet initial et à chaque arête d'un alignement le sommet qui appartient à la SKB qui a le plus petit numéro d'ordre. $sourceE_{sa}$ est le prolongement des applications $sourceE_{skbi}$. Ainsi $\forall e \in E_{skbi}$, $\exists v \in V_{skbi}$ tel que $sourceE_{sa}(e) = sourceE_{skbi}(e) = v$.
$cibleE_{sa} : E_{sa} \rightarrow V_{sa}$ est une application qui associe à chaque arc d'une SKB son sommet terminal et à chaque arête d'un alignement le sommet qui appartient à la SKB qui a le plus grand numéro d'ordre. $cibleE_{sa}$ est le prolongement des applications $cibleE_{skbi}$. Ainsi $\forall e \in E_{skbi}$, $\exists v \in V_{skbi}$ tel que $cibleE_{sa}(e) = cibleE_{skbi}(e) = v$.

La figure 25 présente un exemple d'un graphe SA d'après l'alignement des sources NCBI, Agrovoc et TaxRef. Sur cette image, les correspondances sont représentées par des pointillés et étiquetées par leur degré de fiabilité.

Pour générer ce graphe, nous faisons un parcours sur toutes les sources considérées et pour chaque paire de sources, nous faisons appel à un système d'alignement pour générer les correspondances.

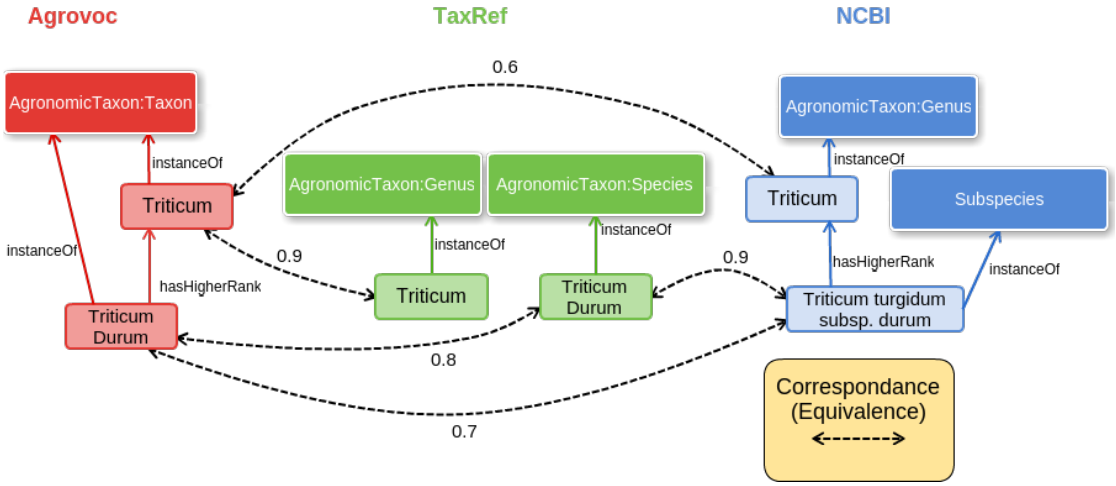


FIGURE 25 – Bases de connaissances sources alignées

## 4.3. Génération de candidats

Un candidat $Cand$ représente un élément possible de la base de connaissance finale. Cet élément est le résultat de la fusion d'éléments de plusieurs SKB considérés comme équivalents. Nous considérons deux types de $Cand$, les candidats sommets $Cand_s$, qui correspondent à un sommet de la KB finale, et les candidats arcs $Cand_a$ qui correspondent à un arc dans la KB finale. Ces arcs représentent des relations pouvant exister entre les candidats sommets extraits ou entre un candidat sommet et un élément du module. Nous ne cherchons pas à découvrir de nouvelles propriétés d'objets ou de types de données, ni à découvrir de nouveaux labels sur les classes du module. Par conséquent, nous pouvons affirmer que chaque candidat arc est associé à au moins un candidat sommet.

### 4.3.1. Candidat sommet

Un candidat sommet $CandS = (V_{candS}, E_{candS}, \Sigma V_{candS}, etiqV_{candS}, valueE_{candS})$ est un graphe non-orienté connexe dont les sommets sont des éléments ontologiques provenant de SKB différentes et les arêtes sont les correspondances issues des alignements entre les SKB. Un candidat sommet est un sous-graphe du multigraphe $SA$ construit à partir des $N$ bases de connaissances sources alignées. Nous définissons les composants d'un candidat sommet de la manière suivante :

Candidat sommet

$V_{candS} \subset V_{sa}$. $\forall v \in V_{candS}$ avec $v \in V_{skbi}$ $\nexists v' \in V_{candS}$ tel que $v' \in V_{skbi}$ et $v \neq v'$. Tous les sommets d'un candidat sommet appartiennent à des SKB différentes. Par conséquent $|V_{Cand_s}| \leq N$.
$E_{CandS} \subset \bigcup_{i=1}^{N-1} \bigcup_{j=i+1}^{N} E_{aij}$. L'ensemble des arêtes d'un candidat sommet est inclus dans l'ensemble des arêtes des $T$ alignements. Les arêtes de $CandS$ sont des correspondances. Un candidat sommet est un graphe connexe. $\forall v_1, v_2 \in V_{candS}$, il existe forcément un chemin $path = \{e_j, ..., e_k\}$ avec $\forall e_i, e_i \in path, e_i \in E_{CandS}$ reliant $v_1$ à $v_2$. Par conséquent, tous les sommets de $CandS$ sont liés à au moins un autre sommet de $CandS$ par une correspondance, ce qui implique que tous les sommets de $CandS$ sont de même nature $\forall v_1, v_2 \in V_{candS}$ $nature(v_1) = nature(v_2)$.

D'après la définition des éléments ontologiques, un candidat sommet ne peut être que de type individu ou classe.

Nous définissons la fonction $etiqCandS(CandS)$ qui permet d'obtenir l'étiquette d'un candidat sommet. Cette étiquette prend la forme d'une URI référençant le candidat sommet en question. Cette fonction sera utilisée plus tard pour la définition des incompatibilités entre candidats et pour la génération d'extensions (Cf. section 4.5 du chapitre III).

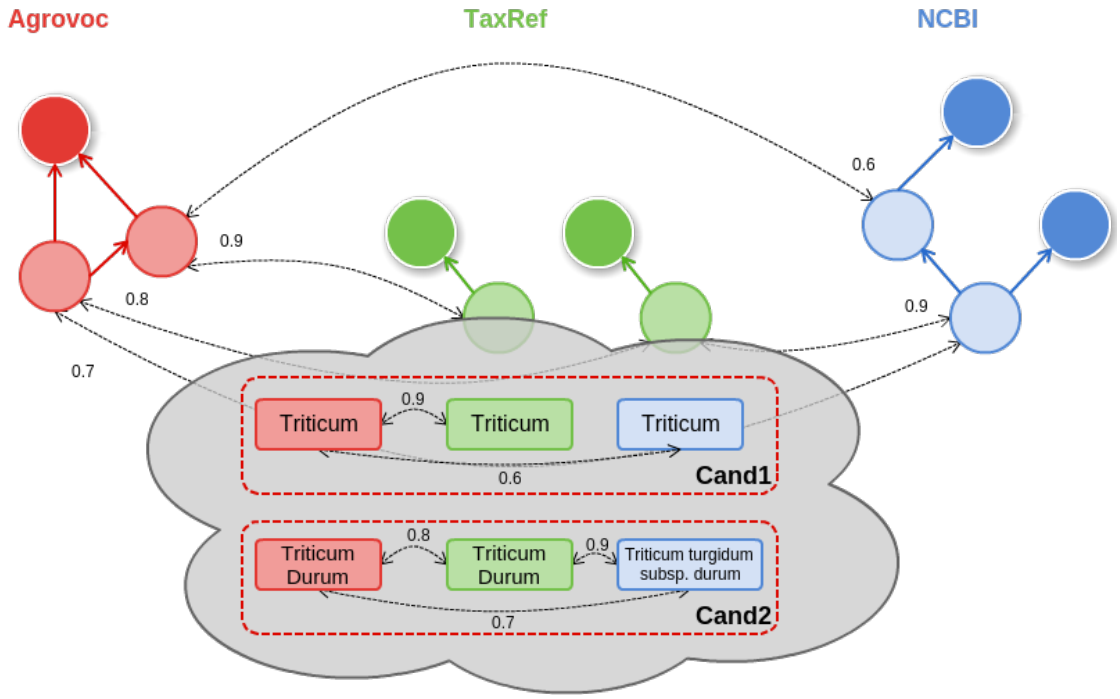


FIGURE 26 – Exemple de deux candidats extraits de trois sources

Dans la figure 26, nous pouvons voir la génération de deux candidats sommets dont les éléments ontologiques sont de nature "individu". Les deux candidats sommets représentent donc des éléments ontologiques potentiels dans la base de connaissances finale, ici "Triticum" et "Triticum Durum".

### 4.3.2. Candidat arc

Un candidat arc $CandA = (V_{CandA}, E_{CandA}, \Sigma V_{CandA}, \Sigma E_{CandA}, etiqV_{CandA}, sourceE_{CandA}, cibleE_{CandA})$ est un graphe biparti orienté étiqueté. Un candidat arc se construit à partir d'au moins un candidat sommet. Les sous-ensembles de sommets de $CandA$ sont soit un ensemble de sommets d'un candidat sommet, soit un singleton composé d'un sommet du module $KB_m$. Tous les arcs de $CandA$ sont étiquetés par la même étiquette d'arc. Les candidats arcs sont des sous-graphes du multigraphe $SA$ construit à partir des $N$ bases de connaissances alignées.

Considérons deux candidats sommets distincts :
$CandS_1 = (V_{CandS1}, E_{CandS1}, \Sigma V_{CandS1}, etiqV_{CandS1}, valueE)$,
$CandS_2 = (V_{CandS2}, E_{CandS2}, \Sigma V_{CandS2}, etiqV_{CandS2}, valueE)$

Candidat arc entre 2 candidats sommets

| |
|---|
| $V_{CandA} = V_{CandS1} \cup V_{CandS2}$. L'ensemble des sommets de $CandA$ se compose de deux ensembles de sommets disjoints $V_{CandS1}$ et $V_{CandS2}$. |
| $\Sigma V_{CandA} = \Sigma V_{CandS1} \cup \Sigma V_{CandS2}$. L'ensemble des étiquettes de sommets de $CandA$ est l'union des ensembles d'étiquettes de sommets $V_{CandS1}$ et $V_{CandS2}$. |
| $etiqV_{CandA} : V_{CandS_1} \cup V_{CandS_2} \rightarrow \Sigma V_{CandS_1} \cup \Sigma V_{CandS2}$ est l'appplication qui associe à tout sommet $v \in V_{CandA}$ son étiquette $l \in \Sigma V_{CandA}$ tel que $etiqV_{CandA}(v) = l$. Cette application est le prolongement des applications $etiqV_{CandS_1}$ et $etiqV_{CandS_2}$. |
| $\Sigma E_{CandA} = \{l_{CandA}\}$ l'ensemble des étiquettes d'arcs est composé d'une seule étiquette $l_{CandA}$ tel que $l_{CandA}$ est défini dans le module $KB_m$. |
| $E_{CandA} \subset \Sigma E_{CandA} \times V_{CandS_1} \times V_{CandS_2}$ est l'ensemble des arcs de $CandA$. Ces arcs sont tous étiquetés par la même étiquette $l_{CandA}$ et tous orientés dans le même sens. Les arcs de $CandA$ vont tous d'un sommet de $CandS_1$ vers un sommet de $CandS_2$. |
| $sourceE_{CandA} : E_{CandA} \rightarrow V_{CandS1}$ est l'application qui à tout arc de $CandA$ associe son sommet initial pris dans $V_{CandS1}$ |
| $cibleE_{CandA} : E_{CandA} \rightarrow V_{CandS2}$ est l'application qui à tout arc de $CandA$ associe son sommet terminal pris dans $V_{CandS2}$ |

Considérons un candidat sommet
$CandS = (V_{CandS}, E_{CandS}, \Sigma V_{CandS}, etiqV_{CandS}, valueE)$,
et un module $KB_m = (V_m, E_m, \Sigma V_m, \Sigma E_m, etiqV_m, etiqE_m, sourceE_m, cibleE_m)$.

Candidat arc entre 1 candidat sommet et 1 sommet du module

| |
|---|
| $V_{CandA} = V_{CandS} \cup \{v_m\}$. L'ensemble des sommets de $CandA$ se compose de deux ensembles de sommets disjoints $V_{CandS}$ et $\{v_m\}$ tels que $v_m \in V_m$ .<br>$\Sigma V_{CandA} \subset \Sigma V_{candS_1} \cup \{etiqV_m(v_m)\}$. L'ensemble des étiquettes de sommets de $CandA$ est l'ensemble d'étiquettes de sommets de $CandS$ augmenté de l'étiquette du sommet du module $v_m$.<br>$etiqV_{CandA} : V_{CandA} \rightarrow \Sigma V_{CandA}$ est l'application qui associe à tout sommet $v \in V_{CandA}$ son étiquette $l \in \Sigma V_{candA}$ tel que $etiqV_{CandA}(v) = l$. Cette application est le prolongement des applications $etiqV_{CandS}$ et $etiqV_m$.<br>$\Sigma E_{CandA} = \{l_{CandA}\}$ l'ensemble des étiquettes d'arcs est composé d'une seule étiquette appartenant au module $l_{CandA}$ tel que $l_{CandA} \in \Sigma E_m$.<br>$E_{CandA} \subset \Sigma E_{CandA} \times V_{CandA} \times V_{CandA}$ est l'ensemble des arcs de $CandA$. Ces arcs sont tous étiquetés par la même étiquette $l_{CandA}$ et tous orientés dans le même sens : soit les arcs de $CandA$ vont d'un sommet de $CandS$ vers $v_m$, soit les arcs de $CandA$ vont de $v_m$ vers un sommet de $CandS$.<br>$sourceE_{CandA} : E_{CandA} \rightarrow V_{CandA}$ est l'application qui à tout arc de $CandA$ associe son sommet initial.<br>$cibleE_{CandA} : E_{CandA} \rightarrow V_{CandA}$ est l'application qui à tout arc de $CandA$ associe son sommet terminal. |

Dans l'exemple présenté dans la figure 27 sont présents trois candidats arcs. Le premier "Cand arc 1" associe le candidat sommet "Cand1" avec une classe du module ("AgronomicTaxon :Genus"). Cet arc représente le fait que le candidat "Cand1" est de type "Genus". Le deuxième candidat arc ("Cand arc2") est entre les deux candidats sommets Cand1 et Cand2 (présentés précédemment) ; il est étiqueté par "hasHigherRank". La relation "hasHigherRank" associant Cand1 et Cand2 est présente dans les deux sources Agrovoc et NCBI. Le dernier candidat arc "Cand arc3" associe un candidat sommet Cand2 à un candidat label. D'après la définition d'un candidat sommet présentée précédemment, un candidat label est un candidat sommet. Néanmoins, nous considérons qu'un candidat sommet label ne peut exister que s'il est un composant d'un candidat arc ayant comme propriété rdfs :label. En effet, il n'est pas possible de représenter un label dans une base de connaissances sans l'associer à un sommet de cette base.

D'après la figure 25, nous pouvons déduire d'autres candidats n'apparaissant pas sur la figure 27 [79]. Un candidat sommet de type classe est créé, du fait de la présence de la classe "Subspecies" dans NCBI. Cette classe n'est en correspondance avec aucune autre classe d'une autre source et n'apparaît pas dans le module. Cette classe permet de créer un candidat sommet ne contenant qu'un seul élément ontologique : la classe "Subspecies" de NCBI. De plus un candidat arc est généré entre le candidat "Cand2" (Triticum Durum) et ce candidat classe "Subspecies" avec l'étiquette "instanceOf". Là encore, ce candidat arc n'apparaît que dans une source NCBI.

79. non représenté pour des raisons de lisibilité de la figure

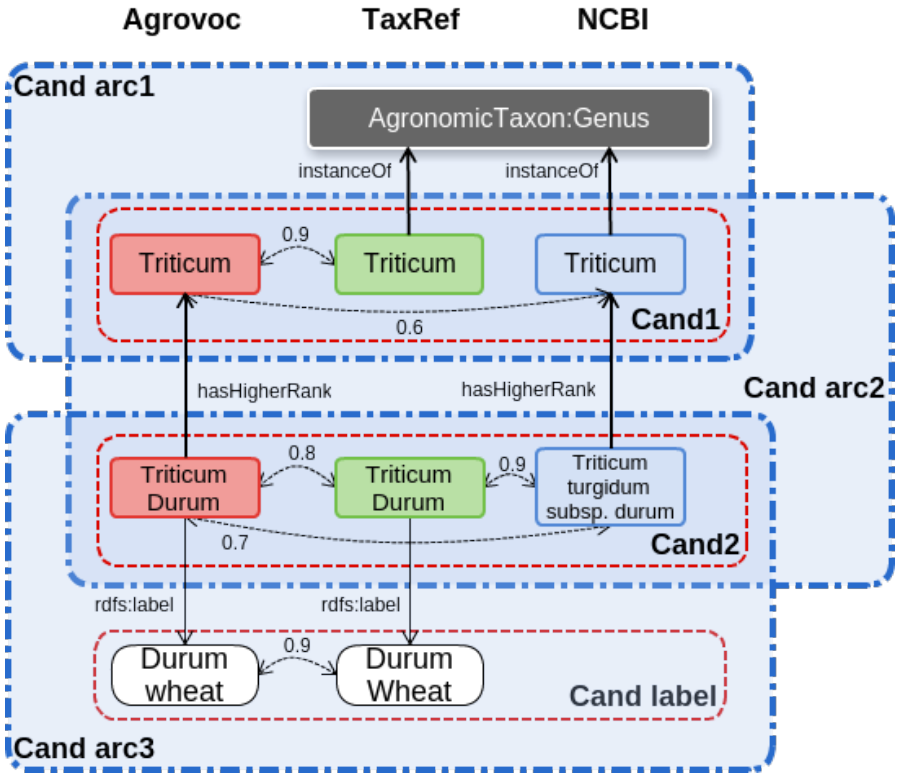


FIGURE 27 – Exemple de candidats arcs

### 4.3.3. Algorithme de génération des candidats

La génération des candidats repose sur deux étapes importantes. La première est la génération de candidats sommets, qui se fait à partir des correspondances générées par les outils d'alignement. L'idée de la génération de ces candidats sommets est de trouver les composantes connexes dans le graphe multiparti $SA$ composé des $N$ KB alignées. Une condition particulière est qu'un candidat ne peut contenir plusieurs éléments ontologiques provenant de la même source. Nous allons chercher uniquement les candidats incluant un maximum d'éléments ontologiques. Si un candidat est inclus dans un autre, alors nous ne garderons que le candidat de taille supérieure. C'est ce que nous appellerons les candidats maximaux.

La génération des candidats sommets à partir du graphe SA revient à effectuer une recherche des composantes connexes dans un graphe multiparti. En effet, chaque élément ontologique d'un candidat doit être lié à tous les autres par une chaîne de correspondances pour faire partie du même candidat. Pour trouver les composantes connexes dans un graphe non-orienté, l'algorithme le plus simple mais aussi le plus efficace est la recherche en profondeur. Cet algorithme se fonde sur un parcours de graphe en explorant le voisinage des sommets étudiés. Il est dit "en profondeur" puisqu'il privilégie d'abord le voisinage du premier voisin étudié avant d'étudier le second voisin.

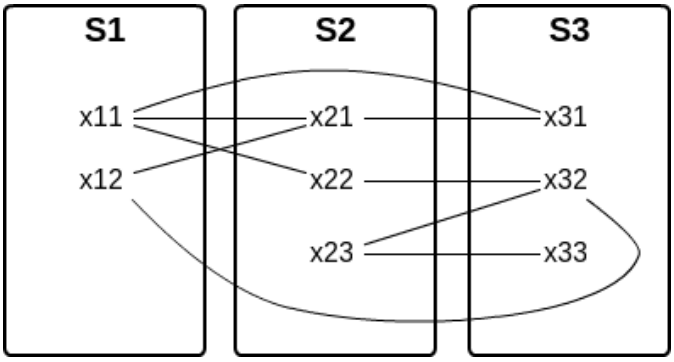


FIGURE 28 – Exemple de graphe non orienté multi-partite

Nous pouvons prendre l'exemple présenté dans la figure 28. Cette figure présente un exemple de graphe multiparti non-orienté. Il contient 3 parties ("S1", "S2", "S3"), chacune contenant 2 ou 3 sommets. Les traits entre les sommets représentent les arêtes du graphe multiparti. Si nous appliquons un algorithme de recherche en profondeur en partant du sommet "x11", nous commencerons par récupérer tous ses voisins, ici "x21", "x22" et "x31". En considérant le premier voisin ("x21"), nous parcourons d'abord les voisins de "x21" avant de parcourir les autres voisins de "x11". Chaque voisin parcouru de cette manière est ajouté à la composante connexe. Nous avons une contrainte supplémentaire qui s'ajoute à notre problème qui est qu'un candidat ne peut pas contenir plus d'un élément par source. Il faut alors vérifier dans notre algorithme qu'une composante connexe n'est pas de taille supérieure au nombre de parties et qu'il n'y a pas deux éléments de la même partie.

Les algorithmes classiques de parcours en profondeur utilisent un système de marquage des sommets pour éviter les boucles infinies lors de l'apparition de cycles dans le graphe. Ce problème ne se pose pas ici puisque la condition d'un sommet par partie permet d'éviter les cycles lors du parcours en profondeur. En effet, l'algorithme ne repassera jamais par un sommet déjà étudié puisque la composante connexe contient déjà un sommet de cette partie.

L'algorithme considère un candidat finalisé lorsque sa taille est égale au nombre de parties, ou qu'il n'y a pas de voisin des sommets de la composante connexe qui peut être ajouté. Cette dernière condition apparaît lorsque les voisins des sommets appartiennent à des parties qui ont déjà un élément dans la composante.

Nous pouvons faire l'analogie entre le graphe multiparti et le graphe SA présenté précédemment, chaque partie représentant une source considérée, les sommets étant des éléments ontologiques. Les arêtes du graphe sont des correspondances issues du système d'alignement.

L'algorithme 2 présente la boucle principale de l'algorithme de génération des candidats sommets. L'intérêt principal de cette boucle étant le parcours de tous les noeuds de SA et d'appliquer l'algorithme DFS [80] sur tous les éléments qui ne sont pas encore impliqués dans un des candidats déjà générés (color(v) == null). Les éléments déjà impliqués dans

80. Depth First Search ou recherche en profondeur en français

```
SA = Graph aligned SKB;
allCands = new List<Cand>();
forAll(v in V_{sa})
{
    if(color(v) == null)
    {
        Cand = new Cand();
        Cand.addElem(v);
        allCands.addAll(DFS(Cand, allCands));
    }
}
```

**Algorithm 2:** Algorithme de génération des candidats

au moins un candidat ont déjà été explorés pour la création de tous les candidats possibles. Nous cherchons ici à découvrir les éléments isolés. Pour chaque nouveau candidat ajouté à la liste finale, chaque élément ontologique du candidat en question est marqué par le candidat en question. De cette manière, il est possible de savoir pour chaque élément ontologique (ou nœud de SA) s'il est présent dans des candidats déjà générés et si oui lesquels.

L'algorithme 3 présente le parcours en profondeur avec les conditions que nous avons présentées précédemment. La fonction "DFS" présentée ici prend en paramètres le candidat en cours de construction ("Cand") et l'ensemble des candidats générés ("allCands"). La première condition "Cand.size() < N", impose que la taille du candidat soit inférieure au nombre de sources considérées. Dans le cas contraire, le candidat est déjà maximal ; il est alors ajouté à la liste des candidats générés (ligne 24). Si la taille du candidat est inférieure au nombre de sources, alors nous récupérons tous les voisins du candidat (ligne 5). La liste des voisins d'un candidat est l'union des voisins de tous les éléments contenus dans ce candidat. Pour chaque voisin, nous vérifions si le candidat contient déjà un élément de cette source (ligne 9). Si ce n'est pas le cas, alors nous créons un nouveau candidat identique au candidat en cours de construction (ligne 12), auquel nous ajoutons le nouvel élément (ligne 13). Ce nouveau candidat va nous permettre de faire l'appel récursif à la fonction "DFS" (ligne 14). Si aucun voisin n'a été ajouté au candidat (ligne 17) alors le candidat est finalisé. Dans ce cas, il est ajouté à la liste des candidats finalisés (ligne 19).

La fonction "addCand" présentée entre la ligne 30 et 37 permet d'ajouter un candidat à la liste des candidats finalisés. Pour cela, nous regardons s'il n'existe pas déjà un candidat identitique dans la liste (ligne 32). Nous considérons deux candidats comme identiques s'ils contiennent les mêmes éléments. Cette condition permet d'éviter l'ajout de doublon. Si ce n'est pas le cas, alors nous l'ajoutons à la liste des candidats finalisés (ligne 34) et nous colorons les éléments contenus dans le candidat (ligne 35). Cette coloration permettra d'éviter d'effectuer le parcours en profondeur en partant de cet élément puisqu'il est déjà dans un candidat.

```
DFS(Cand, allCands)
{
    if(Cand.size() < N)
    {
        nbs = getAllNeighboors(Cand);
        hasNbSuitable = false;
        for(nb in nbs)
        {
            if(!Cand.getElemForSource(nb.getSource())
            {
                hasNbSuitable = true;
                newCand = Cand.clone();
                newCand.addElem(nb);
                allCands.addAll(DFS(newCand, allCands));
            }
        }
        if(!hasNbSuitable)
        {
            addCand(Cand, allCands);
        }
    }
    else
    {
        addCand(Cand, allCands);

    }
    return allCands;
}

addCand(Cand, allCands)
{
    if(!allCands.contains(Cand))
    {
        allCands.add(Cand);
        setColor(Cand);
    }
}
```

**Algorithm 3:** Algorithme Depth First Search (parcours en profondeur)

Si nous appliquons cet algorithme à l'exemple présenté sur la figure 28, nous pouvons définir la trace (non exhaustive) suivante :

```
Cand : {x11}, voisins : {x21, x22, x31}
```

```
    Cand : {x11, x21}, voisins : {x12, x31, x22}
        Cand : {x11, x21, x12} -> deux éléments de la même source
        Cand : {x11, x21, x31} -> nouveau candidat
        Cand : {x11, x21, x22} -> deux éléments de la même source
    Cand : {x11, x22}, voisins : {x32, x21, x31}
        Cand : {x11, x22, x32} -> nouveau candidat
        Cand : {x11, x22, x21} -> deux éléments de la même source
        Cand : {x11, x22, x31} -> nouveau candidat
    Cand : {x11, x31}, voisins : {x21, x22}
        Cand : {x11, x31, x21} -> candidat déjà présent
        Cand : {x11, x31, x22} -> candidat déjà présent
Cand : {x12}, voisins : {x21, x32}
    Cand : {x12, x21}, voisins : {x11, x31, x32}
        Cand : {x12, x21, x31} -> nouveau candidat
...
```

Dans cet exemple, le premier nœud considéré est "x11". Nous parcourons alors tous ses voisins : "x21, x22, x31". Nous considérons alors "x21", le premier voisin découvert de "x11". L'ensemble des voisins du candidat en cours de construction "x11, x21" sont "x12, x31, x22". Si nous considérons l'ajout de l'élément "x12" dans le candidat en cours de construction, nous obtenons "x11, x21, x12". Les éléments "x11" et "x12" sont dans la même source ; ce candidat n'est donc pas possible. Le candidat suivant considéré est "x11, x21, x31" qui est valable ; un nouveau candidat est donc ajouté. De cette manière, tout le graphe est parcouru pour découvrir tous les candidats possibles. Nous pouvons observer que si un candidat est déjà présent, il n'est pas ajouté à la liste.

Une fois ces candidats sommets générés, il est possible de déterminer les candidats arcs liés à ces candidats sommets. Pour chaque couple de candidats sommets, si le même arc est présent dans au moins une source, un candidat arc est généré. Pour chaque relation présente pour un des éléments ontologiques d'un candidat, cette même relation est recherchée dans les autres sources pour les autres éléments ontologiques du même candidat. Ceci est réalisable facilement puisque nous savons que les relations utilisées sont étiquetées par des propriétés du module, donc forcément les mêmes URI. Si cette relation pointe vers un autre candidat, alors le candidat arc est défini entre les deux candidats sommets.

Un cas particulier est le candidat sommet label. Les systèmes d'alignement ne considérant pas les labels comme étant des éléments ontologiques, ceux-ci ne génèrent pas de correspondances entre ces labels. Nous utilisons dans ce cas le même algorithme que celui défini pour les candidats sommets, à la différence que nous utilisons une fonction de similarité de chaînes de caractères pour déterminer les correspondances entre labels. Pour chaque label des éléments ontologiques d'un candidat sommet, nous regardons si celui-ci est similaire à un autre label d'un élément ontologique du candidat provenant d'une autre source. Pour calculer la similarité, nous utilisons la distance de Jaro-Winkler [Winkler, 1990, Winkler, 1999]. Nous considérons deux labels comme "identiques" si le résultat de la similarité de Jaro-Winkler est supérieur à 0,92. La définition de ce seuil

s'inspire de l'outil de recherche d'information Apache Lucene [McCandless et al., 2010]. Cet outil utilise la fonction de calcul de similarité et le même seuil. En considérant les labels comme les sommets et les liens entre labels identiques, nous pouvons utiliser le même algorithme que pour la génération de candidats sommets. Pour chaque candidat label généré, un candidat arc est créé entre le candidat sommet considéré et le candidat label généré. Nous ne considérons pas les candidats label comme étant des candidats sommets au même titre que les candidats sommets représentant des individus ou des classes. Dans la suite du processus, nous ne considérerons pas le candidat sommet label et le candidat arc associé comme distincts. En effet, nous posons une contrainte forte sur l'existence du candidat sommet label uniquement si le candidat arc associé est dans la base de connaissances finale. Il n'est donc pas possible de considérer le candidat sommet label indépendemment du candidat arc associé.

## 4.4. Calcul de la confiance d'un candidat

Nous générons des candidats pour identifier quels sont les éléments qui peuvent, potentiellement, apparaître dans la base de connaissance finale. Cette appartenance potentielle est estimée à partir d'un score de confiance qui est attribué à chaque candidat en fonction de son apparition dans les différentes sources. Nous considérons que si un candidat implique plusieurs sources, alors ces sources sont d'accord entre-elles sur cette représentation. Nous cherchons donc à déterminer le niveau de consensus du candidat dans le contexte du processus de transformation multi-sources.

Nous utiliserons dans ce manuscrit la définition du terme "consensus" donnée par le CNRTL [81] :

> Accord de plusieurs personnes, de plusieurs textes dans un domaine déterminé. Remarque : Dans l'usage récent, consensus glisse vers la signification « opinion ou sentiment d'une forte majorité ». Le syntagme large consensus se lexicalise (cf. Gilb. 1971).

La remarque de cette définition est particulièrement intéressante puisqu'elle précise que le consensus est défini lors d'un accord parmi "une forte majorité". Cette définition permet de faire la distinction entre le consensus et l'unanimité. Cette dernière définit l'accord et l'approbation de tous les participants, alors que le premier exprime la notion de "forte majorité". Le consensus n'exclut pas, contrairement à l'unanimité, les contradictions et les absences d'opinions.

### 4.4.1. Trust likelihood

Suivant l'acception du "consensus" énoncée ci-dessus, il est possible de définir une première fonction de définition du niveau de consensus sous une forme de vote classique. Le Web sémantique se fondant sur l'hypothèse du monde ouvert, l'absence d'un élément n'est pas considérée comme valant faux mais laisse l'incertitude sur son existence. De ce

81. Centre Nationale de Ressources Textelles et Lexicales - http://www.cnrtl.fr/definition/consensus

fait, si un élément est présent dans une source, alors cette source participe à la confiance consensuelle de l'élément. Dans le cas contraire, il n'existe pas de confiance consensuelle contre l'existence de l'élément, mais seulement de l'incertitude. Par exemple si deux sources contiennent le même élément et une dernière source ne le contient pas, alors il y a deux sources participant à la confiance consensuelle de l'élément, mais pas la troisième.

Nous pouvons définir une première façon de calculer le niveau de consensus, assimilable à la fonction "likelihood" issue des probabilité (fonction de vraisemblance en français). Nous évaluons la confiance consensuelle d'un élément en calculant le ratio entre le nombre de sources impliquées dans un candidat et le nombre total des sources considérées. Nous définissons $trust_{likelihood}$ en considérant $Cand$ le candidat étudié. Si le candidat est un candidat sommet, alors ses composantes sont $CandS = (V_{CandS}, E_{CandS}, \Sigma V_{CandS}, etiqV_{CandS}, valueE)$. Cette fonction ne prend en compte que l'ensemble des sommets du candidat $V_{CandS}$ et le nombre de sources alignées $N$.

Trust likelihood pour les candidats sommets

$$trust_{likelihood}(CandS) = \frac{|V_{CandS}|}{N} \tag{2}$$

Si le candidat est un candidat arc, alors ses composantes sont :
$CandA = (V_{CandA}, E_{CandA}, \Sigma V_{CandA}, \Sigma E_{CandA}, etiqV_{CandA}, sourceE_{CandA}, cibleE_{CandA})$.
Cette fonction ne prend en compte que l'ensemble des arcs du candidat $E_{CandA}$ et le nombre de sources alignées $N$.

Trust likelihood pour les candidats arcs

$$trust_{likelihood}(CandA) = \frac{|E_{CandA}|}{N} \tag{3}$$

Si le candidat est un candidat arc de type label, alors nous considérons uniquement le calcul de la confiance concernant la relation. Les sommets labels sont considérés comme identiques d'après le score de similarité, comme nous l'avons vu dans la section 4.3.3 du chapitre III.

La figure 29 est un exemple de la façon dont est calculé le score de confiance $trust_{likelihood}$. Nous pouvons voir que les deux candidats sommets ont le même score (1) puisque tous les deux impliquent les trois sources considérées (ici $N = 3$). Or les correspondances utilisées pour générer $Cand2$ sont plus nombreuses, et donc plus pertinentes, que celles utilisées pour générer $Cand1$.

### 4.4.2. Trust Degree

La fonction de calcul $trust_{likelihood}$ considère uniquement le nombre de sources impliquées dans le candidat. Les candidats étant générés à partir de plus ou moins de correspondances, il semble pertinent de prendre en compte les correspondances dans le

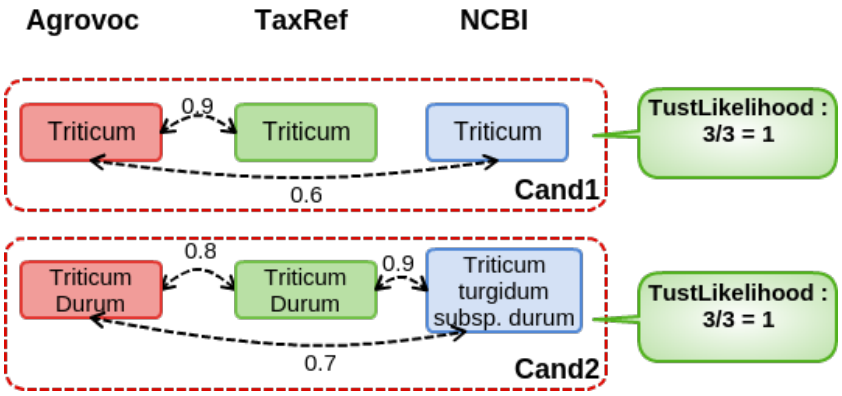


FIGURE 29 – Calcul des scores de confiance avec trustLikelihood

calcul de la confiance. Par exemple sur la figure 29, les deux candidats ont le même score de confiance avec $trust_{likelihood}$ alors que "Cand2" semble plus cohérent que "Cand1" par rapport aux correspondances. Pour prendre en considération ces correspondances dans la fonction de calcul de la confiance consensuelle, nous définissons la fonction $trust_{degree}$. Cette fonction se fonde sur le ratio entre la somme des degrés de fiabilité des correspondances utilisées pour générer le candidat sur le nombre de correspondances possibles entre toutes les sources considérées. De cette façon, nous gardons la propriété selon laquelle la confiance est supérieure si plus de sources sont impliquées dans le candidat. En prenant en compte la somme des degrés de fiabilité des correspondances utilisées, nous ajoutons la propriété selon laquelle un candidat est d'autant plus digne de confiance qu'il a été généré à partir de correspondances avec des degrés de fiabilité élevés. Pour cela, nous définissons la fonction $trust_{degree}$ en considérant $CandS$ le candidat sommet étudié, ainsi que ses composantes $CandS = (V_{CandS}, E_{CandS}, \Sigma V_{CandS}, etiqV_{CandS}, valueE)$. Cette fonction prend en compte $E_{CandS}$ l'ensemble des arcs du candidat, $N$ le nombre de sources alignées et $valueE$ l'application qui à chaque arc de $E_{CandS}$ associe son degré de fiabilité. Cette considération du degré de fiabilité des correspondances utilisées, cette fonction de calcul de la confiance n'est applicable que pour les candidats sommets. Dans ce cas, nous considérons la fonction $trust_{likelihood}$ pour calculer la confiance des candidats arcs.

Trust degree pour les candidats sommets

$$trust_{degree}(CandS) = \frac{\sum_{e_i \in E_{CandS}} valueE(e_i)}{\frac{N(N-1)}{2}} \quad (4)$$

Cette fonction fait la somme de tous les degrés de fiabilité des correspondances utilisées pour générer le candidat. Cette somme est normalisée en divisant le résultat par le nombre maximum de correspondances possibles, c'est-à-dire le nombre de paires possibles entre toutes les sources considérées. Les correspondances étant utilisées dans

le processus de génération des candidats, cette fonction permet de prendre en compte indirectement le nombre d'éléments ontologiques présents dans le candidat. Cette fonction est proportionnelle au nombre d'arêtes : plus le graphe du candidat contiendra d'arêtes, plus il contiendra de nœuds.

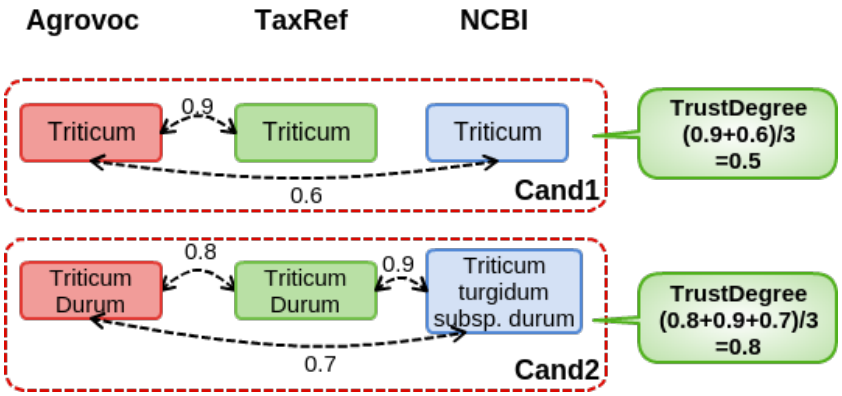


FIGURE 30 – Calcul des scores de confiance avec trustDegree

Nous pouvons observer les résultats des calculs d'après l'exemple précédent dans la figure 30, mais cette fois-ci avec la fonction de calcul $trust_{degree}$. Les résultats sont différents et nous observons que $trust_{degree}(Cand2) > trust_{degree}(Cand1)$. Nous pouvons donc ordonner les candidats suivant la fiabilité des correspondances utilisées pour générer ces candidats.

Concernant les candidats arcs, nous définissons la fonction de calcul de confiance $trust_{degree}$ de la même manière que $trust_{likelihood}$.

Trust degree pour les candidats arcs

$$trust_{degree}(CandA) = trust_{likelihood}(CandA) = \frac{|E_{CandA}|}{N} \quad (5)$$

**Trust degree avec propagation**

La confiance d'un candidat arc peut être influencée par la confiance des candidats sommets reliés par ce candidat arc. Nous considérons ici que la confiance d'un candidat arc est d'autant plus grande que les candidats sommets qu'il relie ont un haut score de confiance. Nous définissons une confiance $trust_{degree_p}$ pour les candidats arcs prenant en compte ce phénomène de propagation de la confiance.

Tout d'abord, si le candidat arc relie deux candidats sommets, alors nous pouvons définir la confiance $trust_{degree_p}$ pour calculer la confiance de ce candidat arc.

Trust degree avec propagation pour candidat arc entre deux candidats sommets

$$trust_{degree_p}(CandA) = \frac{|E_{CandA}| + \frac{trust_{degree}(CandS1)+trust_{degree}(CandS2)}{2}}{N+1} \quad (6)$$
$$\text{tel que } CandS1 \text{ et } CandS2 \text{ sont liés par } CandA$$

Cette formule prend en compte $|E_{CandA}|$ (le nombre d'arcs présents dans le candidat arc) et la moyenne des scores de confiance des candidats sommets liés par le candidat arc. Nous avons normalisé ce résultat avec $N$, qui est la valeur maximale de nombre d'arcs, plus 1, qui est la valeur maximum que la moyenne de deux scores de confiance des candidats sommets peut valoir. Si un candidat arc relie un candidat sommet à un sommet du module, alors le $trust_{degree}$ du sommet du module est égal à 1.

Dans le cas d'un candidat arc représentant une relation "rdfs :label" alors nous pouvons, là aussi, prendre en compte la propagation de la confiance du candidat sommet associé.

Trust degree avec propagation pour candidat arc "rdfs :label"

$$trust_{degree_p}(CandA) = \frac{|E_{CandA}| + trust_{degree}(CandS)}{N+1} \quad (7)$$
$$\text{tel que } CandS \text{ est lié à } CandA$$

### 4.4.3. Intégrale de Choquet

#### Pourquoi l'intégrale de Choquet ?

Le score de confiance que nous cherchons à définir ici est en fait la représentation de la notion de possibilité définie par [Dubois et al., 1988]. Il est d'usage de considérer qu'une notion d'imprécision est assimilable à la théorie des probabilités [Dubois and Prade, 1985]. Cela implique de considérer une notion d'aléa dans notre conception du candidat. Le score de confiance que nous définissons représenterait alors la chance que les experts valident ce candidat. Cette interprétation n'est pas exacte car la validation d'un candidat par un expert [82] est un fait déterminé et non un événement aléatoire comme ça pourrait être le cas si on lance un dé par exemple. Notre score de confiance d'un candidat représente le niveau de possibilité qu'un expert le valide. En d'autres termes, plus le score de confiance du candidat est bas et plus il serait surprenant qu'un expert le valide. A contrario, plus son score de confiance est élevé et plus sa validation par un expert est vraisemblable. Le score de confiance que nous cherchons à définir pour un candidat est donc assimilable au score de possibilité dans la théorie de la logique possibiliste [Dubois et al., 1988].

82. Nous considérons ici que la validation d'un expert est stable. Si nous présentons plusieurs fois le même candidat à l'expert, il le validera de la même façon à chaque fois.

Cette théorie, introduite par Zadeh en 1978 [Zadeh et al., 1978], a été étendue et appliquée à l'informatique par Dubois et al. [Dubois et al., 1988]. La représentation formelle de la possibilité d'un prédicat implique deux valeurs : la possibilité (pos) et la nécessité (nec). Ces deux valeurs permettent de faire la distinction entre le degré de possibilité d'une proposition et son degré de certitude. En d'autres termes, elles sont respectivement la représentation formelle des phrases : "il est possible que ..." et "il est certain que ...".

Considérons la proposition p incertaine, si nous appliquons la théorie des possibilités sur cette proposition, alors les conditions suivantes sont garanties :

Théorie des possibilités

- $nec(p) \in [0, 1]$
- $pos(p) \in [0, 1]$
- $pos(p) \leq nec(p)$
  - Une proposition est possible avant d'être certaine
- $nec(p) = 1 - pos(\bar{p})$
  - La certitude d'une proposition est égale à l'inverse de la possibilité de la proposition contraire
- $nec(p) = 1 \Rightarrow pos(p) = 1$
  - Une proposition totalement certaine est aussi totalement possible
- $pos(p) = 0 \Rightarrow nec(p) = 0$
  - Une proposition pas du tout possible, n'est également pas du tout certaine
- $nec(p) = 0 \Rightarrow pos(p)$ incertaine
  - Si une proposition n'est pas du tout certaine, cela n'apporte aucune information sur sa possibilité
- $pos(p) = 1 \Rightarrow nec(p)$ incertaine
  - Si une proposition est totalement possible, cela n'apporte aucune information sur sa certitude

Dans [Khaleghi et al., 2013], les auteurs dressent un panel des différentes méthodes pour exprimer l'incertitude et les différents types d'incertitudes existants. Ces travaux précisent que la théorie des possibilités permet de manipuler des informations incertaines. De plus, [Hassine-Guetari, 2014] précise que la méthode la plus adaptée pour agréger différentes valeurs d'utilité pour définir la possibilité d'une proposition est l'intégrale de Choquet.

### Présentation de l'intégrale de Choquet

Cette intégrale, initialement proposée par [Schmeidler, 1986] puis étendue par Murofushi et Sugeno dans [Murofushi and Sugeno, 1991] peut se représenter sous la forme suivante (définition inspirée par [Labreuche et al., 2008]) :

Intégrale de Choquet

$$C_\mu(a) = a_{\tau(1)}\mu(N) + \sum_{i=2}^{n}(a_{\tau(i)} - a_{\tau(i-1)})\mu(\{\tau(i), ..., \tau(n)\}) \quad (8)$$

avec $\tau$ est une permutation sur $N$ telle que $a_{\tau(1)} \leq ... \leq a_{\tau(n-1)} \leq a_{\tau(n)}$

L'intégrale de Choquet est particulièrement adaptée pour la fusion multi-critères. Si nous considérons l'ensemble $a$ comme étant la définition de chaque critère, alors nous considérons que chaque élément de $a$ est constitué du critère lui-même ($\tau(i)$) et de la valeur comprise entre 0 et 1 qui lui a été attribuée ($a_{\tau(i)}$). La fonction $\mu(x)$ permet de représenter l'*intérêt* de l'ensemble de critères dans $x$. Ce que nous appelons ici *intérêt* représente le gain que représente la combinaison de cet ensemble de critères.

Prenons par exemple la détermination du choix de l'achat d'un appartement. Nous souhaitons ici un appartement qui respecte 3 critères : grand, bien situé et disposant d'une place de parking. Pour un appartement considéré, nous définissons les valeurs pour chaque critère témoignant de la pertinence de l'appartement par rapport à ce critère. Par exemple pour un appartement nous définissons la valeur 0,8 pour le critère "grand", 0,9 pour le critère "bien situé" et 0,6 pour le critère "place de parking". Afin d'obtenir une valeur représentant la notion de "bon appartement", nous agrégeons ces différents critères en utilisant l'intégrale de Choquet. Pour simplifier la notation dans l'équation suivante, nous utiliserons "a" pour définir le critère "grand", "b" pour définir le critère "bien situé" et "c" pour le critère "place de parking". Nous obtenons alors le calcul suivant :

$$C_\mu(appartement) = \mu(a,b,c) * (0,6-0) + \mu(a,b) * (0,8-0,6) + \mu(b) * (0,9-0,8) \quad (9)$$

Nous avons donc ordonné les critères suivant les valeurs associées. Pour chaque valeur, nous avons effectué une *$\alpha$-coupe* sur les critères de façon à ne garder que les critères qui ont une valeur supérieure ou égale à la coupe. La première coupe (sur 0,6) considère obligatoirement tous les critères. La deuxième coupe (sur 0,8) ne considère plus le critère "c" puisque sa valeur est inférieure à la coupe. Pour chaque coupe, nous multiplions la valeur de la coupe moins la valeur de la coupe précédente par l'intérêt des critères présents dans la coupe. Pour la première coupe (à 0,6) nous multiplions l'intérêt des critères a, b et c par la valeur de la coupe (0,6) puisqu'il n'y a pas de coupe précédente. Pour la deuxième coupe nous multiplions l'intérêt des critères dans cette coupe (a et b) par la valeur de la coupe moins la valeur de la coupe précédente (0,8 - 0,6). Nous procédons de la même manière pour la dernière coupe. Nous sommons ensuite toutes ces valeurs pour obtenir la valeur de l'agrégation des trois critères pour cet appartement. Si la fonction $\mu(x)$ retourne une valeur comprise entre 0 et 1 et que les valeurs pour chaque critère sont elles-aussi comprises entre 0 et 1, alors la valeur de l'intégrale de Choquet sera forcément elle aussi comprise entre 0 et 1.

La fonction $\mu(x)$ représente l'intérêt des critères. Il faut donc pouvoir définir une fonction permettant d'obtenir une valeur entre 0 et 1 à partir d'un ensemble de critères. Par exemple pour $\mu(a,b)$ nous devons définir l'intérêt d'avoir un appartement grand et

bien situé mais sans place de parking. Nous pouvons alors définir arbitrairement pour chaque combinaison de critères une valeur d'intérêt. Une autre méthode serait de définir des valeurs pour chaque critère en considérant les dépendances entre critères si elles existent. Par exemple, nous pouvons définir que le critère "place de parking" a un intérêt de 0,9 si la valeur du critère "bien situé" est en dessous de 0,5, sinon le critère "place de parking" a un intérêt de 0,4. Ceci permet de définir la contrainte "la place de parking est très importante si l'appartement n'est pas bien situé, sinon la place de parking n'est pas très importante".

Considérons pour l'exemple que les critères sont indépendants et de même intérêt (aucune préférence n'est établie entre ces critères). Nous définissons la fonction $\mu(x)$ comme étant le ratio entre le nombre de critères dans $x$ et le nombre de critères considérés dans le processus. Par exemple, nous avons : $\mu(a,b) = \mu(a,c) = \mu(b,c) = 2/3$. Nous pouvons retrouver aussi : $\mu(a,b,c) = 3/3$.

De cette manière, nous pouvons compléter l'équation précédente avec les valeurs de la fonction d'intérêt des critères de la manière suivante :

$$\begin{aligned} C_\mu(appartement) &= (3/3) * (0,6 - 0) + (2/3) * (0,8 - 0,6) + (1/3) * (0,9 - 0,8) \\ &= 0,76 \end{aligned} \tag{10}$$

L'intuition de la pertinence de l'intégrale de Choquet est présentée dans la figure 31. Sur cette figure sont représentées en abscisse les différentes valeurs d'intérêt des trois $\alpha$-coupes présentes dans l'exemple du choix d'un appartement. L'axe des ordonnées présente les différentes valeurs des $\alpha$-coupes. Nous cherchons, par l'intermédiaire de cette intégrale, à calculer la somme des aires des rectangles.

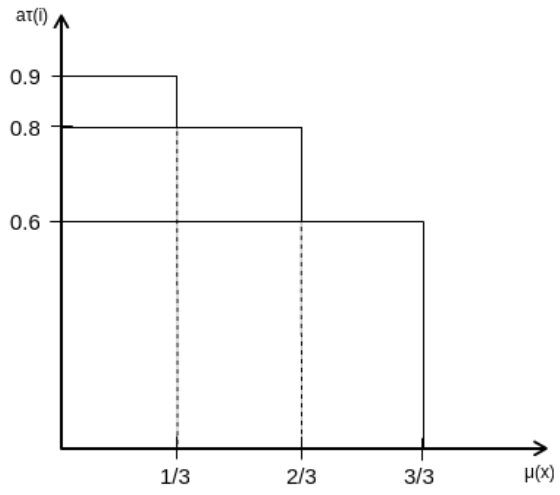


FIGURE 31 – Intuition de l'intégrale de Choquet

### Utilisation de l'intégrale de Choquet dans le calcul de la confiance consensuelle d'un candidat

De la même manière, nous pouvons considérer cette intégrale pour obtenir le score de confiance d'un candidat. Pour cela, nous considérons les sources impliquées comme étant les critères de l'intégrale. Comme nous venons de le voir dans l'exemple de la décision d'un appartement à acheter, pour chaque appartement doit être définie une valeur par critère permettant d'évaluer l'adéquation de l'appartement au critère. Dans le cas de la définition du score de confiance d'un candidat, cette valeur est l'adéquation du candidat par rapport à une source donnée. Cela revient à déterminer la force de l'implication de la source dans la construction de ce candidat. Cette notion peut être calculée en fonction des correspondances utilisées avec l'élément ontologique provenant de ce candidat. En effet, nous considérons que plus un élément ontologique provenant d'une source est lié avec des correspondances fortes vers les autres éléments ontologiques du candidat, plus la source est impliquée dans la construction de ce candidat.

Prenons l'exemple présenté sur la figure 26 à la page 98 et en particulier le candidat "Cand1". Nous pouvons remarquer dans cet exemple que l'élément ontologique "Triticum" provenant de la source Agrovoc a une implication plus forte que les autres. Cette implication peut s'observer au travers des correspondances utilisées, liant l'élément provenant d'Agrovoc vers les deux autres éléments des deux autres sources considérées. Les autres éléments ontologiques n'étant liés qu'à un seul autre élément, leur implication est donc moindre. Pour représenter formellement cette implication, nous pouvons poser la fonction de calcul qui somme les degrés de fiabilité des correspondances liant l'élément ontologique provenant de la source. Afin de normaliser cette valeur, nous divisons cette somme par l'implication maximum que nous pouvons observer, c'est à dire une correspondance avec un degré de fiabilité de 1 vers un élément ontologique de toutes les sources considérées. En d'autres termes, nous divisons par le nombre de sources considérées moins un. Nous calculerons l'implication d'une source $SKB_i$ par rapport à un candidat sommet $CandS$ en utilisant l'équation 11. Rappelons qu'une source $SKB$ est un multigraphe orienté étiqueté $SKB = (V_{skb}, E_{skb}, \Sigma V_{skb}, \Sigma E_{skb}, etiqV_{skb}, etiqE_{skb}, sourceE_{skb}, cibleE_{skb})$. Un candidat sommet $CandS$ est un graphe non-orienté $CandS = (V_{CandS}, E_{CandS}, \Sigma V_{CandS}, etiqV_{CandS}, valueE)$

Implication d'une source dans un candidat

$$implication(SKB, CandS) = \frac{\sum_{\substack{e \in E_{CandS} \\ e=(oe_i, oe_j) \ avec \ oe_i \in V_{skb}}} valueE(e)}{N-1} \tag{11}$$

Si nous prenons l'exemple du candidat "Cand1" de la figure 26, l'implication de la

source "Agrovoc" dans ce candidat peut être définie de la manière suivante :

$$\begin{aligned} implication(Agrovoc, Cand1) &= \frac{0,9+0,6}{3-1} \\ &= \frac{1,5}{2} \\ &= 0,75 \end{aligned} \tag{12}$$

Alors que l'implication de la source TaxRef dans ce même candidat peut être évaluée de la manière suivante :

$$\begin{aligned} implication(TaxRef, Cand1) &= \frac{0,9}{3-1} \\ &= \frac{0,9}{2} \\ &= 0,45 \end{aligned} \tag{13}$$

Nous ne considérons ici que la correspondance qui a un degré de fiabilité de 0,9 puisque c'est la seule qui implique l'élément ontologique provenant de la source TaxRef. De la même manière, nous définissons l'implication de la source NCBI dans le candidat "Cand1" de la manière suivante :

$$\begin{aligned} implication(NCBI, Cand1) &= \frac{0,6}{3-1} \\ &= \frac{0,6}{2} \\ &= 0,3 \end{aligned} \tag{14}$$

Cette notion d'implication est particulièrement pertinente dans cet exemple puisque nous pouvons observer que l'élément ontologique provenant d'Agrovoc est central dans la construction de ce candidat. Les deux autres éléments ontologiques n'ont pas de correspondances entre eux. Si l'élément ontologique provenant d'Agrovoc n'était pas présent, alors le candidat n'existerait tout simplement pas. Il est donc cohérent que l'implication de la source Agrovoc soit bien plus grande que l'implication des deux autres sources.

Si un candidat n'a qu'un seul sommet, et donc aucune correspondance à utiliser, alors nous définissons l'implication de la source de la manière suivante : $\frac{1}{N-1}$.

La deuxième valeur à définir pour utiliser l'intégrale de Choquet dans le cadre de l'estimation de la confiance consensuelle d'un candidat est l'intérêt des sources. Comme nous l'avons observé pour l'utilisation de l'intégrale de Choquet dans la décision de l'achat d'un appartement, la définition de l'intérêt des critères est importante pour définir ceux qui sont à privilégier. Puisque nous considérons ici les sources comme des critères, alors nous devons définir l'intérêt de l'implication des sources dans un candidat. Cela permet de définir des priorités entre les sources. Nous pouvons par exemple prioriser les candidats impliquant la source "TaxRef" plutôt que ceux impliquant "Agrovoc". Comme

nous l'avons vu dans la section 2 (Analyse des sources) du chapitre III nous définissons une valeur comprise entre 0 et 1 permettant de représenter la qualité d'une source. Nous pouvons ici réutiliser cette qualité pour définir l'intérêt des sources.

Dans notre exemple nous considérons trois sources avec des niveaux de qualité différents :

**Agrovoc :** 0,6

**TaxRef :** 0,9

**NCBI :** 0,8

Nous devons définir la fonction $\mu(L_S)$ permettant de caractériser l'intérêt d'un sous-ensemble de sources ($L_S$) en fonction des scores de qualité définis dans la section 2 du chapitre III. De cette façon, nous pouvons prendre en compte la diversité des sources. Ceci permet par exemple de considérer au même niveau un candidat impliquant un grand nombre de sources de mauvaise qualité qu'un candidat impliquant peu de sources de très bonne qualité. Nous considérons non seulement que chaque source a un intérêt variable en fonction de sa qualité mais aussi que l'évolution de l'intérêt des sources n'est pas linéaire. Cette non-linéarité permet de prendre en compte une augmentation d'intérêt variable suivant le niveau d'intérêt déjà présent. Par exemple, nous pouvons considérer que si un candidat implique déjà un grand nombre de sources de bonne qualité, alors l'ajout d'une nouvelle source de bonne qualité dans la définition du candidat ne va pas augmenter l'intérêt de manière significative. Notre intuition sur cette répartition non-linéaire est qu'il existe un point représentatif à partir duquel l'intérêt des sources va croître significativement. Ce point d'explosion [83] est spécifique au problème étudié. Il dépend non seulement du nombre de sources considérées mais aussi de la nécessité de favoriser la qualité ou non du résultat. De plus, l'intensité de l'explosion est aussi spécifique au problème étudié.

Nous définissons la fonction $\mu(L_S)$ suivante, en considérant $L_S$ comme étant un sous-ensemble des sources considérées :

Fonction d'intérêt des sources impliquées dans un candidat

$$\mu(L_S) = \frac{\lambda(\sum_{i=1}^{|L_S|} Q(S_i)) - \lambda(0)}{\lambda(\sum_{i=1}^{N} Q(S_i)) - \lambda(0)}$$

- $\lambda(x) = arctan(\frac{x - x_0}{\gamma})$
- $x_0$ est le point d'explosion
- $\gamma$ est l'indice de linéarité de la courbe
  - $0 < \gamma \leq \sum_{i=1}^{N} Q(S_i)$
  - plus $\gamma$ tend vers 0, plus la courbe est crénelée
  - plus $\gamma$ tend vers $\sum_{i=1}^{N} Q(S_i)$, plus la courbe est linéaire
- $\lim_{\sum_{i=1}^{|L_S|} Q(S_i) \to \sum_{i=1}^{N} Q(S_i)} \mu(n) = 1$
- $\mu(0) = 0$

83. Point lorsque la dérivé $\mu'$ est à son maximum

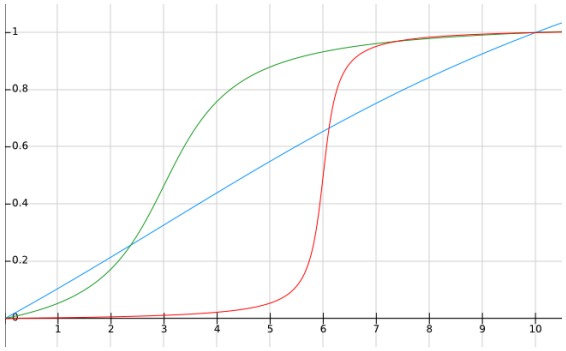


FIGURE 32 – Exemple de répartitions de la fonction $\mu(x)$ avec $\sum_{i=1}^{N} Q(S_i) = 10$, (rouge : $x_0 = 6$, $\gamma = 0{,}2$), (bleu : $x_0 = 3$, $\gamma = 9$) et (vert : $x_0 = 3$, $\gamma = 1$)

La fonction $Q(S_i)$ permet de récupérer le score de qualité de la source $S_i$ définie dans la section 2 du chapitre III. L'équation $\sum_{i=1}^{N} Q(S_i)$ permet d'obtenir la somme des scores de qualité de toutes les sources considérées dans le processus. Dans l'exemple, nous pouvons définir :

$$\sum_{i=1}^{N} Q(S_i) = 0,9 + 0,8 + 0,6 = 2,3 \tag{15}$$

De la même façon, nous utilisons l'équation $\sum_{i=1}^{|L_S|} Q(S_i)$ qui permet d'obtenir la somme des scores de qualité des sources présentes dans le sous-ensemble $L_S$. Ce sous-ensemble représente les sources présentes dans l'$\alpha$-coupe étudiée.

Cette fonction $\mu(L_S)$ permet de représenter l'intérêt du sous-ensemble de sources $L_S$. Nous utilisons la fonction $\lambda(x)$ qui est inspirée de la fonction de répartition de la loi gamma et qui permet d'avoir une répartition qui respecte notre intuition sur l'évolution de l'intérêt des sources. Nous pouvons observer la soustraction de $\lambda(0)$ au résultat de la fonction $\lambda(x)$ appliquée à l'ensemble des sources $L_S$. Cette soustraction permet d'obtenir la valeur caractéristique $\mu(\emptyset) = 0$. De plus, le dénominateur considère l'ensemble de toutes les sources considérées dans le calcul afin de normaliser le résultat entre 0 et 1 avec la valeur caractéristique $\mu(\text{toutes les sources considérées}) = 1$.

Dans la fonction $\lambda(x)$, deux paramètres sont utilisés. Le premier, $x_0$, permet de définir le point d'explosion, point à partir duquel l'intérêt des sources va augmenter particulièrement. Le deuxième paramètre est la valeur $\gamma$. Ce paramètre permet de définir le niveau de linéarité de la courbe. Plus la valeur de $\gamma$ tend vers 0 et plus l'intérêt des sources est crénelé, c'est à dire qu'il est très proche de 0 en dessous de $x_0$ et très proche de 1 au dessus. À l'inverse, plus $\gamma$ s'approche de $\sum_{i=1}^{N} Q(S_i)$ (somme des scores de qualité de toutes les sources considérées) et plus la courbe est linéaire. Pour observer ces variations, nous pouvons nous référer à la figure 32. Sur cet exemple, nous avons considéré que $\sum_{i=1}^{N} Q(S_i) = 10$. Trois courbes sont présentes avec des paramètres différents. Ces courbes permettent de se rendre compte de l'impact de ces paramètres sur la répartition de la courbe.

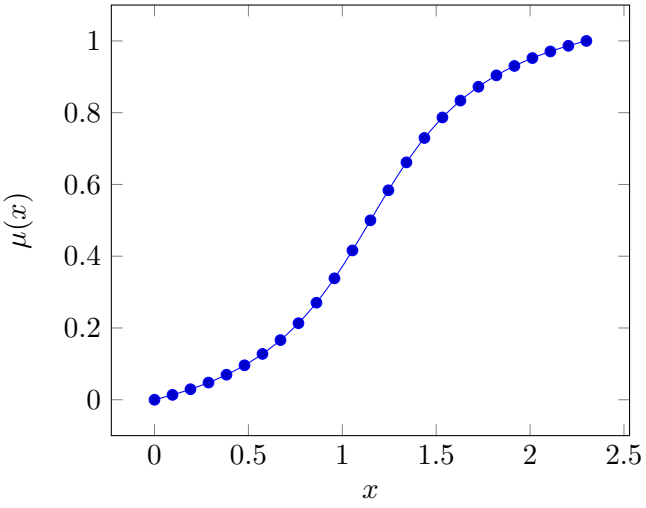


FIGURE 33 – Répartition de la fonction $\mu(x)$ avec les paramètres $\sum_{i=1}^{N} Q(S_i) = 2,3$, $x_0 = 1,15$ et $\gamma = 0,48$

Si nous prenons l'exemple présenté sur la figure 26 à la page 98, nous pouvons appliquer l'intégrale de Choquet pour calculer la confiance des deux candidats sommets définis. Pour cela, nous devons définir les deux paramètres $x_0$ et $\gamma$. Pour définir ces variables, nous utilisons la valeur $\sum_{i=1}^{N} Q(S_i) = 2,3$. Nous souhaitons ici définir le point d'explosion à 50% de la qualité disponible. Cette valeur permet d'obtenir une répartition neutre sur l'importance de la qualité. En effet, plus nous posons une valeur proche de 0 et moins la qualité des sources n'est importante puisque l'intérêt explose pour une faible valeur. Pour cela, nous définissons $x_0 = 0,5 * 2,3 = 1,15$. De la même manière, nous souhaitons un taux de linéarité de la répartition à 20%. Nous obtenons alors $\gamma = 0,2 * 2,3 = 0,48$. La figure 33 présente la répartition de la fonction $\mu(x)$ en fonction des paramètres définis ici.

Comme vu précédemment, si nous considérons le candidat "Cand1" de la figure 26, nous avons les implications des sources suivantes :

— $implication(Agrovoc, Cand1) = 0,75$
— $implication(TaxRef, Cand1) = 0,45$
— $implication(NCBI, Cand1) = 0,3$

Nous pouvons poser le calcul de la confiance du candidat "Cand1" en utilisant l'intégrale de Choquet de la manière suivante :

$$\begin{aligned} trust_{choquet}(Cand1) =& 0,3 * \mu(Agrovoc, TaxRef, NCBI) \\ &+ (0,45 - 0,3) * \mu(Agrovoc, TaxRef) \\ &+ (0,75 - 0,45) * \mu(Agrovoc) \\ =& 0,3 * 1 + 0,15 * 0,77 + 0,3 * 0,14 \\ =& 0,46 \end{aligned} \tag{16}$$

De la même manière, nous pouvons appliquer l'intégrale de Choquet pour calculer la confiance du candidat "Cand2" présenté sur la figure 26. Nous obtenons les implications suivantes :

— $implication(Agrovoc, Cand2) = (0,7 + 0,8)/2 = 0,75$
— $implication(TaxRef, Cand2) = (0,8 + 0,9)/2 = 0,85$
— $implication(NCBI, Cand2) = (0,9 + 0,7)/2 = 0,8$

Nous obtenons alors le calcul suivant :

$$\begin{aligned} trust_{choquet}(Cand2) =& 0,75 * \mu(Agrovoc, TaxRef, NCBI) \\ &+ (0,8 - 0,75) * \mu(TaxRef, NCBI) \\ &+ (0,85 - 0,8) * \mu(TaxRef) \\ =& 0,75 * 1 + 0,05 * 0,86 + 0,05 * 0,29 \\ =& 0,81 \end{aligned} \tag{17}$$

L'intégrale de Choquet est applicable de la même manière sur un candidat arc, mais cela pose un problème : il n'existe pas d'implication d'une source dans la constitution d'un candidat arc car ces candidats ne sont pas générés à partir de correspondances. Nous considérons donc que chaque source est impliquée à un niveau maximum si elle est présente dans le candidat arc. Cette considération implique qu'il n'existe alors qu'une seule $\alpha$-coupe pour un candidat arc puisque toutes les sources impliquées le sont au même niveau. Le calcul de la confiance d'un candidat arc en utilisant l'intégrale de Choquet telle que définie précédemment revient à calculer la fonction $\mu(L_S)$ en considérant $L_S$ comme étant le sous-ensemble des sources impliquées dans le candidat arc. Par exemple nous pouvons considérer le candidat "Cand arc2" dans lequel apparaît la relation "hasHigherRank" entre les candidat sommets "Cand2" et "Cand1". Ce candidat arc implique les sources Agrovoc et NCBI. Si nous appliquons la démarche du calcul de l'intégrale de Choquet concernant ce candidat, nous obtenons les implications suivantes :

— $implication(Agrovoc, Candarc2) = 1$
— $implication(NCBI, Candarc2) = 1$

Par conséquent le calcul de l'intégrale de Choquet est le suivant :

$$\begin{aligned} trust_{choquet}(Candarc2) =& 1 * \mu(Agrovoc, NCBI) \\ =& 1 * 0,70 \\ =& 0,70 \end{aligned} \tag{18}$$

L'utilisation de cette intégrale apporte plusieurs avantages. D'abord nous retrouvons la généricité de $trust_{simple}$ concernant le calcul de la confiance quel que soit le type de candidat. En effet, cette fonction s'applique de la même manière, que ce soit un candidat sommet ou un candidat arc. Nous retrouvons aussi la prise en compte directe de la définition de "consensus" de par la considération des sources impliquées. Ce n'était pas le cas avec $trust_{degree}$ puisque nous nous concentrions sur les correspondances. La considération d'une confiance consensuelle étant spécifique à un problème donné, nous pouvons adapter cette définition avec les paramètres $x_0$ et $\gamma$ de la fonction$\mu(x)$. Nous

retrouvons aussi la force d'implication d'une source pour la constitution du candidat. Cette implication était prise en compte dans la fonction $trust_{degree}$ mais pas dans la fonction $trust_{simple}$. Cette fonction de calcul $trust_{choquet}$ permet de regrouper les avantages des deux fonctions présentées précédemment. De plus, l'utilisation de la fonction $\mu(x)$ la rend particulièrement paramétrable. Nous avons considéré ici que les sources étaient indépendantes mais nous pouvons étendre nos travaux en ajoutant une notion de dépendance entre les sources pour définir leur intérêt. Nous reviendrons sur ce point dans les perspectives.

## 4.5. Découverte de l'extension optimale

### 4.5.1. Hypothèse d'incompatibilité

Comme défini dans la section 4.2.1 du chapitre III, nous n'utilisons ici que des correspondances d'équivalence simple. Néanmoins, les systèmes d'alignement génèrent des correspondances de type 1 : $n$. Ce type de correspondances peut avoir plusieurs sens. Malheureusement, les travaux sur les correspondances complexes ne proposent pas encore de solution pour les désambiguïser de manière automatique. Nous posons donc l'hypothèse de considérer ces relations comme erronées. Cette hypothèse implique que deux candidats sommets contenant un même élément ontologique sont incompatibles.

Une incompatibilité $Inc$ est une paire de candidats sommets partageant un élément ontologique.

$Cand_{S1}$ est un graphe $CandS = (V_{Cand_{S1}}, E_{Cand_{S1}}, \Sigma V_{Cand_{S1}}, etiqV_{Cand_{S1}}, valueE_{CandS1})$ représentant un candidat sommet.

$Cand_{S2}$ est un graphe $CandS = (V_{Cand_{S2}}, E_{Cand_{S2}}, \Sigma V_{Cand_{S2}}, etiqV_{Cand_{S2}}, valueE_{Cand_{S2}})$ représentant un candidat sommet.

Incompatibilité

| |
|---|
| $Inc = (Cand_{S1}, Cand_{S2})$ tel que $\exists oe_{com}, oe_{com} \in V_{Cand_{S1}}, oe_{com} \in V_{Cand_{S2}}$<br>l'ensemble des incompatibilités entre candidats sommets est intitulé $INC$. |

D'après la définition des incompatibilités, celle-ci ne s'appliquent que sur les candidats sommets. Nous ne considérons pas ici la notion d'incompatibilités entre candidats arcs, mais nous aborderons ce sujet dans les perspectives de ce manuscrit.

La figure 34 présente un exemple d'une incompatibilité entre deux candidats sommets. Le candidat étiqueté $Cand1$ et le candidat étiqueté $Cand3$ partagent l'élément ontologique "Triticum" provenant de la source NCBI. Pour cette raison, nous définissons ces deux candidats comme incompatibles. Ils ne pourront donc pas apparaître dans la même base de connaissances finale.

Nous allons maintenant définir le graphe des incompatibilités $G_{inc}$ qui est un graphe non orienté $G_{inc} = \{V_{inc}, E_{inc}, \Sigma V_{inc}, etiqV_{inc}\}$.

Pour construire ce graphe nous allons définir une fonction bijective $URI()$ permettant de construire un identifiant pour tous les candidats issus du graphe des sources alignées $SA$.

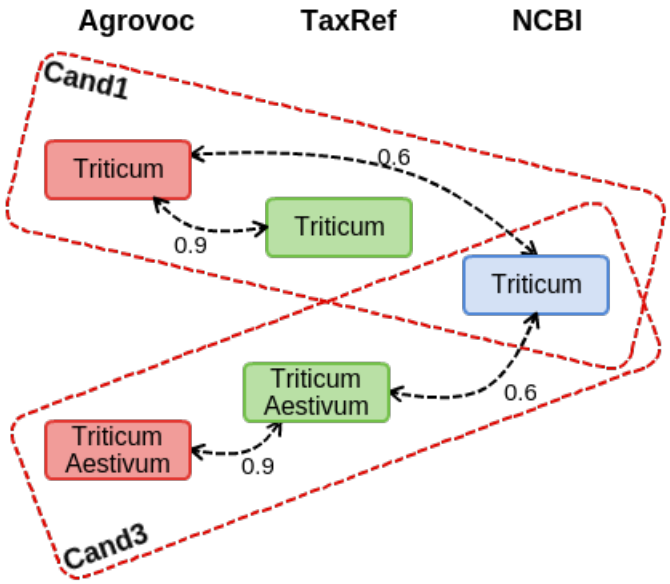


FIGURE 34 – Exemple d'incompatibilité entre deux candidats sommets

Nous définissons les composants du graphe $G_{inc}$ de la manière suivante :

Graphe d'incompatibilité

| |
|---|
| $V_{inc}$ est l'ensemble des sommets de $G_{inc}$. Chaque sommet représente un candidat sommet issu du graphe des sources alignées $SA$. |
| $\Sigma V_{inc}$ est l'ensemble des étiquettes de sommets construit à partir des URI des candidats sommets de $SA$. |
| $etiqV_{inc}$ est la fonction qui associe à chaque sommet de $V_{inc}$ son étiquette dans $\Sigma V_{inc}$. |
| les sommets de $V_{inc}$ remplissent tous la contrainte suivante : $\forall v \in V_{inc}, \exists$ un unique candidat sommet $CandS$ de $SA$ tel que $etiqV_{inc}(v) = URI(CandS)$. |
| $E_{inc}$ est l'ensemble des arêtes de $G_{inc}$ tel que $\exists$ une arête $e \in E_{inc}$ liant deux sommets $v_1 \in V_{inc}$ et $v2 \in V_{inc}$ ssi $\exists$ une incompatibilité $i \in INC$ entre deux candidats sommets $Cand_{S1}$ et $Cand_{S2}$ tels que $etiqV_{inc}(V_1) = URI(Cand_{S1})$ et $etiqV_{inc}(V_2) = URI(Cand_{S2})$. |

### 4.5.2. Définition d'une extension

D'après la définition précédente des incompatibilités entre les candidats sommets, nous allons chercher l'ensemble $Exts$ des extensions $Ext$ représentant un sous-ensemble de candidats sommets compatibles, autrement dit qui ne sont pas incompatibles.

Pour déterminer une extension, nous allons définir un nouveau graphe issu du graphe des sources alignées $SA$ qui est le graphe des extensions $G_{ext} = \{V_{ext}, E_{ext}, \Sigma V_{ext}, \Sigma E_{ext}, etiqV_{ext}\}$. $G_{ext}$ est un graphe non-orienté dont chaque nœud représente un candidat sommet ou arc issu du graphe des sources alignées $SA$.

Graphe des extensions

| |
|---|
| $V_{ext} = V_{sommet} \cup V_{arc}$ est l'ensemble des sommets de $G_{ext}$ composé de deux ensembles disjoints $V_{sommet}$ et $V_{arc}$. Chaque sommet $v \in V_{sommet}$ représente un candidat sommet de $SA$. Chaque sommet $v \in V_{arc}$ représente un candidat arc de $SA$.<br>$\Sigma V_{ext}$ est l'ensemble des étiquettes de sommets construit à partir des URI des candidats de $SA$.<br>$etiqV_{ext}$ est la fonction qui associe à chaque sommet de $V_{ext}$ son étiquette dans $\Sigma V_{ext}$.<br>les sommets de $V_{ext}$ remplissent tous la contrainte suivante : $\forall v \in V_{sommet}, \exists$ un unique candidat sommet $CandS$ de $SA$ tel que $etiqV_{ext}(v) = URI(CandS)$. $\forall v \in V_{arc}, \exists$ un unique candidat arc $CandA$ de $SA$ tel que $etiqV_{ext}(v) = URI(CandA)$.<br>$E_{ext}$ est l'ensemble des arêtes de $G_{ext}$. Ces arêtes indiquent un lien de composition entre un candidat arc $CandA$ et ses candidats sommets $CandS$. Ainsi, l'ensemble des sommets de $CandA$ est inclus dans l'ensemble des sommets de $CandS$. $V_{CandA} \subset V_{CandS}$. |

Dans le cas où il n'existe aucune incompatibilité entre candidats sommets, une seule extension est possible et correspond au graphe des extensions précédemment défini. Dans le cas où il existe des incompatibilités entre candidats sommets, une extension se définit comme un sous-graphe du graphe des extensions tel que ses sommets sont tous compatibles entre eux. Nous définissons donc l'ensemble des contraintes suivantes, que doit respecter une extension $G_{ext} = \{V_{ext}, E_{ext}, \Sigma V_{ext}, \Sigma E_{ext}, etiqV_{ext}\}$ :

Graphe des extensions avec des incompatibilités

| |
|---|
| $\forall v_{s1}, v_{s2} \in V_{sommet}, \nexists inc \in INC$ tel que $inc = (Cand_{S1}, Cand_{S2})$ avec $etiqV_{ext}(v_{s1}) = URI(Cand_{S1})$ et $etiqV_{ext}(v_{s2}) = URI(Cand_{S2})$<br>Il n'existe pas deux sommets de l'extension qui représentent des candidats sommets incompatibles entre eux.<br>$\forall v_{arc} \in V_{arc}$ ssi $\forall e \in E_{ext}$ tel que $e = (v_{arc}, v_{sommet})$ et $v_{sommet} \in V_{sommet}$<br>Pour qu'un sommet $v_{arc}$ représentant un candidat arc $CandA$ appartienne à l'extension, il faut que l'ensemble de ses voisins appartiennent à l'extension. Ses voisins $v_{sommet}$ représentent forcément un candidat sommet $CandS$ composant du candidat arc $CandA$. |

D'après ces contraintes, nous pouvons observer qu'un sommet de type arc représentant un candidat arc a 1 ou 2 voisins. Dans le cas où le candidat arc représente une relation "rdfs :label" ou une relation avec un élément ontologique du module ontologique, alors le sommet associé du graphe $G_{ext}$ n'aura qu'un voisin. Si ce candidat arc représente une relation entre deux candidats sommets, alors le sommet associé du graphe $G_{ext}$ aura 2 voisins : le sujet et l'objet de la relation.

Pour simplifier le discours nous considérons dans la suite de ce rapport une extension comme étant un graphe d'extension.

D'après les exemples précédents, nous pouvons déduire les deux extensions présentées dans la figure 35. Dans cette figure, les carrés verts représentent des candidats sommets.

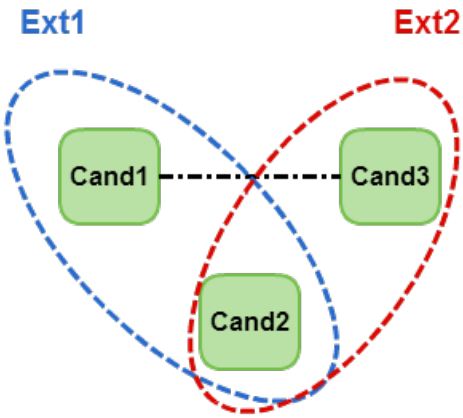


FIGURE 35 – Exemple de génération des extensions

Les identifiants de ces candidats ($Cand1$, $Cand2$ et $Cand3$) font référence aux mêmes candidats présentés dans les exemples précédents. Les liens entre les candidats définissent les incompatibilités entre candidats. Ces candidats sont donc regroupés en extensions représentées par des ovales pointillés entourant les candidats présents dans cette extension. Le lien entre $Cand1$ et $Cand3$ définit une incompatibilité. Nous définissons donc deux extensions $Ext1$ et $Ext2$, l'une considérant $Cand1$ et l'autre considérant $Cand3$. $Cand2$ est présent dans les deux extensions car il n'y a aucune incompatibilité avec d'autres candidats.

### 4.5.3. Algorithme de recherche de clique maximale

Considérant le graphe d'incompatibilité, nous générons son graphe complémentaire – qui ne lie donc que des candidats qui ne sont pas incompatibles, autrement dit des candidats compatibles. Nous cherchons à générer les cliques maximums du graphe de compatibilités pour obtenir les extensions. Nous retrouvons un problème connu en théorie des graphes : MCE (Maximum Clique Enumeration), qui est un problème NP-difficile. Chaque clique maximum est un sous-ensemble de candidats, tous compatibles entre-eux.

Pour résoudre ce problème de MCE, plusieurs algorithmes existent. Le plus courant est le Bron Kerbosch ([Bron and Kerbosch, 1973]).

Dans cet algorithme, R représente l'ensemble des sommets faisant partie de la clique en cours d'analyse, P est l'ensemble des sommets potentiels à ajouter à la clique, et X est l'ensemble des sommets déjà visités. Pour initialiser cet algorithme, il faut lancer l'appel avec R et X vides et P contenant tous les sommets du graphe étudié. La méthode "intersection" retourne les sommets communs entre la liste sur laquelle est appelée la méthode et la liste passée en paramètre. La méthode "N" permet d'obtenir les voisins du sommet passé en paramètre.

Considérons l'exemple présenté sur la figure 36. Cette figure présente un graphe non-orienté contenant six sommets. Nous considérons que la fonction "newClique" affiche le

```
BronKerbosch(R, P, X)
{
  if(P.isEmpty() && X.isEmpty())
  {
    newClique(R);
  }
  for( Vertex v in P)
  {
    BronKerbosch(R.add(v), P.intersection(N(v)), X.intersection(N(v)));
    P = P.remove(v);
    X = X.add({v});
  }
}
```

**Algorithm 4:** Algorithme de Bron-Kerbosch

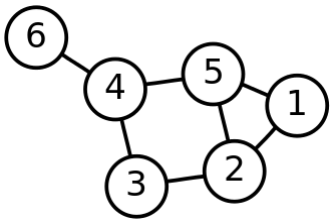


FIGURE 36 – Exemple d'un graphe non-orienté

résultat dans la console. Si nous appliquons l'algorithme de BronKerbosch sur ce graphe, nous obtenons la sortie suivante :

```
BronKerbosch(∅, {1,2,3,4,5,6}, ∅)
       BronKerbosch({2}, {1,3,5}, ∅)
           BronKerbosch({2,3}, ∅, ∅): output {2, 3}
           BronKerbosch({2,5}, {1}, ∅)
               BronKerbosch({1,2,5}, ∅, ∅): output {1,2,5}
       BronKerbosch({4}, {3,5,6}, ∅)
           BronKerbosch({3,4}, ∅, ∅): output {3,4}
           BronKerbosch({4,5}, ∅, ∅): output {4,5}
           BronKerbosch({4,6}, ∅, ∅): output {4,6}
       BronKerbosch({6}, ∅, {4}): no output
```

Le premier appel de l'algorithme place R et X à vide et tous les sommets dans P. P et X n'étant pas vides, un premier sommet est sélectionné, ici le sommet 2. Un appel récursif à la fonction est donc appelé avec le sommet 2 dans la liste R, tous les voisins de 2 (1, 3 et 5) dans la variable P et la liste X est toujours vide. Là encore, P et X ne sont pas vides, donc le sommet 3 est sélectionné et ajouté à l'ensemble X (pour considérer la clique 2, 3). L'ensemble des voisins de 3 (4 et 2) ne font pas partie de la liste P à cette étape ($\{4, 2\} \cap \{1, 3, 5\} = \emptyset$) donc l'ensemble vide est passé en paramètre pour la variable P. P et X se retrouvant vides à cette étape, la première clique est donc trouvée avec l'ensemble 2, 3. L'algorithme continue ensuite de la même façon pour découvrir toutes les cliques du graphe.

Il existe un certain nombre d'améliorations de cet algorithme, tel que l'algorithme de Tomita ([Tomita et al., 2006]) ou, plus récemment, l'algorithme de Eppstein et Strash ([Eppstein and Strash, 2011]). Ces algorithmes permettent d'améliorer un peu la complexité du problème en ajoutant un pivot pour l'algorithme de Tomita, et d'ordonner le parcours des nœuds dans P pour l'algorithme de Eppstein. Cependant, ces améliorations ne permettent pas de résoudre le problème du MCE en temps raisonnable. De par la complexité du problème, il est inenvisageable de générer toutes les cliques maximum possibles. Cette complexité recèle une combinatoire particulièrement élevée qui fait que le résultat de l'algorithme ne peut pas être atteint dans un temps raisonnable. Notre problème est donc de trouver un moyen d'obtenir l'extension dans ces conditions de complexité.

### 4.5.4. Algorithme supervisé (branch and bound)

Pour résoudre ce problème, nous utilisons un solveur CSP (Constraint Satisfaction Problem ou problème de satisfaction de contraintes en français). L'utilisation d'un solveur de ce type permet l'utilisation de la technique de Branch and Bound (séparation et évaluation en français) qui nous permettra de trouver une solution au problème en maximisant une fonction objective. Les solveurs CSP prennent en entrée un modèle d'un problème sur lequel ils appliquent l'algorithme de Branch and Bound.

Pour cela, nous allons d'abord présenter un modèle permettant de maximiser le nombre de candidats présents dans une extension. Nous cherchons alors l'extension la plus grande possible. Nous verrons ensuite comment maximiser la confiance des candidats d'une extension.

Nous avons défini le modèle suivant (exprimé en GMPL [84]) :

```
/* nbCands est le nombre de candidats*/
param nbCands > 0 integer;
/* cands est l'ensemble des candidats */
set cands := 1..nbCands;
/* incomp est la matrice stockant les incompatibilités entre candidats,
incomp[i][j]=1 si i et j sont incompatibles, 0 sinon*/
param incomp{ i in cands, j in cands};

/* ext est l'extension, ext[i]=1 si i est sélectionné dans l'extension
sinon ext[i]=0 */
var ext{i in cands},  binary;

/* Fonction objective */
maximize opti: sum{i in cands}ext[i];

/* Contraintes */
s.t.Cincomp{i in cands, j in cands}:incomp[i,j]*(ext[i]+ext[j])<=1;
end;
```

**Algorithm 5:** Modèle GMPL de recherche de l'extension maximale

Le modèle défini dans l'algorithme 5 cherche à maximiser la somme des éléments de $ext$. Cette variable est un tableau de booléens qui place à vrai la valeur à l'indice des candidats faisant partie de l'extension. Maximiser la somme des valeurs du tableau $ext$ revient à maximiser le nombre de candidats présents dans l'extension. La contrainte $Cincomp$ qui impose que : $incomp[i,j] * (ext[i] + ext[j]) <= 1$, définit que si il y a une incompatibilité entre le candidat $i$ et le candidat $j$ ($incomp[i,j] = 1$), alors $ext[i]$ et $ext[j]$ ne peuvent pas être tous les deux égaux à 1. En d'autres termes, ils ne peuvent pas tous les deux appartenir à l'extension s'ils sont incompatibles. En effet, s'ils sont incompatibles, alors $incomp[i,j] = 1$ et s'ils sont tous les deux dans l'extension, alors $ext[i] = ext[j] = 1$. Par conséquent, $incomp[i,j] * (ext[i] + ext[j]) = 2 > 1$. La contrainte $Cincomp$ n'étant pas respectée, l'extension n'est pas une solution possible.

Grâce au modèle de contrainte précédent, nous pouvons trouver une extension possible maximisant le nombre de candidats compatibles. Nous devons maintenant déterminer comment découvrir l'extension optimale qui contient des candidats validés par un expert. En effet, le modèle présenté précédemment permet d'obtenir une solution parmi toutes les

84. GNU Mathematical Programming Language - https://en.wikibooks.org/wiki/GLPK/GMPL_%28MathProg%29

solutions possibles, mais ce n'est pas forcément la meilleure solution. Certains candidats de l'extension peuvent être, en définitive, de mauvais candidats que nous ne souhaitons pas voir apparaître dans la base de connaissances finale. Il est donc nécessaire de passer par une phase de validation des candidats de l'extension afin de déterminer si l'extension est correcte ou non. Cette phase de validation des candidats peut être utile pour ajouter des contraintes au problème afin de converger vers une solution optimale, autrement dit, pour ce qui nous concerne, vers une extension dont tous les candidats sont corrects.

L'idée est de présenter à un expert les candidats d'une extension, un à un, pour validation. Si l'expert valide le candidat présenté, alors on peut ajouter la contrainte selon laquelle l'extension optimale doit contenir ce candidat. Inversement, si l'expert ne valide pas le candidat, alors on ajoute la contrainte selon laquelle l'extension optimale ne doit pas contenir ce candidat. Il suffit pour cela d'ajouter le paramètre $defined$ dans le modèle GMPL présenté avec l'algorithme 5 et d'ajouter la contrainte de nécessité d'appartenance à l'extension ou non suivant les valeurs de $defined$.

```
/* nbCands est le nombre de candidats*/
param nbCands>0 integer;
/* cands est l'ensemble des candidats */
set cands := 1..nbCands;
/* incomp est la matrice stockant les incompatibilités entre candidats,
incomp[i][j]=1 si i et j sont incompatibles, 0 sinon*/
param incomp{ i in cands, j in cands};
/* defined est un tableau stockant si un candidat doit appartenir
(defined[i]=1) ou non (defined[i]=0) ou a definir (defined[i]=-1) */
param defined{i in cands};

/* ext est l'extension, ext[i]=1 si i est sélectionné sinon ext[i]=0 */
var ext{i in cands},  binary;

/* Fonction objective */
maximize opti: sum{i in cands}ext[i];

/* Contraintes */
s.t.Cincomp{i in cands, j in cands}:incomp[i,j]*(ext[i]+ext[j])<=1;
s.t.Cdefined{i in cands: defined[i]!=-1}:ext[i] == defined[i];
end;
```

**Algorithm 6:** Modèle GMPL de recherche de l'extension maximale (avec le paramètre defined)

L'algorithme 6 présente le nouveau modèle prenant en considération cette spécificité. À l'indice du candidat associé dans le vecteur $defined$, la valeur doit être à 1 si le candidat est validé, 0 s'il n'est pas validé ou $-1$ si le candidat n'est pas encore validé ou invalidé. De cette manière, la contrainte $Cdefined$ permet d'imposer la valeur dans $ext$ si elle est

différente de $-1$. Par exemple en considérant trois candidats, considérons que nous avons le vecteur $defined$ suivant : 0, -1, -1. Dans ce cas, nous savons que le candidat 1 ne sera pas présent dans l'extension (puisque $defined[1] = 0$) et que les deux autres candidats sont à définir par le modèle.

À chaque candidat non validé, l'algorithme est relancé afin de trouver une nouvelle extension avec les nouvelles contraintes posées dans $defined$. L'extension est considérée comme optimale si tous les candidats ont été validés par l'expert.

Grâce à cette méthode, nous pouvons être assurés de ne présenter à l'expert qu'un minimum de candidats à valider en déduisant automatiquement que tous les candidats incompatibles avec un candidat validé ne peuvent pas apparaître dans l'extension optimale. Ceci permet de réduire le nombre d'interactions pour faciliter le travail d'un expert.

La fonction objective présentée ici ne permet pas de prendre en compte la qualité des candidats d'une extension. En effet, nous cherchons à maximiser le nombre de candidats, quels qu'ils soient. Nous avons défini précédemment trois fonctions de calcul de la confiance d'un candidat suivant son niveau de consensus. Nous pouvons donc prendre en compte ces scores de confiance et maximiser non plus le nombre de candidats mais la confiance des candidats. De ceci peut résulter l'obtention d'une extension plus petite mais avec des candidats de meilleure qualité. De ce fait, les candidats avec le plus haut score de confiance sont présentés en premier à l'expert. Si un candidat avec un haut score est incompatible avec un candidat avec un score de confiance plus faible, il est préférable de présenter en premier le candidat avec un haut score, puisqu'il a plus de chance d'être validé par l'expert. Cette validation va rejeter tous les candidats incompatibles pour le modèle utilisé. Il reste alors moins de candidats à valider par l'expert. Pour déterminer la nouvelle fonction objective, nous allons avoir besoin de reporter dans le graphe des extensions le score de confiance des candidats. Pour cela, nous définissons une nouvelle fonction $trust$ qui s'applique sur les sommets $v \in V_{ext}$ du graphe des extensions $G_{ext} = \{V_{ext}, E_{ext}, \Sigma V_{ext}, \Sigma E_{ext}, etiqV_{ext}\}$.

$trust : V_{ext} \times \mathbb{R}$ est une application qui, à tout sommet de $G_{ext}$, associe un score compris entre 0 et 1 qui est le score de confiance du candidat que le sommet représente. $\forall v \in V_{ext}$ alors $\exists Cand$ un candidat dans le graphe des sources alignées $SA$ tel que $etiqV(v) = URI(Cand)$, dans ce cas $trust(v) = trust(Cand)$.

La fonction $trust$ peut être l'une des trois fonctions définies dans la section 4.4 du chapitre III.

Pour calculer la fonction objective, nous allons définir tout d'abord une fonction de pondération des sommets de $G_{ext}$ représentant un des candidats sommets, c'est à dire $v_s \in V_{sommet}$. Cette fonction s'intitule confiance intrinsèque. Elle évalue pour chaque sommet $v \in V_{sommet}$ l'apport de confiance du candidat sommet représenté par $v$ et des autres candidats arcs qui ne sont liés qu'à lui, c'est à dire les candidats sommets labels et les candidats arcs qui n'ont que lui comme composant. Chacun de ces candidats arcs sont représentés dans l'extension par un sommet $v_a \in V_{arc}$ qui ont un unique voisin $v$.

Confiance intrinsèque

$\forall v_s \in V_{sommet}$ nous définissons
$trustIntr(v_s) = trust(v_s) + \sum_{\substack{v_a \in V_{arc}, \\ v_a \in voisinage(v_s), \\ arite(v_a)=1}} trust(v_{arc})$
avec $voisinage(v)$ une application qui retourne l'ensemble des sommets voisins de $v$ et $arite(v)$ une application qui retourne le nombre d'arêtes liées à $v$

Nous définissons ensuite la confiance de voisinage $trustNeighbour$ d'un couple de sommet $v1$, $v2$ de $V_{sommet}$ du graphe des extensions $G_{ext}$. Cette valeur représente la somme des confiances des candidats arcs ayant comme composants les candidats sommets représentés par $v1$ et $v2$, c'est à dire qu'à tout couple $v1$ et $v2$ dans $G_{ext}$, nous associons la confiance dans leurs sommets voisins représentant des candidats arcs à deux composants.

Confiance de voisinage

$trustNeighbour(v1, v2) = \sum_{\substack{v_a \in V_{arc}, \\ v_a \in voisinage(v1) \cap voisinage(v2)}} trust(v_{arc})$

Nous proposons une fonction objective pour maximiser la somme des confiances en utilisant ces fonctions de pondération. La fonction objective devient alors la somme des confiances intrinsèques des candidats sommets de l'extension et la somme des confiances de voisinage entre tous les couples de candidats sommets de l'extension.

Nous obtenons le modèle GMPL présenté dans l'algorithme 7.

Dans le modèle 7, nous pouvons observer l'apparition de deux paramètres supplémentaires : $trustIntr$ et $trustNeighbour$. Le premier paramètre $trustIntr$ représente le score intrinsèque de chaque candidat sommet considéré ici. Le paramètre $trustNeighbour$ est la confiance de voisinage entre les candidats sommets associés.

Par nécessité d'obtenir un modèle linéaire, il n'était pas possible d'ajouter la condition $ext[j]$ dans la fonction objective. Pour garder le problème linéaire et maintenir la maximisation de candidats partageant le plus de candidats arcs de bonne qualité entre eux, nous avons considéré dans $trustNeighbour[i, j]$ tous les candidats arcs de $i$ vers $j$ mais aussi dans l'autre sens. Par conséquent, $trustNeighbour[i, j] = trustNeighbour[j, i]$. De cette manière, si le candidat $i$ et le candidat $j$ sont présents dans l'extension, la valeur $trustNeighbour[i, j]$ sera comptabilisée deux fois, et donc la fonction objective s'en verra augmentée.

Le temps nécessaire pour la validation par l'expert des candidats est, dans le pire des cas, égal, en nombre d'interactions (validation d'un candidat), au nombre de candidats générés. En d'autres termes, dans le pire des cas, l'expert aura à valider tous les candidats s'ils sont tous compatibles les uns avec les autres. Dès qu'une incompatibilité apparaît, ce nombre d'interactions est forcément diminué.

Si nous prenons l'exemple présenté sur la figure 37, nous pouvons voir la prise en compte de la confiance dans la découverte d'une extension. Cet exemple se fonde sur

```
/* nbCands est le nombre de candidats*/
param nbCands>0 integer;
/* cands est l'ensemble des candidats */
set cands := 1..nbCands;
/* incomp est la matrice stockant les incompatibilités entre candidats,
incomp[i][j]=1 si i et j sont incompatibles, 0 sinon*/
param incomp{ i in cands, j in cands};
/* defined est un tableau stockant si un candidat doit appartenir
(defined[i]=1) ou non (defined[i]=0) ou à définir (defined[i]=-1) */
param defined{i in cands};
/* trustIntr est le vecteur donnant la somme des confiances intrinsèques
du candidat*/
param trustIntr{i in cands};
/* trustNeighbour est la matrice qui donne la somme des confiances des
candidats arcs reliant i et j */
param trustNeighbour{i in cands, j in cands};
/* ext est l'extension, ext[i]=1 si i est selectionne sinon ext[i]=0 */
var ext{i in cands},  binary;

/* Fonction objective */
maximize opti:
    sum{i in cands}(ext[i]*(trustIntr[i]+
            sum{j in cands}(trustNeighbour[i, j])));
/* Contraintes */
s.t.Cincomp{i in I, j in I}:conflict[i,j]*(x[i]+x[j])<=1;
s.t.Cdefined{i in I: defined[i]!=-1}:x[i] == defined[i];
end;
```

**Algorithm 7:** Modèle GMPL de recherche de l'extension maximale (avec considération des scores de confiance)

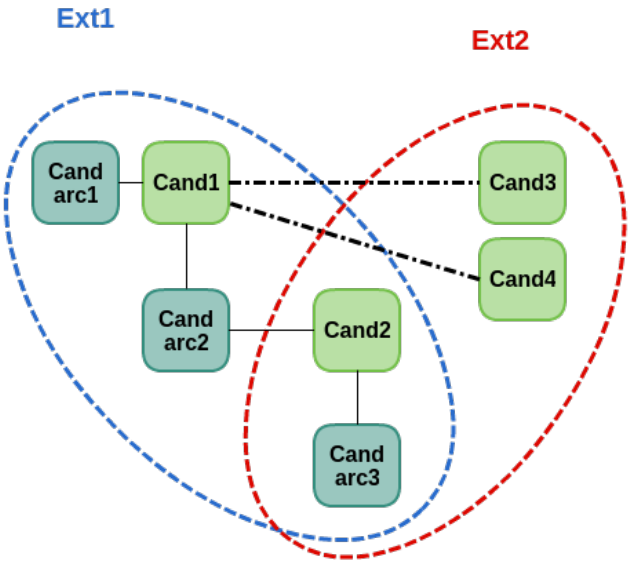


FIGURE 37 – Exemple d'extensions avec la prise en compte de la confiance

l'exemple présenté sur la figure 27 à la page 101. Nous avons présenté les différents candidats sommets et arcs en considérant l'incompatibilité entre le candidat sommet Cand3 (présenté sur la figure 34). Nous avons, pour l'exemple, ajouté un nouveau candidat sommet Cand4 avec une incompatibilité avec le candidat sommet Cand1. Nous avons ici deux extensions possibles. L'extension Ext1 contient deux candidats sommets (Cand1 et Cand2) alors que l'extension Ext2 contient trois candidats sommets (Cand2, Cand3 et Cand4). Donc si nous considérons la fonction objective qui ne considère que le nombre de candidats sommets, alors l'extension qui sera obtenue est l'extension Ext2.

Si nous prenons en compte les candidats arcs, nous pouvons en observer deux types. Le candidat Candarc1 (une relation "rdf :type" vers une classe du module) et le candidat Candarc3 (le label "Durum Wheat") sont associés à un seul candidat sommet. Ce n'est pas le cas pour le candidat arc Candarc2 (la relation "hasHigherRank" entre Cand2 et Cand1) qui associe deux candidats. C'est ici la différence entre les candidats arcs qui seront considérés dans le calcul de la confiance intrinsèque d'un candidat sommet ou non. Nous posons alors : $trustIntr(Cand1) = trust(Cand1) + trust(Candarc1)$ et $trustIntr(Cand2) = trust(Cand2) + trust(Candarc3)$. Le candidat arc Candarc2 n'est pas considéré dans le calcul d'une confiance intrinsèque d'un candidat sommet puisqu'il dépend de l'appartenance de deux candidats sommets dans l'extension pour être considéré. Si nous considérons le calcul de la fonction objective "opti" pour les deux extensions nous

obtenons :

$$\begin{aligned} opti(Ext1) &= trustIntr(Cand1) + trustIntr(Cand2) + trust(Candarc2) \\ &= trust(Cand1) + trust(Candarc1) + trust(Cand2) + trust(Candarc3) \\ &\quad + trust(Candarc2) \end{aligned} \tag{19}$$

$$\begin{aligned} opti(Ext2) &= trustIntr(Cand2) + trustIntr(Cand3) + trustIntr(Cand4) \\ &= trust(Cand2) + trust(Candarc3) + trust(Cand3) + trust(Cand4) \end{aligned} \tag{20}$$

Nous pouvons observer dans ces deux calculs que $trustIntr(Cand3) = trust(Cand3)$ et $trustIntr(Cand4) = trust(Cand4)$ puisque ces deux candidats n'ont pas de candidats arcs ne dépendant uniquement d'eux. Ensuite, nous voyons que la confiance du candidat arc Candarc2 n'est considérée que dans le calcul de la fonction objective de l'extension Ext1. Ceci provient du fait que le candidat arc Candarc2 dépend des candidats Cand2 et Cand1. Le candidat Cand1 n'étant pas dans l'extension Ext2, alors le candidat arc Candarc2 n'est pas considéré dans cette extension.

De cette manière nous pouvons obtenir l'extension qui maximise la confiance non seulement des candidats sommets présents dans ces extensions, mais aussi la confiance des candidats arcs considérés.

# 5. Mise à disposition des extensions sur le LOD

Afin de pouvoir rendre accessible sur le LOD l'extension finale découverte, il est nécessaire de définir un format d'export. Nous présentons ici deux formats pour exporter une extension. Le premier format permet l'export du graphe des extensions. Nous présenterons dans la section 5.1 notre spécialisation de l'ontologie PROV-O rendant possible cet export. Ensuite, nous présenterons dans la section 5.2 la façon d'exporter l'interprétation des candidats d'une extension pour permettre un export au format OWL.

## 5.1. Ontologie de provenance : PROV-O

Après la génération des candidats, le calcul de scores de confiance associés aux candidats et la découverte de l'extension optimale, la dernière étape est l'exportation des candidats. L'objectif n'est pas de fournir une base de connaissances directement exploitable par des applications du Web sémantique, mais de proposer une interprétation pouvant faciliter le travail de l'ontologue pour la création de cette base de connaissances. De ce fait, il est primordial, lors de l'export des candidats, de garder l'information sur la provenance de ces candidats et les scores de confiances associés.

Pour exporter ces candidats et leurs provenances nous utilisons l'ontologie de provenance créée par le W3C : PROV-O [85][Lebo et al., 2013].

85. http://www.w3.org/TR/prov-o/

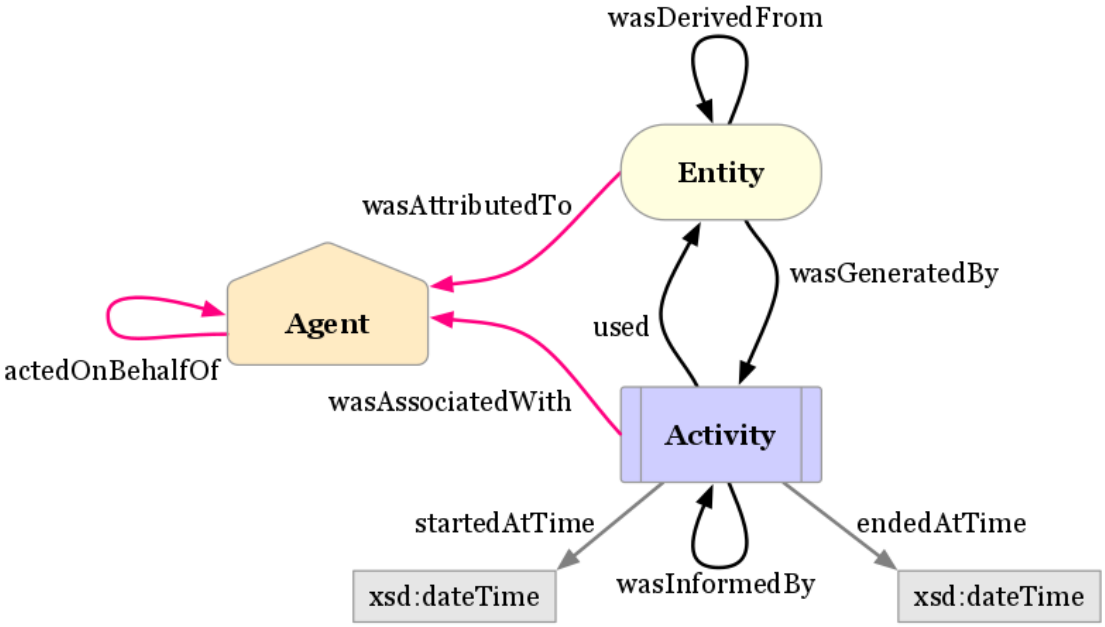


FIGURE 38 – Ontologie de provenance : PROV-O

La figure 38 présente la structure de base de cette ontologie. Cette ontologie est fondée sur trois composantes principales : Agent, Entity et Activity. De cette manière, il est possible de représenter un agent réalisant une activité pour générer et/ou utiliser une entité. Si nous souhaitons définir plus finement la provenance des éléments, d'autres classes sont définies dans l'ontologie de provenance, comme le montre la figure 39.

Ce qui nous intéresse particulièrement dans cette version détaillée de l'ontologie de provenance PROV-O c'est l'apparition des Collections. Une collection telle que définie dans l'ontologie PROV-O permet la représentation d'un ensemble d'entités. Nous pouvons alors représenter les bases de connaissances sources comme étant des Collection PROV-O contenant des éléments ontologiques. La base de connaissances finale résultant de notre méthodologie peut être modélisée sous la forme d'une Collection PROV-O contenant des candidats.

Il est possible de spécialiser l'ontologie de provenance en exploitant ces collections pour faire une représentation de nos candidats et de leurs provenances. Cette spécialisation est présentée dans la figure 40. Nous spécialisons "SoftwareAgent" (qui est lui même une spécialisation de "Agent") avec notre système "Muskca". Muskca est le système que nous proposons dans ce manuscrit. Il est possible de créer plusieurs instances de "Muskca" pour définir les différentes versions du système. L'activité de "Fusion" est définie comme une spécialisation de "Activity" de PROV-O. Une activité de Fusion est associée à une date de début et une date de fin, en utilisant respectivement les propriétés "prov :startedAtTime" et "prov :endedAtTime". Par soucis de clarté du schéma, nous n'avons pas représenté ces

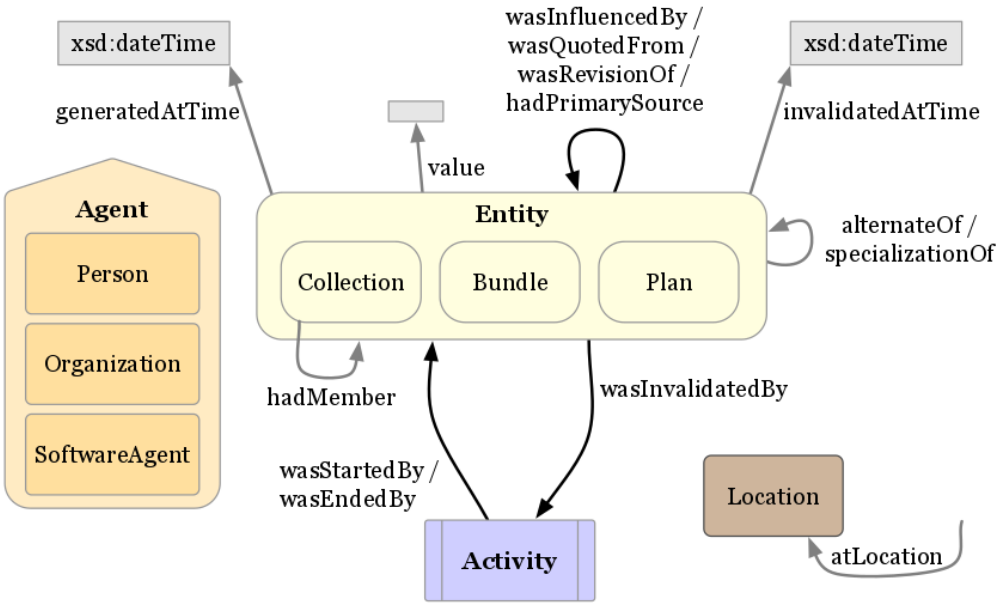


FIGURE 39 – Ontologie de provenance : PROV-O (détaillée)

propriétés. Cette activité utilise des SKB (bases de connaissances sources) pour générer une KB finale, qui sont toutes deux des "Collection". Une "Collection", telle que définie dans PROV-O, est une "Entity" pouvant contenir d'autres "Entity". Les SKB contiennent un ensemble d'éléments, qui peuvent être des éléments ontologiques ou des relations. Les KB finales contiennent un ensemble de candidats, pouvant être des candidats sommets ou des candidats arcs. Les éléments et les candidats sont des spécialisations de "Entity" dans PROV-O. Nous gardons l'information de la provenance des candidats en faisant le lien vers les éléments des SKB utilisées pour générer ces candidats. Chaque candidat est aussi associé à un score de confiance avec la relation "hadTrustScore". Cette propriété est une spécialisation de la propriété "prov :value" (non représentée ici par soucis de clarté). Il existe une spécialisation de "KB finale" qui est "Extension". Ceci permet de représenter une extension, qui est un sous-ensemble de tous les candidats générés.

La représentation des relations et des candidats arcs repose sur la définition de triplets. En effet, un individu de type "Relation" tel que défini dans cette spécialisation de PROV-O est un triplet RDF. Cela signifie que nous souhaitons représenter le fait qu'un triplet RDF est une instanciation d'une classe spécifique (ici "Relation"). RDF permet d'effectuer ce genre de manipulations en utilisant la notion de réification, qui utilise la classe "Statement" du vocabulaire RDF.

La figure 41 présente un exemple de l'utilisation de Statement. En haut de la figure se trouve le triplet représentant la relation "hasDirectHigherRank" entre "Triticum Durum" et "Triticum". Le graphe en bas de la figure présente la même relation avec l'utilisation de la classe Statement. Nous pouvons remarquer que ce triplet dispose maintenant d'une URI "Cand_Relation_1". Cette URI provient de l'étiquette attribuée au candidat

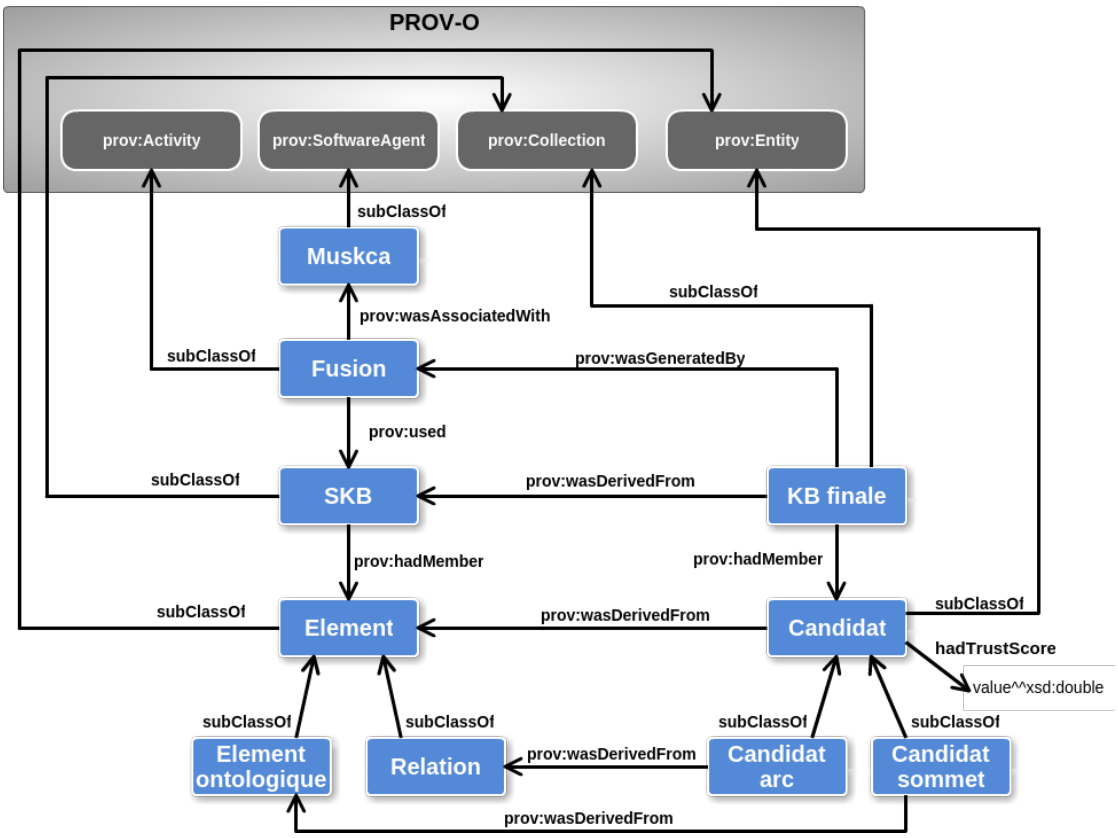


FIGURE 40 – Spécialisation de l'ontologie de provenance PROV-O

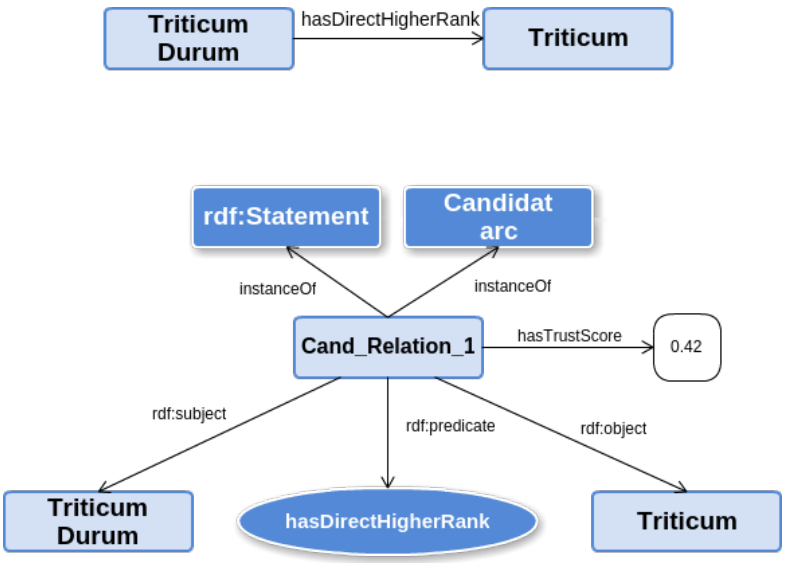


FIGURE 41 – Exemple de l'utilisation de la classe rdf :Statement

avec la fonction $URI(Cand)$. Ce triplet est donc du type "Candidat arc" défini dans notre spécialisation de l'ontologie PROV-O (Cf. figure 40). Puisque ce triplet devient un candidat, il est alors possible d'attribuer un score de confiance au triplet (ici 0,42). Cette affectation n'était pas possible avec la représentation classique d'un triplet RDF puisque l'utilisation d'une relation de type propriété de données a forcément pour sujet une URI. Il faut donc passer par la déclaration d'un individu "Statement" pour avoir l'URI d'un triplet RDF. De la même façon que nous avons défini le score de confiance de ce candidat, il est possible de définir la provenance de ce candidat avec les triplets provenant des sources du processus.

## 5.2. Export OWL

Un problème dans l'utilisation de la classe Statement est que le serveur d'interrogation de triplets RDF ne permet pas l'interrogation des individus de Statement de manière *transparente*. Prenons par exemple la requête SPARQL suivante :

```
SELECT ?taxon WHERE
{
    :Triticum_Durum :hasDirectHigherRank ?taxon.
}
```

Cette requête permet de chercher tous les taxons reliés à "Triticum Durum" par la propriété "hasDirectHigherRank". Si nous interrogeons le premier graphe de la figure 41, nous obtenons bien la réponse "Triticum". Si nous interrogeons le deuxième graphe de cette même figure, aucune réponse n'est retournée. Il faut pour cela passer par la

déclaration d'un objet "Statement" comme défini dans la figure 41. Si nous voulons une requête SPARQL pour interroger le deuxième graphe de cette figure alors nous obtenons :

```
SELECT ?taxon WHERE
{
    ?statement rdf:type rdf:Statement.
    ?statement rdf:subject :Triticum_Durum.
    ?statement rdf:predicate :hasDirectHigherRank.
    ?statement rdf:object ?taxon.
}
```

Comme nous souhaitons pouvoir générer une base de connaissances finale exploitable directement par des outils du Web sémantique, cette représentation est trop contraignante pour être suffisamment générique. Il faudrait pour cela intégrer, pour chaque utilisation de la base de connaissances finale, une étape de réécriture de requête pour interroger les candidats.

C'est pour cette raison que nous ajoutons une dernière étape d'exportation de la base de connaissances finale sans l'utilisation des individus de type "Statement". Cette étape permet d'extraire tous les candidats de la base de connaissances finale et d'en générer les représentations standard (sans "Statement"). Par conséquent cet export ne contient plus d'information concernant la provenance, ni concernant les candidats. Néanmoins la provenance des éléments ontologiques générés à partir des candidats sommets est représentée par des relations "owl :sameAs". Ces relations permettent d'identifier l'équivalence sémantique entre des éléments de l'export et des éléments des sources considérées. La définition de ces équivalences sémantiques permet une insertion dans le LOD (Cf. section 1.3 du chapitre I). Elle permet néanmoins une interrogation directe par les différents outils du Web sémantique.

La base de connaissances finale et l'exportation des candidats sont deux bases de connaissances qui se complètent l'une l'autre. En effet, les URI générées pour représenter les candidats sommets sont les mêmes entre les deux formats. Par conséquent, lorsque nous souhaitons obtenir la provenance d'un élément ontologique de l'export, nous pouvons interroger les candidats de la base de connaissances en réutilisant le même URI. Ces deux sorties peuvent même cohabiter sur un même serveur d'interrogation de triplets RDF. Néanmoins, il est nécessaire de maintenir à jour ces deux sorties. Par exemple si une relation est modifiée, il faut penser à la modifier aussi dans l'autre sortie.

Ces formats de représentation de la base de connaissances finale permettent une réutilisation flexible de celle-ci. Il est important de maintenir les informations concernant la provenance des candidats générés, mais il est aussi intéressant de pouvoir utiliser les outils du Web sémantique sur ce résultat. Nous proposons ici la possibilité d'exploiter ces deux formats.

## 6. Conclusion

Nous avons présenté dans cette section notre proposition d'une méthodologie permettant la transformation de plusieurs sources simultanément. Cette méthodologie considère dans un premier temps une transformation automatique des sources d'après la définition d'un modèle commun. Ce modèle commun, représenté par la partie haute d'un module ontologique, permet d'harmoniser la modélisation mais aussi de définir les bornes du domaine d'étude. Les sources transformées sont ensuite alignées et des candidats sont générés. Ces candidats sont composés d'éléments provenant d'une ou plusieurs sources en fonction des correspondances découvertes. Pour chaque candidat est calculé un score de confiance permettant la représentation de la confiance consensuelle que nous pouvons attribuer à ce candidat. Les candidats partageant un même élément sont considérés comme incompatibles. En d'autres termes, ils ne peuvent pas apparaître dans la même base de connaissances finale. Cette incompatibilité découle de notre hypothèse selon laquelle nous considérons les correspondances complexes comme des erreurs du système d'alignement. Nous générons alors une extension qui est le sous-ensemble de candidats qui maximise la somme des confiances. Nous mettons ce résultat à disposition sur le LOD grâce à deux formats d'exportation. Le premier permet la représentation des candidats et de leur provenance en utilisant l'ontologie PROV-O. Le second permet une exportation plus adaptée aux outils du Web sémantique.

# Quatrième partie . Implémentation

Notre proposition a été implémentée dans un prototype que nous avons appelé "Muskca". Ce nom est un acronyme signifiant MUlti Sources Knowledge CAndidate. Il a été développé en utilisant le langage Java [86]. Le choix s'est porté sur ce langage pour sa portabilité, quel que soit le système d'exploitation (Linux, OSX, Windows, ...).

Le prototype Muskca est accessible sur le dépôt GitHub suivant : https://github.com/Murloc6/Muskca. Il est possible de récupérer directement l'exécutable à l'adresse suivante : https://github.com/Murloc6/Muskca/releases/latest.

Nous pouvons observer sur la figure 42 le flux des données entre la récupération des sources et la génération de la base de connaissances finale. Nous avons détaillé les flux entrants et sortants de Muskca. N'apparaissent sur cette figure que les interactions avec des outils extérieurs à Muskca. Ces interactions sont composées de 6 processus :

1. **Transformation des sources :** Ce processus, décrit dans la section 3.2 du chapitre III, permet de transformer les sources en bases de connaissances sources (SKB).
2. **Stockage des SKB :** Les SKB générées à partir des sources sont stockées sur un serveur de gestion de données RDF (SPARQL Endpoint). Nous utilisons Fuseki server.
3. **Récupération des SKB :** Les SKB sont récupérées dans Muskca pour être fusionnées.
4. **Alignements des SKB :** Pour chaque paire de SKB, un alignement est généré. Pour cela, la plate-forme Seals est utilisée pour faire l'interface avec le système d'alignement LogMap. Ces alignements entre les SKB seront utilisés pour générer les candidats.
5. **Génération d'extensions :** En réutilisant les candidats obtenus et les incompatibilités entre ces candidats, nous générons une extension en utilisant le solveur Choco. Ce processus implique une intervention humaine lors de la validation des candidats pour la découverte de l'extension optimale.
6. **Génération de la KB finale :** Une fois la base de connaissances finale obtenue, nous l'exportons vers le serveur Fuseki et dans un fichier.

L'expert impliqué dans le processus intervient trois fois. Il participe tout d'abord à la création du module ontologique qui sera utilisé dans Muskca. Il intervient ensuite pour définir les patrons de transformation qui seront utilisés pour transformer les SKB en fonction du module ontologique. Il intervient enfin durant le processus 5 pour valider les candidats afin de générer l'extension optimale.

Nous reviendrons dans cette section sur quatre aspects importants de cette implémentation :

86. https://www.java.com/fr/

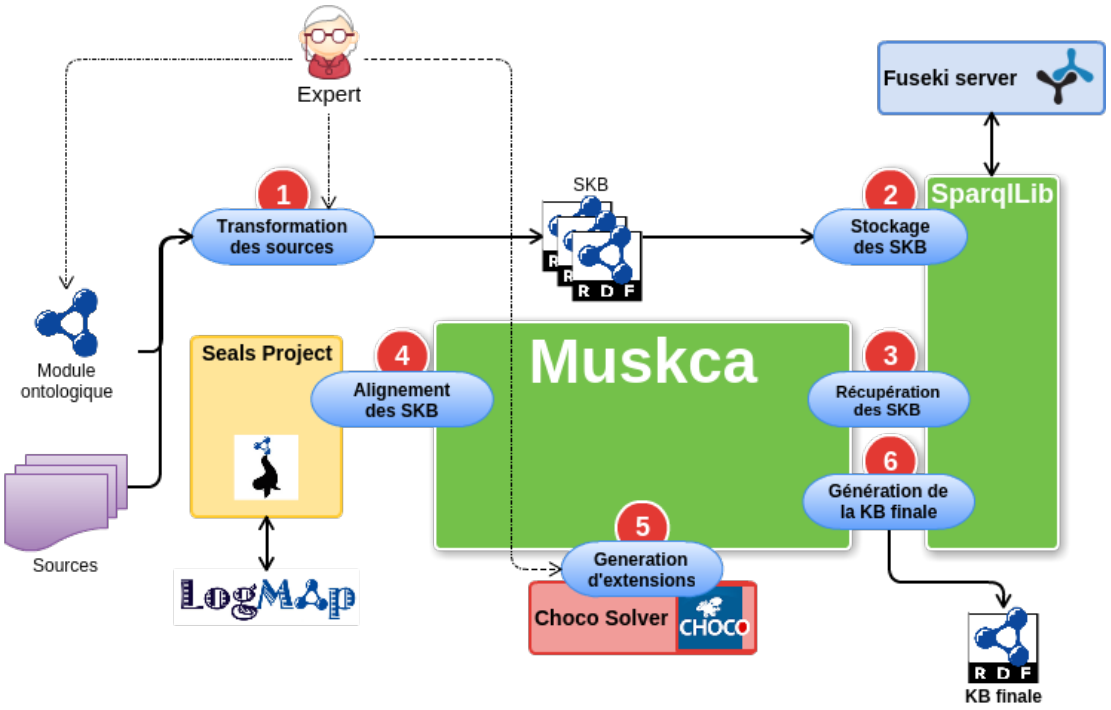


FIGURE 42 – Flux des données

— Tout d'abord nous aborderons nos motivations pour l'utilisation d'un serveur de gestion de données RDF (Fuseki) ainsi que notre façon de l'utiliser.
— Nous aborderons ensuite les avantages liés à l'utilisation de la plate-forme Seals pour interfacer LogMap, puis nous verrons comment l'adapter à un autre système d'alignement.
— Nous avons pu observer dans la section 4.5.4 du chapitre III que nous utilisons un modèle de programmation par contraintes pour découvrir l'extension maximisant une fonction objective. Nous verrons ici pourquoi nous avons utilisé le solveur Choco pour l'implémentation d'un tel modèle.
— Enfin, nous aborderons la modélisation que nous avons choisie pour l'implémentation de Muskca et les différents points permettant une amélioration et une gestion simplifiée de l'outil.

## 1. Serveur de données RDF (SPARQL endpoint)

Un SPARQL endpoint est un serveur permettant de gérer des données au format RDF. Il permet de stocker des données sous forme de triplets RDF, de les modifier et de les récupérer. Pour cela, un tel serveur est manipulable par le biais de requêtes exprimées en SPARQL (Cf. section 1.4). D'après la recommandation du W3C "SPARQL 1.1 Graph Store HTTP Protocol" [87], un serveur de gestion de données RDF (SPARQL endpoint) est interrogeable en utilisant le protocole HTTP. Des formats de requêtes HTTP spécifiques sont définis afin de pouvoir interagir avec le SPARQL endpoint. Il est par exemple possible d'interroger les données à travers une requête GET en passant la requête SPARQL en paramètre. De plus, ce type de serveurs permet de manipuler plusieurs jeux de données sur le même serveur. Pour cela, il faut spécifier dans l'URL de la requête GET le jeu de données à interroger.

L'exemple de la figure 43 est une URL permettant d'interroger un SPARQL endpoint, qui se décompose de la manière suivante :

**Serveur :** L'adresse du SPARQL endpoint auquel est soumise la requête.

**Jeu de données :** Le nom du jeu de données à interroger.

**Type de requête :** Le type de requête à exécuter, ce qui permet de différencier les ajouts, les suppressions ou les modifications. Ici "query" spécifie que nous souhaitons interroger la base.

**Requête :** Le paramètre "query" de la requête HTTP permet de spécifier la requête SPARQL à exécuter. Dans cet exemple, la requête envoyée est “SELECT DISTINCT ?Class WHERE [] a ?Class LIMIT 100”.

**Format de retour :** Le paramètre "format" permet de spécifier le format dans lequel nous souhaitons récupérer les données. Ici le format demandé est JSON, mais le résultat aurait pu être renvoyé en RDF-XML ou Turtle-RDF par exemple.

87. http://www.w3.org/TR/2013/REC-SPARQL11-http-rdf-update-20130321/

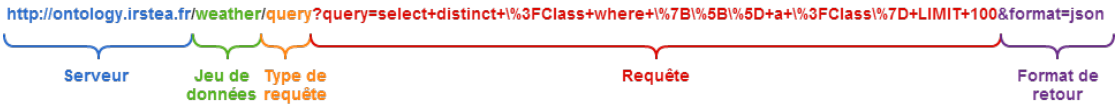


FIGURE 43 – Exemple de requête HTTP sur un SPARQL endpoint

Le jeu de données "weather" sur le SPARQL endpoint "http ://ontology.irstea.fr" contient des données provenant de capteurs météorologiques implantés dans une ferme expérimentale de l'Irstea [88]. La requête présentée ici permet d'obtenir les 100 premiers résultats de la liste des classes ayant au moins une instanciation dans ce jeu de données.

## 1.1. L'utilisation de Fuseki

Comme nous l'avons vu sur la figure 42, l'interaction avec le SPARQL endpoint intervient sur le processus "Stockage des SKB", "Récupération des SKB" et "Génération de la KB finale". Dans le premier processus, nous avons besoin d'ajouter de nouveaux éléments dans un jeu de données, dans le second de récupérer les éléments qui nous intéressent et enfin, dans le dernier, de générer un fichier à partir d'un jeu de données spécifique. Nous pouvons préciser que la recommandation du W3C sur l'interface en HTTP d'un SPARQL endpoint propose une requête spécifique pour récupérer l'intégralité d'un jeu de données. Nous utilisons cette fonctionnalité pour générer le fichier contenant la base de connaissances finale.

Nous avons fait ce choix pour quatre raisons :

1. **La pérennité de Muskca :** Le fait de ne pas contraindre l'utilisation d'une librairie spécifique de manipulation de données RDF facilite la maintenance et les développements futurs. L'interfaçage avec un SPARQL endpoint étant issu d'une recommandation du W3C, il est facile de changer le SPARQL endpoint sans avoir à modifier Muskca. La manipulation de gros volumes de données RDF étant particulièrement complexe, les système pour mettre en place un SPARQL endpoint sont en constante évolution pour améliorer les temps de réponse. Nous pouvons alors exploiter ces améliorations et mettre à jour le SPARQL endpoint sans modifier Muskca.
2. **La simplicité de manipulation :** Ce deuxième argument provient de la complexité de manipulation d'une base de connaissances en utilisant l'Ontology API d'Apache Jena [89]. Il est plus simple d'écrire une requête SPARQL plutôt qu'un algorithme de parcours de la base de connaissances. L'utilisation d'un SPARQL endpoint permet aussi de s'assurer que l'on applique des algorithmes de parcours déjà optimisés.
3. **La flexibilité de Muskca :** Utiliser comme entrée de notre outil l'adresse d'un SPARQL endpoint nous permet de décentraliser les exécutions. Nous pouvons alors

88. http://ontology.irstea.fr/pmwiki.php/Site/WeatherData
89. https://jena.apache.org/documentation/ontology/

lancer Muskca sur un ordinateur tout en ayant le SPARQL endpoint sur un serveur distant. De plus, nous pouvons imaginer l'utilisation de Muska sur des données provenant du Web de données liées qui sont, pour la plupart, accessibles par un SPARQL endpoint spécifique.

4. **La cohérence avec les outils du Web sémantique :** Le choix d'utiliser un SPARQL endpoint et non un autre système de stockage (tel que NoSQL) provient de notre volonté d'intégrer Muskca dans les technologies du Web sémantique. Les jeux de données du Web de données liées sont, très souvent, accessibles par des SPARQL endpoints. Muskca permet de proposer une exportation directement dans un SPARQL endpoint. Cette exportation concerne la base de connaissances finale, mais également les SKB générées. D'autres outils prévus pour les technologies du Web sémantique deviennent alors facilement utilisables.

Afin de permettre de tester notre application, nous avons mis en place le SPARQL endpoint Fuseki qui fait partie de l'ensemble des outils d'Apache Jena [90]. Nous avons fait ce choix car les outils provenant d'Apache Jena sont réputés pour leur fiabilité. De plus, l'installation et la configuration d'un serveur Fuseki sont particulièrement simples. Cette simplicité d'installation a un prix puisque, d'après des études [91], ce n'est pas le SPARQL endpoint le plus rapide existant. Néanmoins, le temps de réponse nécessaire à Fuseki lors de nos expérimentations n'a pas été un problème.

## 1.2. SparqlLib

Comme nous venons de le décrire, nous utilisons un SPARQL endpoint pour manipuler des données RDF dans notre outil Muskca. Puisque nous avons été amenés à manipuler plusieurs jeux de données, pouvant être sur plusieurs serveurs différents et avec différents types de requêtes SPARQL, il a été nécessaire de développer une librairie permettant de faciliter ces manipulations : SparqlLib [92]. Cette librairie se présente sous la forme d'un projet développé en Java. Elle se concentre sur deux aspects nécessaires à la manipulation de SPARQL endpoints. Le premier aspect est la simplification d'accès à plusieurs serveurs et plusieurs jeux de données simultanément. Le deuxième aspect est la simplification pour l'écriture de requêtes SPARQL.

### 1.2.1. Modélisation du projet SparqlLib

Nous pouvons observer sur la figure 44 la présence de deux paquetages principaux : Query et Proxy.

Le paquetage Proxy ne contient qu'une seule classe : SparqlProxy. Cette classe permet de faire l'interface avec un jeu de données spécifique d'un SPARQL endpoint. Nous pouvons créer autant de SparqlProxy que nous avons de jeux de données à manipuler, chaque SPARQL proxy pouvant être localisé sur un serveur différent. Cette classe

90. https://jena.apache.org/documentation/fuseki2/index.html
91. http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinSPARQLbenchmark/results/V7/
92. https://github.com/NSeydoux/SparqlLib/

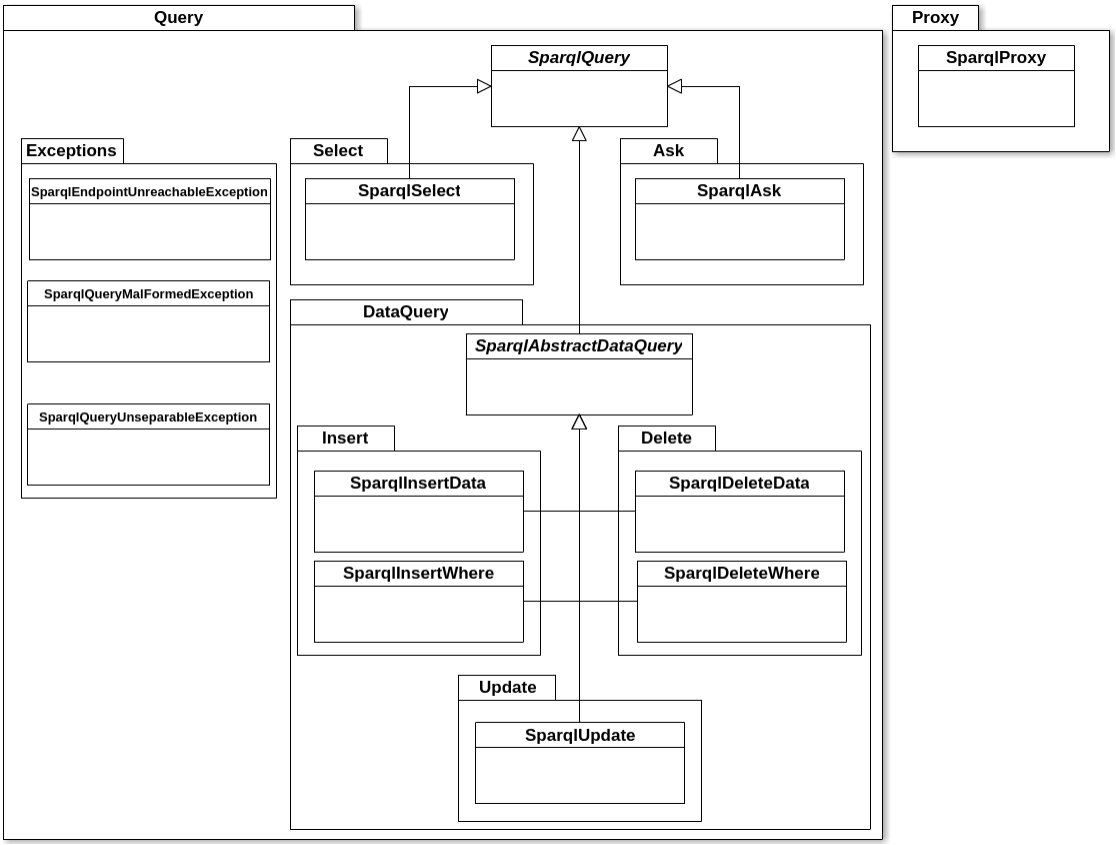


FIGURE 44 – Diagramme de classes SparqlLib

permettra l'envoi de requêtes directement sur le SPARQL endpoint associé. Pour cela, elle implémente la création de la requête HTTP associée, l'envoi de cette requête, la récupération et le traitement du résultat.

Le paquetage Query est constitué de plusieurs sous-paquetages et d'une classe principale, SPARQLQuery. Cette classe est la représentation abstraite d'une requête SPARQL. Elle est ensuite spécialisée en plusieurs sous-requêtes de type "Select", "Ask" ou "DataQuery". Les requêtes de type "Select" sont utilisées pour récupérer des triplets. Les requêtes de type "Ask" sont utilisées pour interroger l'existence de triplets. Les requêtes de type "DataQuery" sont utilisées pour modifier des triplets. Il est possible d'effectuer trois types de modifications : "Insert", "Delete" ou "Update". Dans le cas des insertions ("Insert") et des suppressions ("Delete") il est possible d'ajouter une condition en utilisant les classes finissant par "Where". Le sous-paquetage "Exceptions" regroupe les différentes exceptions pouvant intervenir lors de l'interrogation d'un SPARQL endpoint.

Il est donc possible d'utiliser une requête définie grâce aux classes présentes dans le package Query. De cette manière, nous pouvons faire abstraction de la notion de syntaxe d'une requête SPARQL lors de sa création puisqu'elle sera créée automatiquement. Un SparqlProxy pourra prendre en paramètre une requête de type SPARQLQuery pour interroger ou modifier la base.

Le projet SparqlLib a pour but de simplifier les interactions qui ont été nécessaires dans Muskca et dans d'autres projets utilisant un SPARQL endpoint. Il n'est ni exhaustif, ni parfait. Il nécessite encore diverses améliorations pour devenir utilisable par des personnes qui ne sont pas familières avec le langage SPARQL. Ce projet a été développé en collaboration avec Nicolas Seydoux, stagiaire dans l'équipe MELODI au moment du développement.

### 1.2.2. Exemple d'utilisation

Nous allons maintenant présenter un exemple d'utilisation du projet SparqlLib pour interroger un SPARQL endpoint. Prenons comme exemple la requête SPARQL suivante :

```
SELECT ?a ?b ?c
WHERE
{
    ?a ?b ?c.
}
LIMIT 10
```

Cette requête est très basique puisqu'elle permet d'obtenir la liste des 10 premiers triplets récupérés, sans condition sur ces triplets. Les SPARQL endpoints ne garantissant pas l'ordre des réponses, sauf si un ordre est stipulé dans la requête. Les 10 premiers triplets d'une même base peuvent donc être différents lors de plusieurs exécutions. Cette requête est souvent utilisée pour vérifier si la base contient des triplets.

Le code 8 présente la création et l'éxécution de la requête présentée précédemment. Les lignes 3 et 4 présentent la création d'un SparqlProxy pour le SPARQL endpoint : "http ://SPARQL.endpoint.url/". Les lignes 5 et 6 présentent la création de la requête.

```java
public static void main(String[] args)
{
    SparqlProxy spIn =
        SparqlProxy.getSparqlProxy("http://SPARQL.endpoint.url/");
    SparqlSelect query = new SparqlSelect("?a ?b ?c", "?a ?b ?c.");
    query.setLimit(10);
    try
    {
        for(JsonNode jnode : spIn.getResponse(query))
        {
            System.out.println("new triple : ");
            System.out.println("\t subject : "+
                jnode.get("a").get("value").asText());
            System.out.println("\t predicate : "+
                jnode.get("b").get("value").asText());
            System.out.println("\t object : "+
                jnode.get("c").get("value").asText());
        }
    }
    catch (SparqlQueryMalFormedException ex)
    {
        System.err.println("Query mal formed ...");
    }
    catch (SparqlEndpointUnreachableException ex)
    {
        System.err.println("SPARQL endpoint unreachable ...");
    }
}
```

**Algorithm 8:** Exemple de l'utilisation de SparqlLib

Nous définissons ici une requête de type "SparqlSelect" dont le constructeur nécessite en premier argument les variables de la requête et en deuxième argument le contenu de la clause WHERE de la requête. La ligne 6 précise la limite pour n'obtenir que les 10 premiers triplets. Les lignes 9 à 18 présentent le traitement du résultat. La requête est exécutée sur le SPARQL endpoint à la ligne 9, grâce à l'instruction : "spIn.getResponse(query)". La réponse est un tableau (de type ArrayList) de JsonNode. Un JsonNode est un objet JSON représenté grâce à la librairie Jackson [93]. La ligne 9 permet donc de parcourir tous les résultats retournés par la requête. Pour chaque résultat, le triplet obtenu est affiché. Nous observons qu'il est possible de récupérer la valeur de la variable "a" grâce à l'instruction :

```
jnode.get("a").get("value").asText();
```

D'un résultat ("jnode"), nous récupérons l'objet JSON associé à la clef du nom de la variable (ici "a") avec la méthode : "jnode.get("a"). Le format de retour JSON recommandé par le W3C propose de retourner non seulement la valeur de la variable mais aussi son type s'il existe. Nous traitons les résultats en considérant que tous les résultats sont de type chaîne de caractères. C'est pour cela que nous récupérons la valeur de la variable avec la méthode "asText()". Néanmoins, nous aurions pu ajouter un traitement du type de variable pour appliquer un traitement spécifique suivant les types attendus. De cette manière, nous aurions pu obtenir une variable de type entier si le résultat retourne une variable de ce même type.

## 2. Interface Seals

Comme nous l'avons vu dans le processus de fusion, il est nécessaire de pouvoir aligner chaque paire de SKB (Cf. section 4.2 du chapitre III). Ces alignements permettent ensuite de générer les candidats sommets. Pour cela, nous avons montré dans l'état de l'art (Cf. section 2 du chapitre II) que le système disponible le plus adapté à nos besoins est LogMap. Ce projet propose une interface en Java permettant l'utilisation de ce système d'alignement dans un projet [94]. Les systèmes d'alignements sont en constante évolution. Nous pouvons notamment citer la campagne d'évaluation de l'OAEI sur laquelle nous nous sommes appuyés pour nos analyses des systèmes d'alignements. Cette campagne propose chaque année de comparer les différents systèmes d'alignements existants avec des résultats différents tous les ans. Il est possible qu'à l'avenir, le système LogMap ne soit plus le système le mieux adapté à nos besoins. Il est donc souhaitable de proposer un moyen de changer simplement de système d'alignement. C'est dans cette optique que nous avons réutilisé le projet Seals qui est une interface permettant une utilisation transparente d'un système d'alignement.

La campagne de l'OAEI impose que chaque système participant utilise le projet Seals [95]. Ce projet propose une interface et une organisation particulière des systèmes d'alignement,

93. https://github.com/FasterXML/jackson
94. https://code.google.com/p/logmap-matcher/wiki/LogMapFromApplications
95. http://oaei.ontologymatching.org/2014/seals-eval.html

permettant leur utilisation uniformisée. Nous avons donc utilisé ce projet dans Muskca. De cette manière, nous pouvons utiliser un système d'alignement à travers le projet Seals. Ainsi, Muskca pourra facilement s'adapter à tous les systèmes d'alignement évalués par l'OAEI.

Prenons l'exemple présenté dans l'algorithme 9 permettant d'aligner deux bases de connaissances.

```
1 File f1 = new File("out/temp/"+fileNameS1);
2 File f2 = new File("out/temp/"+fileNameS2);
3
4 OClient c = new OClient("logmap2");
5 URL ret = c.align(f1.toURL(), f2.toURL());
6
7 RDFAlignReader reader = new RDFAlignReader(ret);
8 Set<MappingObjectStr> mappings = reader.getMappingObjects();
```

**Algorithm 9:** Utilisation de l'interface Seals

Le code présenté dans l'algorithme 9 implémente l'utilisation du projet Seals. Nous utilisons ici la classe OClient présente dans l'API proposée par Seals. Le constructeur de cette classe a besoin d'une chaîne de caractères représentant le dossier dans lequel est présent le système d'alignement que nous souhaitons utiliser ; nous utilisons "logmap2". Dans ce dossier doivent être présents les fichiers permettant l'exécution du système d'alignement en question. Ils doivent respecter une organisation particulière spécifiée par Seals. Le détail de cette organisation est présenté ici : http://oaei.ontologymatching.org/2011.5/tutorial/tutorialv3.pdf. La classe "OClient" s'appuie sur la valeur d'une variable d'environnement pour savoir où sont stockés les systèmes d'alignement disponibles sur le disque dur. Nous exécutons alors Seals dans un sous-processus pour lui passer la valeur de cette variable. Le projet SealsAligner.jar implémente l'utilisation de la librairie Seals pour aligner les sources. Une fois le sous-processus terminé, nous récupérons un fichier contenant les alignements entre les deux bases de connaissances. Ce fichier est au format RDFAlignement [96]. Nous pouvons parcourir ce fichier pour récupérer les alignements associés.

Nous voyons dans l'algorithme 9 qu'il est très simple de changer le système d'alignement. En effet, il suffit d'ajouter le dossier contenant l'implémentation du nouveau système d'alignement au format Seals dans le projet Muskca et changer le paramètre du constructeur "OClient". Pour simplifier encore plus cette transition, nous avons défini cette variable comme paramètre de Muskca. De cette manière, il est possible de changer de système d'alignement sans toucher au code source de Muskca.

Un autre fait notable sur l'utilisation de cette interface est la récupération des bases de connaissances à aligner. En effet, nous pouvons voir dans l'algorithme que les deux bases de connaissances sont récupérées comme deux fichiers. Comme nous l'avons montré précédemment, nous utilisons un Sparql endpoint pour stocker les SKB à aligner. Néanmoins,

96. http://alignapi.gforge.inria.fr/format.html

nous avons aussi vu qu'il est possible de récupérer tout le contenu d'un jeu de données contenu dans un Sparql endpoint par une requête spécifique. Nous effectuons cette tâche ici. Nous récupérons, pour chaque source, l'intégralité du contenu des SKB à aligner que nous enregistrons dans des fichiers temporaires. Une fois les alignements récupérés, nous pouvons supprimer ces fichiers puisque nous continuons ensuite d'interagir avec les SKB par l'intermédiaire de requêtes Sparql. L'implémentation de cette interface dans l'outil Muskca a été réalisée grâce à l'aide d'Elodie Thieblin, stagiaire à l'IRIT lors du développement.

## 3. Solveur Choco

Comme nous l'avons vu dans la section 4.5.4 du chapitre III, nous utilisons un modèle de contraintes à résoudre pour réussir à obtenir une extension à partir du graphe d'incompatibilités. Ce graphe permet la représentation des incompatibilités entre candidats sommets. Le modèle de contraintes à résoudre que nous avons présenté dans cette section est présenté au format GMPL car il permet une représentation lisible du modèle. Afin de trouver une solution à ce modèle de contraintes, nous utilisons un solveur. Un solveur permet de trouver une solution à partir d'un modèle et des données passées en paramètres. Un solveur particulièrement rapide et permettant l'interprétation d'un modèle GMPL est GLPK [97]. Bien que ce solveur propose une librairie pouvant faire l'interface entre une application Java et le solveur, il n'en reste pas moins particulièrement complexe à installer et à utiliser. Pour simplifier l'utilisation de Muskca, nous avons fait le choix d'utiliser un solveur implémenté en Java : Choco [98]. De cette manière, l'utilisation de Muskca ne nécessite aucune installation supplémentaire et nous préservons le caractère multi-plates-formes du projet. De plus, sur trois des quatre catégories de la campagne d'évaluation MiniZinc [99], le solveur Choco est classé troisième sur 12 participants.

Le solveur Choco ne permet pas de traiter directement un modèle exprimé en GMPL. Il propose une syntaxe qui lui est particulière pour implémenter le modèle. Nous avons donc dû réécrire le modèle sous cette forme pour pouvoir exploiter ce solveur [100].

Lors de l'implémentation, nous avons été contraint de modifier les valeurs des scores de confiance des candidats (sommets et arcs). Ces valeurs sont des décimaux compris entre 0 et 1. Si nous essayons de résoudre le modèle en considérant des valeurs décimales, alors nous passons à une catégorie de problèmes non-linéaires. Pour éviter ce problème, nous avons d'abord arrondi ces valeurs à deux décimales après la virgule, puis nous avons multiplié ces valeurs par 100. De cette manière, nous obtenons uniquement des entiers à manipuler. Cela ne change pas le résultat obtenu puisque la fonction objective cherche à maximiser la somme des valeurs des scores de confiance.

97. https://www.gnu.org/software/glpk/
98. http ://www.choco-solver.org/
99. http://www.minizinc.org/challenge2014/results2014.html
100. https://github.com/Murloc6/Muskca/blob/master/src/MultiSources/Solver/ExtensionChocoSolver.java

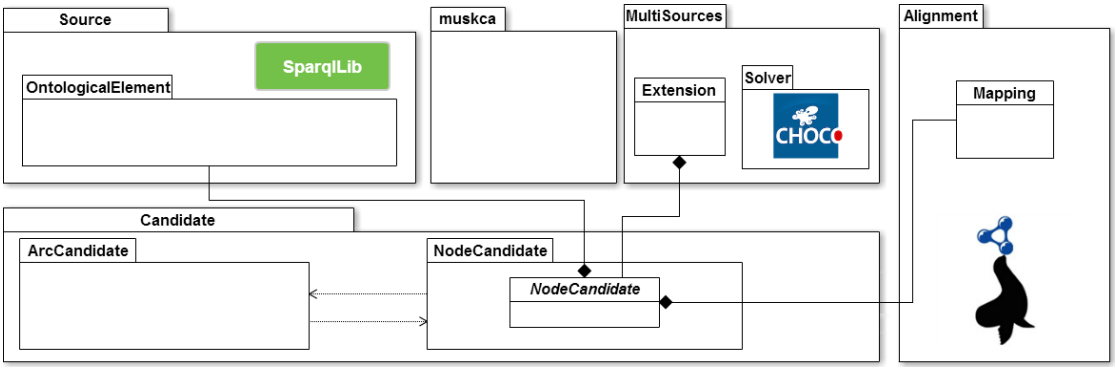


FIGURE 45 – Diagramme des paquetages Muskca

# 4. Modélisation de Muskca

La figure 45 présente le diagramme de paquetages du projet Muskca et les relations principales entre ces paquetages. Ce diagramme comporte cinq paquetages principaux :

**muskca :** contient les classes principales pour lancer l'application,

**Sources :** contient la définition des sources avec les éléments ontologiques qui les composent,

**MultiSources :** contient les éléments qui concernent la fusion multi-sources,

**Aligment :** contient les classes permettant l'alignement entre deux sources,

**Candidate :** contient la définition des candidats nœuds et arcs.

Nous pouvons observer les paquetages dans lesquels nous avons réutilisé les éléments que nous avons présentés précédemment. Nous retrouvons tout d'abord la librairie SparqlLib dans le paquetage *Source*. Dans le paquetage *Alignement* est présent le projet Seals. Enfin, le solveur Choco est utilisé dans le paquetage *Solver*, lui-même contenu dans le paquetage *MultiSources*.

## 4.1. muskca

Le paquetage *muskca* est présenté en détail sur la figure 46. Ce paquetage contient les deux classes nécessaires à l'exécution de l'application. La classe Muskca est la classe principale qui contient la méthode "main" du projet. C'est cette méthode qui est exécutée au lancement de l'application. Cette méthode définit le processus global de Muskca. La classe ParamsReader permet de récupérer les informations contenues dans le fichier de configuration passé en argument. Ce fichier contient tous les détails pour les paramètres pour l'exécution de Muskca.

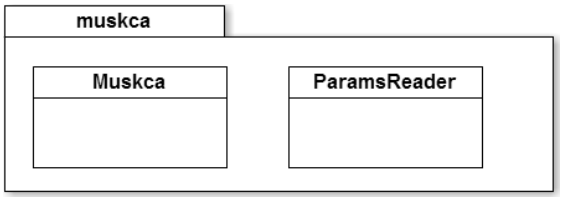


FIGURE 46 – Detail du paquetage *muskca*

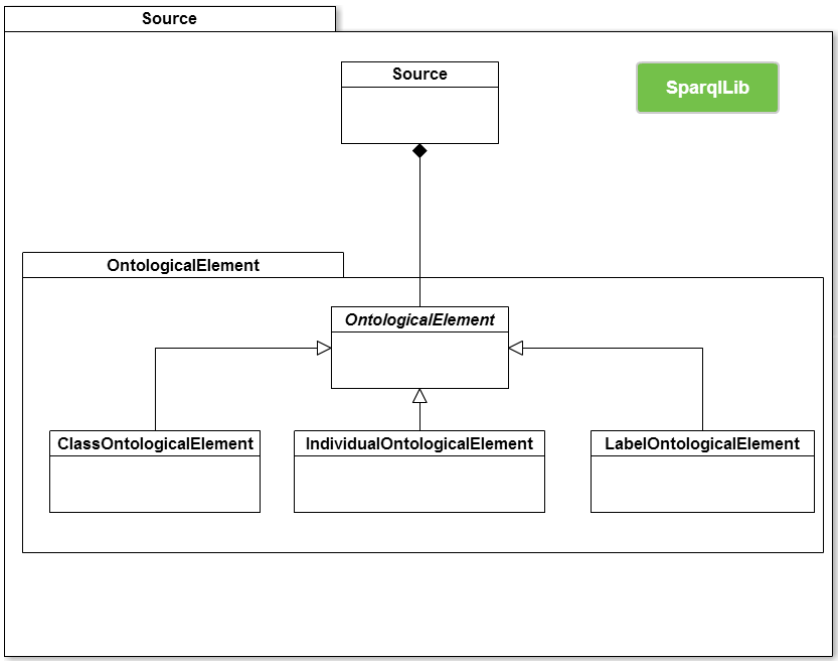


FIGURE 47 – Detail du paquetage *Source*

## 4.2. Source

Le paquetage Source (présenté sur la figure 47) contient la définition des sources et les éléments ontologiques qui les composent. La classe *Source* est la représentation d'une source considérée dans Muskca. Nous utilisons la librairie SparqlLib pour générer un *SparqlProxy* (Cf. section 1.2 du chapitre IV) pour chaque source. Le sous-paquetage *OntologicalElement* contient la définition de ce qu'est un élément ontologique. Au début du processus de Muskca, tous les éléments ontologiques de la source sont récupérés. Un élément ontologique de Muskca peut être de type classe, individu ou label, c'est pourquoi les éléments ontologiques de la source sont stockés sous forme d'instances des classes *ClassOntologicalElement*, *IndividualOntologicalElement* et *LabelOntologicalElement*. Ces classes héritent de la classe abstraite *OntologicalElement*.

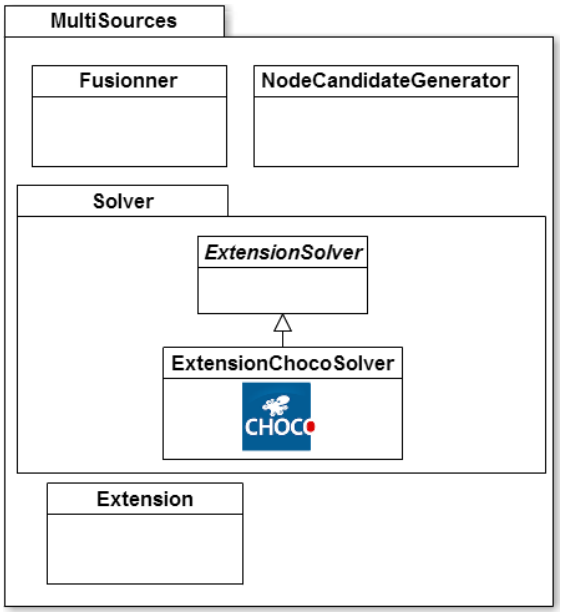


FIGURE 48 – Détail du paquetage *MultiSources*

## 4.3. MultiSources

Le paquetage MultiSources (présenté sur la figure 48) est celui qui fusionne les différentes sources pour générer des candidats. La classe *Fusionner* est la classe permettant de faire le lien entre tous les éléments du projet Muskca. La classe *NodeCandidateGenerator* est celle qui implémente l'algorithme DFS pour générer les candidats nœuds présentés précédemment (Cf. algorithme 2). Le sous-paquetage *Solver* contient les classes nécessaires à l'implémentation du modèle utilisé par le solveur Choco dans le but de générer des extensions. L'utilisation d'une classe abstraite *ExtensionSolver* permet de changer plus facilement de solveur. Cette classe est implémentée par *ExtensionChocoSolver*, qui interface le modèle Choco. La classe *Extension* est la représentation d'une extension, résultat du solveur.

## 4.4. Candidate

Le paquetage *Candidate* (détaillé sur la figure 49) regroupe toutes les classes pour la modélisation de candidats. La classe abstraite *Candidate* regroupe toutes les informations génériques d'un candidat. Elle est implémentée par deux classes abstraites *ArcCandidate* et *NodeCandidate* qui représentent respectivement les candidats arcs et les candidats sommets. Les classes spécialisant *ArcCandidat* permettent la définition des différents types de candidats arcs considérés dans Muskca : *LabelCandidat*, *RelationCandidate* et *TypeCandidate*. Les *TypeCandidate* permettent de représenter les relations rdf :type et rdfs :subClassOf vers un élément du module. La classe *NodeCandidate* est spécialisée par les deux types de candidats sommets considérés : *IndividualCandidate* et *ClassCandidate*.

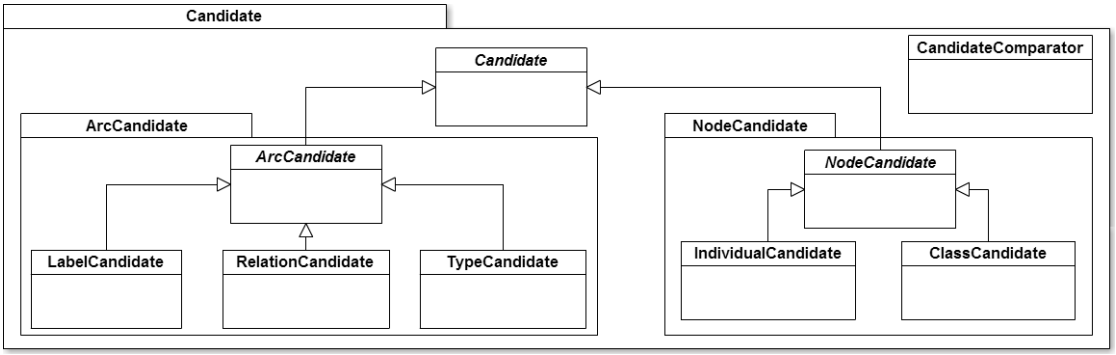


FIGURE 49 – Détail du paquetage *Candidate*

Comme nous l'avons vu dans la section 4.3.2 du chapitre III, un candidat sommet label est directement dépendant du candidat sommet associé. Nous considérons un candidat label comme un candidat arc et non comme un candidat sommet. La classe *CandidateComparator* permet de trier les candidats suivant leur score de confiance en utilisant la méthode générique présente dans Java : Collection.sort [101].

## 4.5. Alignment

Le dernier paquetage, *Alignement*, présenté sur la figure 50, contient les classes permettant de générer et représenter les alignements entre les sources. La classe *Alignment* représente l'alignement entre deux sources. Elle est composée de correspondances, représentées par la classe *Mapping*. La classe *MappingComparator* permet de trier les correspondances en fonction de leurs scores. La classe abstraite *Aligner* est un prototype de système d'alignement. Elle est implémentée par la classe *AlignerSeals*. Cette classe appelle le projet Seals, comme nous l'avons présenté dans la section 2 du chapitre IV. La classe StringDistance permet l'implémentation de la fonction de calcul Jaro-Winkler utilisée pour calculer la distance entre les labels.

101. https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html#sort(java.util.List,java.util.Comparator)

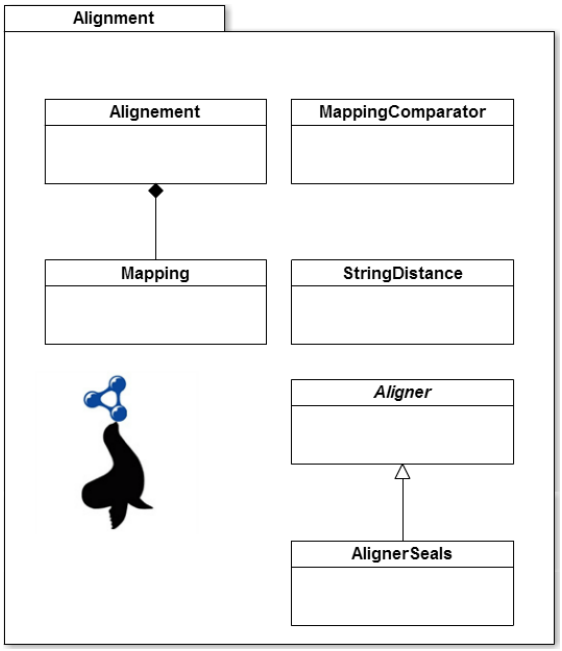


FIGURE 50 – Détail du paquetage *Alignement*

# Cinquième partie . Expérimentations

Nous présentons dans ce chapitre les expérimentations que nous avons menées afin d'évaluer notre approche et notamment son implémentation avec l'outil Muskca.

Nous souhaitons tout d'abord évaluer la capacité de notre approche à générer une base de connaissances d'intérêt à partir de sources non-ontologiques. Pour cela, nous analysons sur un cas réel deux de ses principales caractéristiques.

D'une part, nous cherchons à analyser si l'hypothèse consistant à exploiter les éléments partagés par les différentes sources est pertinente. Pour cela, nous devons demander à des experts d'évaluer les candidats générés et analyser l'impact du score de confiance (reflet du consensus) dans leur classement.

D'autre part, nous voulons évaluer dans quelle mesure ce score de confiance peut être exploité pour assister un expert du domaine dans la sélection des candidats menant à une extension optimale et donc une base de connaissances répondant à son besoin.

Ces deux analyses nous permettront d'évaluer l'intérêt de considérer plusieurs sources non-ontologiques simultanément dans notre processus. Nous présenterons ces analyses dans la section 1.

Nous souhaitons ensuite évaluer la généricité de notre approche. Pour cela, nous devons la tester sur un jeu de données qui ne soit pas issu du domaine de l'agriculture. Nous souhaitons plus particulièrement considérer un nouveau jeu de données de taille importante, qui soit à l'échelle des jeux de données disponibles sur le Web de données liées et qui soit également aux formats préconisés par cette initiative. Nous souhaitons ainsi analyser dans quelle mesure notre approche est aussi adaptée à la génération d'une base de connaissances à partir de plusieurs bases de connaissances déjà constituées dans un domaine. Nous souhaitons ainsi analyser si notre approche est également utile pour fouiller et exploiter les données présentes dans les entrepôts du LOD qui ne cessent d'être enrichis. Nous présenterons cette expérimentation dans la section 2.

## 1. Expérimentation sur un cas réel : la taxonomie des blés

Une composante importante d'une base de connaissances pouvant être utilisée pour annoter les BSV que nous avons présentée dans la section 2.1 du chapitre I est la taxonomie des plantes. Cette taxonomie permet la représentation des connaissances associées aux différentes plantes et ainsi qu'à leur classification. Afin de cibler plus spécifiquement le problème, nous avons restreint le cas réel sur lequel nous évaluons notre approche à la génération d'une base de connaissances portant sur la taxonomie des blés. Cette restriction a été nécessaire pour mettre en place un processus de validation manuel par des experts du domaine. Notre objectif est en effet de leur demander d'évaluer notre approche dans un temps raisonnable.

Sur ce sous-domaine de l'agriculture, nous avons défini un cadre expérimental sur

lequel nous avons appliqué l'approche proposée dans ce mémoire. Différents aspects de cette approche ont été évalués manuellement par 3 experts du domaine des données en agriculture :

— Franck Jabot de l'Irstea Clermont-Ferrand
— Jacques Le Gouis de l' INRA Clermont-Ferrand
— Vincent Soulignac de l'Irstea Clermont-Ferrand

Nous profitons de ce paragraphe pour les remercier pour leur aide précieuse.

Nous allons présenter dans les sections qui suivent le cadre expérimental mise en place ainsi les résultats de l'évaluation. Nous étudierons d'abord les évaluations concernant la qualité des candidats générés avant d'étudier l'intérêt d'utiliser la confiance des candidats dans la génération d'une extension.

## 1.1. Mise en place du cadre expérimental

Mettre en place notre approche sur un cas réel implique de définir le module ontologique (présenté dans la section 1.1.1) et d'identifier les sources et les patrons d'extraction permettant d'exploiter ces sources (présenté dans la section 1.1.2). Nous avons également défini un processus d'interaction avec les experts (décrit dans la section 1.1.3) pour valider l'approche.

### 1.1.1. Module ontologique : AgronomicTaxon

Conformément au processus défini dans notre méthodologie (Cf. section 3.1 du chapitre III), nous avons construit, après avoir consulté les experts, la partie haute du module ontologique. Ce module permet la définition des bornes du domaine que nous souhaitons étudier ici et la modélisation que nous souhaitons avoir dans la base de connaissances finale. Ce module, présenté dans la figure 21, permet la représentation de la taxonomie des plantes.

Ce module ontologique réutilise un certain nombre de patrons de conception provenant d'autres ontologies ou du portail de patrons de conception ontologique ODP [102]. Nous retrouvons par exemple le patron de conception consacré à la représentation de taxonomie : LinnaeanTaxonomy [103].

Afin de garder une trace des méthodes employées pour le développement de ce module ontologique, un site internet a été mis en place pour regrouper toutes les informations relatives au module : https://sites.google.com/site/agriontology/home/irstea/agronomictaxon. Le processus de création de la partie haute d'un module ontologique a été décrit dans la section 3.1 du chapitre III.

Afin d'enrichir ce module, trois sources ont été sélectionnées. Nous allons maintenant présenter les patrons de transformation qui ont été appliqués sur ces sources d'après le module AgronomicTaxon.

102. Ontology Design Patterns - http://ontologydesignpatterns.org/
103. http://ontologydesignpatterns.org/wiki/Submissions:LinnaeanTaxonomy

### 1.1.2. Sources

À partir des besoins représentés dans le module, trois sources ont été sélectionnées par les experts comme étant pertinentes pour l'enrichir : Agrovoc, TaxRef et NCBI. Nous souhaitons représenter la taxonomie des blés. Il est donc important de retrouver des informations concernant cette taxonomie dans la source avec un intérêt particulier pour les labels. En effet, la base de connaissances devant être utilisée pour annoter des textes, il est important de représenter de la façon la plus exhaustive possible les termes du domaine. C'est pour cela que la source Agrovoc a été choisie. Elle contient un grand nombre de labels disponibles en plusieurs langues (Cf. section 2.4.2 du chapitre I). La source NCBI présente elle l'intérêt de contenir un grand nombre de taxons (Cf. section 2.4.1 du chapitre I). La source TaxRef, quant à elle, est incontournable car elle est considérée comme la référence nationale sur le domaine de la taxonomie des plantes (Cf. section 2.4.1 du chapitre I). En choisissant ces trois sources, nous cherchons à tendre vers l'exhaustivité des taxons grâce à NCBI, vers l'exhaustivité des labels grâce à Agrovoc et vers une certaine qualité grâce à TaxRef.

#### Agrovoc

L'analyse de la source Agrovoc a notamment permis de déterminer l'intérêt de cette source pour les nombreux labels utilisés, mais également pour sa réputation et son format d'implémentation. Agrovoc est un thésaurus consacré à l'agriculture, particulièrement utilisé par de nombreux acteurs dans différents pays grâce à ses nombreux labels en plusieurs langues. De plus, son format (SKOS) ouvert et disponible facilite grandement sa réutilisation. La FAO, créateur de la source, la tient à jour et profite de la réputation certaine de cette source pour qu'elle soit enrichie à partir des travaux qui sont faits sur ce thesaurus.

Notre étude porte sur la taxonomie des blés. Nous allons donc limiter notre transformation (définie dans la section 3.2 du chapitre III) à tous les taxons qui sont des sous-rangs du taxon "Triticum" (terme scientifique pour "blé").

Nous définissons un patron de transformation permettant de récupérer tous les taxons qui sont, d'après la hiérarchie SKOS (skos :broader), sous ce taxon. Nous considérons ce taxon comme un taxon "limitant", puisque c'est toute la branche située sous ce taxon dans la hiérarchie que nous souhaitons récupérer. Tous les taxons présents dans Agrovoc qui ne sont pas dans cette branche (donc directement ou indirectement liés à "Triticum" par la relation "skos :broader") ne seront pas extraits car considérés comme hors domaine. Pour chaque taxon extrait, nous considérons la relation "hasTaxonomicLevel" comme représentant la relation entre un taxon et son type. Afin de suivre la modélisation souhaitée dans le module AgronomicTaxon, nous transformons cette relation "hasTaxonomicLevel" en une relation "instanceOf" dans la base de connaissances source. Pour déterminer la classe correspondant au type du taxon, nous observons d'abord s'il existe une correspondance vers le module. Dans ce cas, nous typons le taxon directement avec la classe du module, sinon nous créons une nouvelle classe correspondante qui sera une sous-classe de "Taxon". Entre tous les taxons extraits, nous récupérons les relations

"skos :broader", qui représentent ici la relation hiérarchique de la taxonomie. Dans notre module, cette relation est équivalente à "hasHigherRank". Nous transformons donc toutes les relations "skos :broader" entre deux taxons extraits par la relation "hasHigherRank" du module. Grâce à son implémentation sous le format SKOS, nous exploitons cette source directement par des requêtes Sparql.

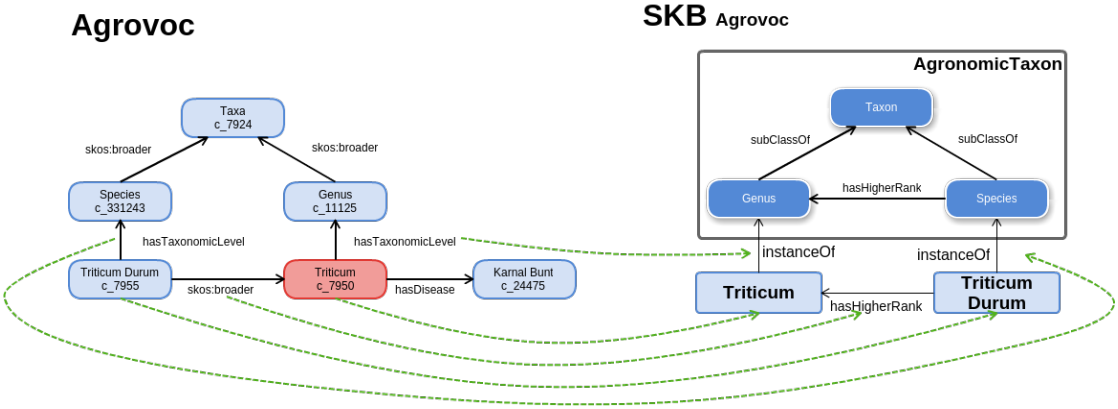


FIGURE 51 – Exemple de transformation de la source Agrovoc d'après le module AgronomicTaxon

La figure 51 présente un exemple de l'utilisation de ce patron de transformation sur une sous-partie de la source Agrovoc. Le taxon "Triticum" dans la source est représenté en rouge puisqu'il est le taxon limitant. Nous pouvons remarquer la transformation des différents taxons en individus dans la base de connaissances source (SKB). Le type des taxons est défini à partir de la relation "hasTaxonomicLevel" de la source. Nous n'avons pas représenté les correspondances entre la source et le module qui ont été définies par soucis de clarté de la figure, mais ces correspondances sont visibles sur la figure 23. La relation "skos :broader" entre les deux taxons est transformée dans la base de connaissances source en "hasHigherRank". La transformation des labels n'est pas représentée dans cette figure pour éviter de surcharger le schéma.

Le détail et l'implémentation de cette transformation sont disponibles sous la forme d'un projet Java accessible à cette adresse : `https://github.com/Murloc6/T2RKB`.

### TaxRef

La source TaxRef, présentée dans la section 2.4.1 du chapitre I est la taxonomie de référence en France pour les organismes vivants. Ce statut de référentiel permet d'avoir une assurance de qualité des éléments provenant de cette source. C'est pour cela que nous avons choisi d'intégrer cette source au processus. La mise à jour de cette source se faisant manuellement pour garantir une qualité optimale, la fraîcheur et l'exhaustivité de celle-ci en sont affectées.

L'implémentation de la source TaxRef est faite sous forme de fichiers TSV [104]. Ces fichiers sont liés les uns aux autres en utilisant un système de clef primaire/clef étrangère, comme dans une base de données. Nous retrouvons l'organisation générale de la source présentée sur la figure 8.

La table qui nous intéresse particulièrement est la table "TAXREF" qui contient l'ensemble des taxons. Toutes les informations dont nous avons besoin sont dans ce fichier. L'implémentation sous un format TSV ne permet pas une représentation simplifiée des taxons. Pour cela, chaque ligne représente un terme d'un taxon. Si le taxon contient plusieurs termes (synonymes), alors il sera représenté sur plusieurs lignes. Afin de déterminer si des termes appartiennent au même taxon, il faut observer l'attribut "CD_REF" qui est l'identifiant du terme de référence pour ce taxon. L'identifiant d'un terme est stocké dans l'attribut "CD_NOM". L'attribut "CD_SUP" représente l'identifiant du terme de référence directement supérieur dans la hiérarchie taxonomique représentée. Les attributs "FR" et "EN" représentent les termes en français et en anglais. D'après cette description, nous avons défini l'algorithme d'extraction en partant du terme référent du taxon "Triticum" qui correspond aux blé. Nous avons ensuite cherché tous les termes référents qui ont pour "CD_SUP" l'identifiant du terme de référence "Triticum". Nous regardons pour chacun de ces termes le contenu de l'attribut "RANG". Nous regardons si pour ce rang donné, une correspondance existe avec le module sinon, si la classe n'est pas déjà créée, nous créons une nouvelle classe pour ce nouveau rang. Cette nouvelle classe sera une spécialisation de "Taxon" dans notre module. Nous créons ensuite un individu avec comme type soit la classe du module s'il y a une correspondance ou la nouvelle classe créée sinon. Nous définissons la relation "hasHigherRank" vers le taxon supérieur (ici "Triticum") et récupérons tous les termes (scientifiques et vernaculaires). Nous cherchons ensuite tous les termes associés pour enrichir les labels du taxon nouvellement créé. Nous cherchons alorq tous les taxons qui ont pour "CD_SUP" ce taxon nouvellement créé et nous appliquons ce même processus. Nous arrêtons le processus lorsqu'il n'y a plus de taxon ayant l'attribut "CD_SUP" vers un taxon extrait.

La figure 52 présente un exemple de la transformation d'une sous-partie de TaxRef en utilisant la transformation que nous venons de présenter. Nous pouvons observer à gauche une sous-partie du fichier de TaxRef qui nous a permis de générer la base de connaissances source associée. Nous remarquons que dans ce fichier apparaissent trois taxons. Le premier est Triticum : c'est le taxon limitant dans notre algorithme. Ce taxon a pour rang "GN" que nous avons identifié (manuellement) comme étant la classe "Genus" de notre module. Nous définissons alors le taxon "Triticum" de type "Genus". Nous pouvons observer ensuite le taxon qui a pour "CD_TAXSUP" (taxon supérieur) l'identifiant du taxon "Triticum" : "Triticum Turgidum". Ce taxon est du rang "ES" qui a été identifié comme étant "Species" dans notre module. L'apparition de l'identifiant de "Triticum" (198676) dans l'attribut "CD_TAXSUP" de "Triticum Turgidum" nous permet de poser la relation "hasHigherRank" entre ces deux taxons dans la base de connaissances source générée. Le taxon "Triticum turgidum subsp. durum" qui est immédiatement en-dessous dans la hiérarchie de "Triticum Turgidum" et séparé en trois lignes. Pour

104. Tab Separated Value - https://fr.wikipedia.org/wiki/Tabulation-separated_values

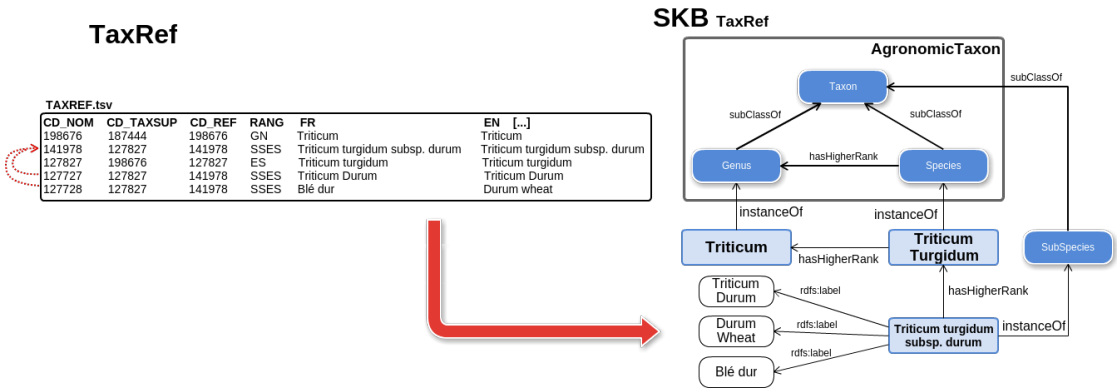


FIGURE 52 – Exemple de transformation de la source TaxRef d'après le module AgronomicTaxon

représenter les différents labels, TaxRef sépare la description du taxon en plusieurs lignes comme présenté dans l'exemple. Le terme de référence est donc "Triticum turgidum subsp. durum" et les deux autres lignes sont des labels supplémentaires : "Triticum Durum", "Durum wheat" et "Blé dur". Nous avons représenté par des flèches pointillées rouges les dépendances de ces lignes avec le terme de référence. Le taxon supérieur au taxon "Triticum turgidum subsp. durum" est "Triticum Turgidum", c'est pour cela que nous retrouvons la relation "hashHigherRank" dans la base de connaissances source. Le rang de ce taxon est "SSES". Ce rang représente une sous-espèce, ou une "sub-species" en anglais. Cette classe n'existe pas dans le module AgronomicTaxon. Nous définissons donc une nouvelle classe pour représenter ce rang. Nous attribuons le type du taxon "Triticum turgidum subsp. durum" à cette nouvelle classe "SubSpecies". Nous avons représenté uniquement les labels associés au taxon "Triticum turgidum subsp. durum" pour éviter de surcharger le schéma.

Le détail et l'implémentation de cette transformation est disponible sous la forme d'un projet Java disponible à cette adresse : https://github.com/Murloc6/TaxRef2RKB/.

### NCBI

La source NCBI se différencie particulièrement des deux autres sources par sa vocation à être exhaustive. Son processus de validation de nouveaux taxons étant particulièrement rapide, elle contient un grand nombre de taxons. Cette rapidité de validation entraîne néanmoins certaines incohérences et erreurs. NCBI est particulièrement fournie en labels bien que la langue utilisée soit toujours l'anglais, mais elle référence des labels plus anciens qui ont été remplacés depuis. Ces anciens labels permettent de faire le lien avec de vieux documents utilisant d'anciens labels, par exemple. Son exhaustivité fait d'elle une source incontournable lorsque l'on traite de la taxonomie des êtres vivants.

La transformation de cette source se présente sous la forme de plusieurs fichiers. Nous

en avons utilisé deux :

**nodes.dmp :** Ce fichier référence tous les taxons présents dans la source NCBI

**names.dmp :** Ce fichier regroupe les différents termes utilisés pour désigner un taxon

Ces deux fichiers sont sous un format spécifique, qui ressemble au TSV, si ce n'est que le caractère séparant les différentes valeurs n'est pas la tabulation mais la barre verticale "|". La signification des différents attributs présents dans ces fichiers est documentée dans le document accessible à l'adresse ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump_readme.txt.

Comme pour les deux autres sources, nous cherchons à extraire tous les taxons inférieurs dans la hiérarchie taxonomique au taxon "Triticum". À partir du taxon "Triticum", nous récupérons tous les taxons du rang directement inférieur en observant tous les taxons du fichier "Nodes" qui contiennent l'identifiant de Triticum dans le deuxième attribut ("parent tax_id"). Pour chacun de ces taxons, nous analysons le troisième attribut ("rank") qui permet de déterminer le rang taxonomique du taxon. Si ce rang apparaît dans une correspondance avec un rang du module ontologique, alors nous créons un nouvel individu du type correspondant dans le module. Sinon, si la classe associée n'est pas déjà créée, nous créons une nouvelle classe sous-classe de "Taxon" dans le module. Pour chacun des taxons, nous récupérons l'ensemble des labels présents dans le fichier "Names". Pour chacun de ces taxons, nous cherchons ensuite tous les taxons qui lui qont directement inférieurs (qui ont l'attribut "parent tax_id" correspondant à l'id du taxon courant) et nous appliquons le même processus. Nous arrêtons ce processus lorsqu'il n'existe plus de taxon ayant pour attribut "parent tax_id" un taxon déjà transformé.

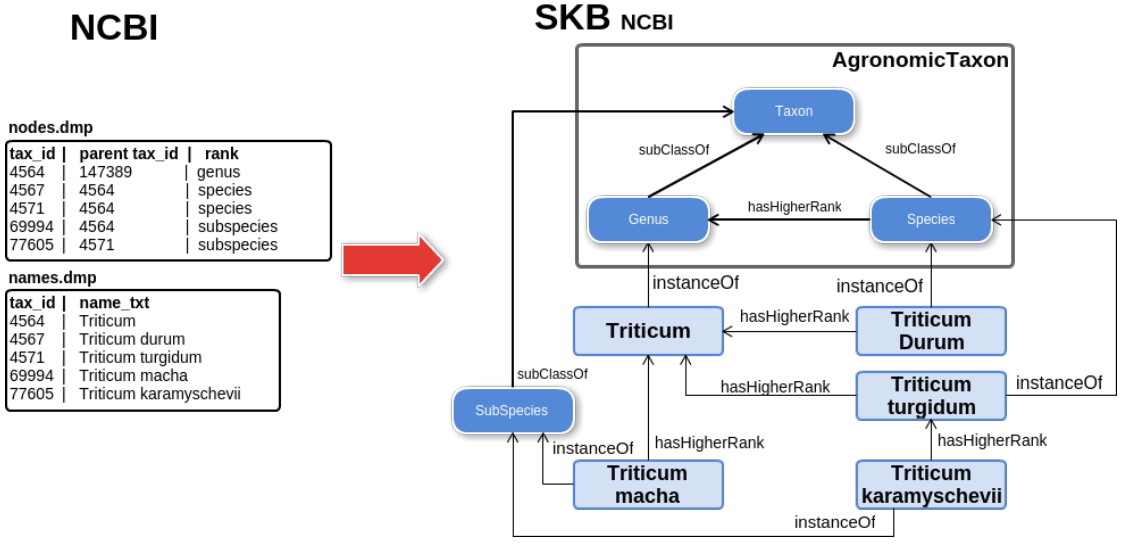


FIGURE 53 – Exemple de transformation de la source NCBI d'après le module AgronomicTaxon

Un exemple est présenté sur la figure 53. À gauche de cette figure sont présents les deux fichiers que nous avons utilisés sous leur format TSV. Dans le fichier nodes.dmp sont présents les taxons à représenter dans la base de connaissances source. Le premier taxon (tax_id 4564) est le taxon représentant "Triticum" qui est le taxon limitant. Nous récupérons ensuite tous les taxons inférieurs à ce taxon limitant. Ici, nous avons "Triticum Durum", "Triticum turgidum" et "Triticum macha". Les deux premiers sont identifiés comme du rang "species" d'après le fichier nodes.dmp. Ce rang est représenté dans le module donc, en utilisant les correspondances posées manuellement, nous obtenons des taxons de type "Species" du module. Le dernier, "Triticum macha", est lui identifié comme un "subspecies". Cette classe n'est pas définie dans le module. Nous en créons donc une nouvelle. Ces trois taxons sont les sujets de trois relations "hasHigherRank" vers "Triticum". Un dernier taxon est extrait de cet exemple : "Triticum karamyschevii". Ce taxon est du type "SubSpecies", classe nouvellement créée et a une relation "hasHigherRank" vers "Triticum turgidum". Nous n'avons pas représenté les labels pour simplifier le schéma.

Le détail et l'implémentation de cette transformation sont disponibles sous la forme d'un projet Java accessible à cette adresse : https://github.com/Murloc6/NCBI2RKB/.

En appliquant ces patrons de transformation nous obtenons les trois bases de connaissances sources issues de chaque source. Nous pouvons résumer les éléments extraits de chaque source avec le tableau 13.

| Source | Classes | Taxons | Labels |
|---|---|---|---|
| Agrovoc | 5 | 21 | 108 |
| TaxRef | 6 | 17 | 89 |
| NCBI | 15 | 99 | 157 |

TABLE 13 – Quantités d'éléments extraits des sources pour la taxonomie des blés

Nous observons dans ce tableau que la source NCBI est bien plus fournie que les deux autres sources avec 15 classes extraites et 99 taxons. Nous avons extrait 108 labels pour 21 taxons et 5 classes provenant d'Agrovoc. Nous avons restreint cette extraction aux labels français et anglais, puisque ce sont les seules langues disponibles dans les autres sources. Il n'était pas nécessaire d'extraire tous les labels des autres langues provenant d'Agrovoc pour notre expérimentation puisqu'ils n'auraient pas amélioré l'alignement des sources, ni amené plus de candidats labels impliquant plusieurs sources. Dans un cas réel, nous pourrions lever cette limitation pour obtenir un maximum de candidat.

### 1.1.3. Données de référence

Une fois les bases de connaissances sources générées, nous avons demandé aux experts d'analyser les éléments de chacune d'entre-elles, afin de déterminer s'ils devaient être représentés dans la base de connaissance finale. Nous avons choisi de suivre cette démarche de façon à avoir des données de référence pouvant être comparées à différentes versions de la base de connaissances générée par notre approche. Une alternative aurait été de demander aux experts d'évaluer directement la base de connaissances générée par notre

approche. Nous avons préféré mettre en place ce processus d'évaluation plus générique pour tester plusieurs configurations de notre approche, sans solliciter à chaque fois les experts. De cette manière, nous évaluerons si les éléments validés par les experts apparaissent bien dans au moins un candidat de la base de connaissances finale. Un autre intérêt de cette approche est de pouvoir déterminer s'il existe un consensus au sein des sources et si les candidats que nous générons s'appuient effectivement sur les éléments intéressants des sources et donc s'ils ont été validés par les experts.

Nous avons implémenté une interface Web permettant de demander aux experts l'intérêt de considérer les éléments de chaque source dans la base de connaissances finale. Pour simplifier ce travail, nous avons ciblé cette validation sur les taxons. Les experts n'ont donc pas validé les classes extraites. En d'autres termes, nous avons extraits les individus de type "Taxon" (ou d'un sous-type de "Taxon). Pour chacun des taxons nous leur avons posé quatre questions :

1. **Est-ce que le taxon appartient au domaine ?** Nous demandons dans un premier temps si l'élément présenté est vraiment un taxon (un élément de la taxonomie). Nous voulons également savoir s'il est dans le périmètre de la base de connaissances, c'est à dire Triticum. Cette question nous permettra de savoir si cet élément doit être retrouvé dans la base de connaissances finale.
2. **Est-ce que les labels désignent la même entité ?**

   Il y a plusieurs labels disponibles dans la base de connaissances (particulièrement dans Agrovoc) mais parfois, certains labels sont inexacts, parce qu'ils ne sont pas synonymes ou parce qu'ils ne sont pas des traductions exactes (si la source contient des labels en plusieurs langues). Nous considérerons la réponse à cette question comme une validation de labels associés à un taxon.
3. **Est-ce que le taxon est plus spécifique qu'un autre taxon ?**

   Lorsque l'on peut extraire une relation "hasHigherRank" entre deux taxons de la source, nous voulons valider ce lien. Nous voulons donc savoir si le premier taxon est un sous-taxon du second en considérant une hiérarchie taxonomique.
4. **Est-ce que ce taxon appartient au rang donné ?** Il y a parfois des informations concernant le rang du taxon dans la source. Nous souhaitons donc valider cette extraction et définir le type du taxon : est-ce une espèce, un genre, etc. Cette question permet de valider la relation "rdf :type" entre le taxon et une classe du module (puisque nous n'ajoutons pas de nouvelles classes ici).

L'interface de validation est présentée en figure 54. Cette figure montre que pour chaque question, il y a trois possibilités de réponse : Valide, Non valide, Ne sait pas. À la fin de la validation, nous avons une liste d'éléments ontologiques validés par les experts. Nous pouvons les comparer aux candidats générés par Muskca.

Pour exploiter les réponses données par les experts et analyser les candidats générés par notre approche, nous souhaitons analyser dans quelle mesure nos candidats s'appuient sur des éléments validés par les experts dans les sources. Pour utiliser les résultats obtenus comme référence de comparaison, nous avons vérifié que les experts étaient d'accord sur les éléments validés au sein des sources. Nous considérons qu'il existe un accord entre les

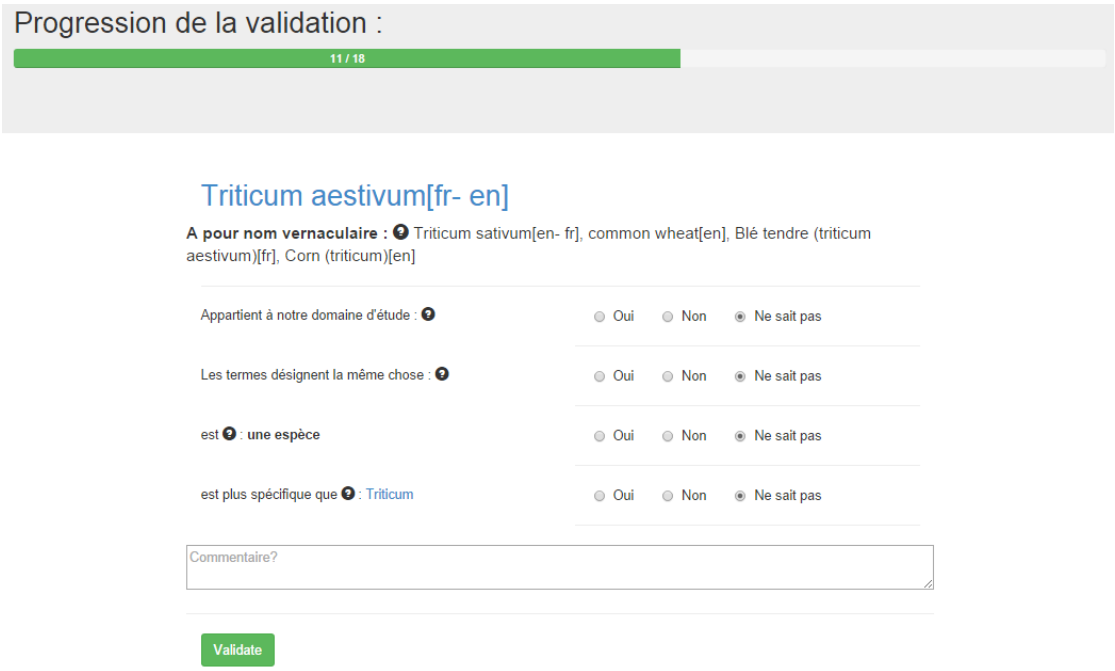


FIGURE 54 – Interface de validation

experts quand au moins deux experts ont validé le même élément et que le troisième a sélectionné l'option "Ne sait pas". Nous avons tout d'abord calculé le rapport entre le nombre d'experts et leur nombre de validations et nous obtenons une valeur de **0,82**. Nous avons également calculé le score de Fleiss Kappa [Fleiss and Cohen, 1973] sur les validations des experts. Nous obtenons alors un score de Fleiss Kappa de **0,69**. Ces deux valeurs montrent que les experts s'accordent la plupart du temps sur la validation des éléments. Nous pouvons donc utiliser les retours des experts comme des données de référence permettant d'évaluer les candidats.

## 1.2. Évaluation de la qualité des candidats générés

Cette section vise à décrire comment nous avons évaluer la qualité de la base de connaissances générée par notre approche. Pour cela nous allons utiliser le jeu de données de référence généré avec l'aide de trois experts, présenté dans le paragraphe précédent. Nous comparerons les éléments ontologiques impliqués dans les candidats et les éléments validés par les experts. Cette comparaison repose sur la calcul de score de de précision, rappel et F-mesure

Dans cette section nous présenterons dans un premier temps notre stratégie d'évaluation avant de présenter les résultats que nous analyserons ensuite. Nous avons comparé les résultats en utilisant les trois fonctions de calculs définis dans la section 4.4 du chapitre III.

### 1.2.1. Stratégie de validation

Notre objectif ici est de vérifier la qualité des candidats générés et l'impact de la fonction de calcul de confiance utilisée pour les ordonner.

En appliquant notre approche sur les bases de connaissances sources décrites dans la section 1.1.2 de ce chapitre, nous obtenons un ensemble de candidats sommets et de candidats arcs.

Nous avons mené 3 expérimentations afin d'évaluer les candidats que nous générons. Chacune de ces expérimentations porte sur l'utilisation d'une des fonctions de calcul de confiance présentées dans la section 4.4 du chapitre III. Pour chacune d'entre-elles, nous comparons les candidats obtenus avec les éléments validés par les experts. Rappelons qu'un candidat sommetest défini tel que
$Cand = (V_{Cand}, E_{Cand}, \Sigma V_{Cand}, etiqV_{Cand}, valueE)$
et un candidat arc tel que
$Cand = (V_{Cand}, E_{Cand}, \Sigma V_{Cand}, \Sigma E_{Cand}, etiqV_{Cand}, sourceE_{Cand}, cibleE_{Cand})$.
Nous cherchons à déterminer quels sont les éléments validés par les experts apparaissant dans les candidats. Pour cela nous travaillerons alors uniquement sur les $V_{cand}$ des candidats quelque soit leur type.

Nous ne générons pas d'extension ici car nous souhaitons nous concentrer sur l'évaluation des candidats. Comme nous l'avons vu, les experts ont validé les éléments des bases de connaissances sources qu'ils souhaitent retrouver dans la base de connaissance finale. Nous considérons un candidat comme valide si tous les éléments qui le composent ($\forall v \in V_{Cand}$) ont été validés par les experts. De plus, un élément est considéré comme validé par les experts si au moins 2 des 3 experts ont indiqué que l'élément est juste et qu'aucun n'a annoncé qu'il est faux. En d'autres termes, un candidat est validé si tous ses éléments ont été annoncé par au moins 2 des 3 experts comme valides. Cette validation a été effectuée grâce à l'interface proposé aux experts. Les individus ont été validés par la première question, les arcs "rdfs :label" par la deuxième, les arcs "hasHigherRank" par la troisième et les arcs "rdf :type" par la dernière.

En considérant cette notion de candidat validé par les experts nous pouvons calculer les valeurs de précisions, rappel et f-mesure pour évaluer la pertinence des candidats générés. La précision nous permettra de déterminer combien de candidats générés sont considérés comme valides par rapport au nombre de candidats générés. Cette précision représente directement la qualité des candidats générés. Nous définissons la fonction de calcul de la précision de la manière suivante :

$$Precision(KB_{finale}) = \frac{\sum_{i=0}^{|KB_{finale}|} |Cand_i, V_{Cand_i} \subseteq Gold_{standard}|}{|KB_{finale}|} \quad (21)$$

Nous considérons dans cette formule que $KB_{finale}$ est la base de connaissances finale contenant tous les candidats générés. $Gold_{standard}$ est l'ensemble des éléments validés par les experts dans le processus de création du jeu de données de référence. Le numérateur de cette précision est le nombre de candidats $Cand_i$ dont tous les éléments le composant $V_{Cand_i}$ apparaissent dans le jeu de données de référence $Gold_{standard}$ (donc validés par

| | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **Individus** | 0,87 | 1 | 0,93 |
| **Relations** | 0,74 | 0,99 | 0,85 |
| **Types** | 0,45 | 1 | 0,62 |
| **Labels** | 0,74 | 0,79 | 0,77 |

TABLE 14 – Résultats sur la taxonomie des blés sans filtrage

les experts). Le dénominateur est le nombre de candidats générés. Plus la précision est élevée et plus les candidats générés sont considérés comme validés par les experts.

Nous calculons aussi le rappel qui est la valeur représentant l'exhaustivité du résultat obtenu. Ce rappel permet d'évaluer la proportion d'éléments validés par les experts apparaissant dans les candidats générés. Nous calculons le rappel de la manière suivante :

$$Rappel(KB_{finale}) = \frac{\sum_{i=0}^{|Gold_{standard}|} |elem \in Gold_{standard}, elem \in V_{Cand_j}|}{|Gold_{standard}|} \quad (22)$$
$$Cand_j \in KB_{finale}$$

Le numérateur compte le nombre d'éléments *elem* présents dans le jeu de données de référence $Gold_{standard}$ et qui apparaissent aussi dans un candidat $Cand_j$ de la base de connaissances finale $KB_{finale}$. Le dénominateur est le nombre d'éléments dans le jeu de données de référence. Plus ce rappel sera élevé et plus les éléments validés par les experts seront présents dans des candidats et donc plus le résultat s'approchera de l'exhaustivité.

Pour permettre de calculer une valeur unifié et représenter la pertinence des résultats prenant en compte la précision et le rappel nous calculons le F-Mesure. Cette fonction permet d'agréger les deux valeurs de précision et de rappel.

$$F - Mesure(KB_{finale}) = \frac{2 * Precision(KB_{finale}) * Rappel(KB_{finale})}{Precision(KB_{finale}) + Rappel(KB_{finale})} \quad (23)$$

Nous allons calculer ces scores de précision, rappel et F-mesure dans le cadre de différentes expérimentations.

### 1.2.2. Résultats

Nous commencerons par calculer la précision, le rappel et la f-mesure pour tous les candidats générés.

Nous menons ensuite une expérimentation pour évaluer l'impact de chacune des fonctions de calcul de confiance sur le classement des candidats. Nous considérons d'abord $trust_{likelihood}$ puis $trust_{degree}$ et enfin $trust_{choquet}$. Pour chacune de ces expérimentations nous appliquerons plusieurs filtres sur le score de confiances. Nous appliquerons un filtre sur les candidats par pas de 0.1 jusqu'à la valeur 0,9. Nous calculerons les 3 mesures pour chacun des pas. De cette manière il sera possible de comparer l'intérêt des fonctions de

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,87 | 1 | 0,93 |
| **0,2** | 0,87 | 1 | 0,93 |
| **0,3** | 0,87 | 1 | 0,93 |
| **0,4** | **0,99** | 0,63 | 0,77 |
| **0,5** | 0,99 | 0,63 | 0,77 |
| **0,6** | 0,99 | 0,63 | 0,77 |
| **0,7** | 1 | 0,60 | 0,75 |
| **0,8** | 1 | 0,60 | 0,75 |
| **0,9** | 1 | 0,60 | 0,75 |

TABLE 15 – Expérimentations $trust_{likelihood}$ : individus

calcul. La fonction de calcul qui permettra une meilleure représentation du consensus verra son score de précision être d'autant plus élevé que le filtre sera élevé lui aussi.

Dans le cadre de chacune des expérimentations, nous avons également évalué notre approche sur les différents types de candidats qu'elle permet de générer :

— les "candidats individus" qui représentent les taxons extraits des différentes sources.
— les "candidats relations" qui sont les relations "hasHigherRank" entre deux taxons.
— les "candidats type" qui sont les relations instanceOf pour chaque taxon, représentant l'attribution à un rang taxonomique.
— les "candidats label" qui sont les labels associés aux taxons.

Nous ne considérons pas ici les "candidats classe" car ils n'ont pas été validés par les experts.

**Analyse de l'ensemble des candidats générés** Le premier fait notable sur ces résultats sont les scores encourageants sur le tableau 14. Par exemple, la f-mesure des candidats individus est à 0,93. Ce score signifie que notre proposition permet de générer des candidats de bonnes qualités même sans filtrage. Le score le moins élevé ici concerne les candidats Types avec une précision de 0,45. Ceci s'explique par de nombreuses erreurs dans les sources concernant la catégorisation des taxons. Les taxons, les labels et les relations entre taxons sont corrects et consensuels mais ils sont mals placés dans la classification taxonomique.

**Fonction de confiance : $trust_{likelihood}$**

Nous présentons dans cette section les résultats obtenus en utilisant la fonction de calcul $trust_{likelihood}$. Les tableaux 15, 16, 17, 18 présentent ces résultat en fonction du seuil utilisé pour le filtrage. Ce seuil permet de considérer uniquement les candidats ayant un score supérieur ou égale à ce seuil.

Les résultats obtenus grâce à la fonction $trust_{likelihood}$ sont intéressants, notamment ceux présentés dans le tableau concernant les individus (tableau 15). Nous voyons qu'ici

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,74 | 0,99 | 0,84 |
| **0,2** | 0,74 | 0,99 | 0,84 |
| **0,3** | **0,74** | 0,99 | 0,84 |
| **0,4** | 0,58 | 0,43 | 0,50 |
| **0,5** | 0,58 | 0,43 | 0,50 |
| **0,6** | **0,58** | 0,43 | 0,50 |
| **0,7** | **1** | 0,21 | 0,35 |
| **0,8** | 1 | 0,21 | 0,35 |
| **0,9** | 1 | 0,21 | 0,35 |

TABLE 16 – Expérimentations $trust_{likelihood}$ : relations

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,45 | 1 | 0,62 |
| **0,2** | 0,45 | 1 | 0,62 |
| **0,3** | **0,45** | 1 | 0,62 |
| **0,4** | 0,48 | 0,48 | 0,46 |
| **0,5** | 0,48 | 0,45 | 0,46 |
| **0,6** | 0,48 | 0,45 | 0,46 |
| **0,7** | 0,55 | 0,37 | 0,44 |
| **0,8** | 0,55 | 0,37 | 0,44 |
| **0,9** | 0,55 | 0,37 | 0,44 |

TABLE 17 – Expérimentations $trust_{likelihood}$ : types

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,74 | 0,79 | 0,77 |
| **0,2** | 0,74 | 0,79 | 0,77 |
| **0,3** | 0,74 | 0,79 | 0,77 |
| **0,4** | 0,49 | 0,60 | 0,54 |
| **0,5** | 0,49 | 0,60 | 0,54 |
| **0,6** | 0,49 | 0,60 | 0,54 |
| **0,7** | 0,45 | 0,18 | 0,26 |
| **0,8** | 0,45 | 0,18 | 0,26 |
| **0,9** | 0,45 | 0,18 | 0,26 |

TABLE 18 – Expérimentations $trust_{likelihood}$ : labels

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,99 | 0,6 | 0,77 |
| **0,2** | 0,99 | 0,63 | 0,77 |
| **0,3** | **1** | 0,62 | 0,77 |
| **0,4** | 1 | 0,60 | 0,75 |
| **0,5** | 1 | 0,60 | 0,75 |
| **0,6** | 1 | 0,60 | 0,75 |
| **0,7** | 1 | 0,26 | 0,41 |
| **0,8** | 1 | 0,26 | 0,41 |
| **0,9** | 1 | 0,26 | 0,41 |

TABLE 19 – Expérimentations $trust_{degree}$ : individus

la précision est à 0,99 à partir du seuil 0,4. Ceci implique que si un candidat individu est composé d'éléments provenant d'au moins deux sources il est très probablement validé par les experts. Ce score est plus mitigé pour les candidats Types (tableau 17) puisque la précision est à 0.45. Ceci s'explique par le fait que les relations de types ("instanceOf") sont souvent spécifiques à une source et n'apparaissent que rarement dans une autre. Ceci s'observe d'autant plus que la précision n'augmente que très peu lorsque le seuil augmente. Un point important dans ces résultats apparaît dans le tableau des candidats relations (tableau 16) représentant la relation "hasHigherRank". La précision est de 0,74 pour les seuils de 0,1 à 0,3, puis de 0.58 jusqu'au seuil 0,6 et passe directement à 1 ensuite. Cette chute avant d'augmenter à nouveau s'explique par le fait que beaucoup de candidats relations considérés comme valide n'apparaissent que dans une seule source. Ils sont donc rejetés à un niveau de seuil impliquant qu'il y ait au moins 2 sources pour considérer le candidat, ce qui explique la chute. Néanmoins la remontée à 1 à la fin signifie que tous les candidats relations qui apparaissent dans 3 sources sont validés par les experts. Ce phénomène apparaît aussi sur les candidats labels (tableau 18) dont la précision diminue avec l'augmentation du seuil. Ceci peut s'expliquer aussi par le fait que les labels n'apparaissent majoritairement que dans une source.

Globalement nous pouvons observer une augmentation de la précision avec l'augmentation du seuil dans les différents tableaux. Ceci permet de vérifier notre hypothèse de départ. Si un élément est présent dans plusieurs sources alors il est de meilleur de qualité. Les candidats labels font exceptions puisque la précision diminue.

#### Fonction de confiance : $trust_{degree}$

La fonction de calcul de confiance $trust_{degree}$ pour les candidats arcs étant la même que $trust_{likelihood}$ nous ne présentons ici que les résultats concernant les candidats individus.

En observant les résultats de cette fonction $trust_{degree}$ présentés dans le tableau 19 et en les comparant avec ceux de la fonction $trust_{likelihood}$ (tableau 15) nous pouvons observer plusieurs phénomènes. Tout d'abord nous pouvons observer une augmentation plus significative de la précision dés le seuil à 0,1. Néanmoins le rappel est lui fortement

| | Seuils | Precision | Rappel | F-Measure |
|---|---|---|---|---|
| **Relations** | 0,6 | 0,68 | 0,51 | 0,58 |
| | 0,9 | 0,93 | 0,32 | 0,47 |
| **Types** | 0,6 | 0,77 | 0,39 | 0,52 |
| | 0,9 | 0,81 | 0,39 | 0,53 |
| **Labels** | 0,6 | 0,39 | 0,24 | 0,30 |
| | 0,9 | 0 | 0 | 0 |

TABLE 20 – Expérimentations $trust_{degree_p}$ : sur les candidats arcs

impacté par ce seuil. Ceci s'explique par le fait que l'utilisation des correspondances dans le calcul de la confiance consensuelle est beaucoup plus discriminante. Les candidats ayant un haut score de confiance sont ceux qui impliquent beaucoup de correspondances. Les candidats impliquant moins de 3 sources ici sont rejetés dès les seuils assez bas, bien que certains soient valides. Ceci explique la diminution rapide du rappel. Les candidats impliquant trois sources n'utilisent pas forcément beaucoup de correspondances. Les éléments ontologiques ne sont alors pas fortement connectés entre eux par des correspondances. Ce qui explique la diminution significative du rappel pour des seuils hauts.

Nous pouvons en déduire ici que l'utilisation des correspondances dans le calcul de la confiance permet une représentation plus discriminante de la confiance. Les candidats n'impliquant pas beaucoup de sources auront un score de confiance assez bas. Néanmoins nous maintenons la vérification de notre hypothèse ici. Plus un candidat utilise de correspondances (et donc plus son score $trust_{degree}$ est élevé) plus sa qualité est assurée. Nous l'observons avec la précision à 1 dés le seuil 0,3.

#### Fonction de confiance : $trust_{degree_p}$ avec propagation

Le tableau 20 présente les résultats obtenus en utilisant la fonction de calcul $trust_{degree_p}$ sur les candidats arcs. Cette fonction permet de prendre en compte la propagation de la confiance des candidats sommets associés au candidat arc. Pour ces expérimentations nous avons sélectionné uniquement deux seuils représentatifs. Si nous comparons ces résultats avec ceux obtenus pour $trust_{likelihood}$ (tableaux 16, 17, 18) nous pouvons observer une nette amélioration pour les candidats Types (arcs étiquetés $instanceOf$). Nous pouvons aussi observer une amélioration des résultats pour les candidats relations même si cette amélioration est moins grande. Néanmoins les résultats concernant les labels sont moins bons que pour $trust_{likelihood}$. Nous pouvons alors déduire que la propagation de confiance d'un candidat sommet sur la confiance des candidats arcs associés améliore les résultats mais pas pour tous les types de candidats.

**Fonction de confiance : $trust_{choquet}$**

Nous utilisons maintenant la fonction de calcul de la confiance $trust_{choquet}$. Cette fonction, comme définie dans la section 4.4.3 du chapitre III, nécessite plusieurs paramètres à définir.

Tout d'abord nous définissons la qualité des sources utilisées. Pour cela nous réutilisons les scores de qualités définis avec les experts lors du processus d'analyse des sources. Nous obtenons alors les scores présentés dans le tableau 21.

| Agrovoc | TaxRef | NCBI |
|---|---|---|
| 0,6 | 0,9 | 0,8 |

TABLE 21 – Qualité des sources pour l'expérimentation qualitative

Nous retrouvons la source TaxRef qui est une référence nationale dans ce domaine, ce qui explique son score de qualité à 0,9. De part son processus de validation manuel et sa mise à jour régulière, la source NCBI a un score relativement élevé aussi : 0,8. La source Agrovoc quant à elle a un score de qualité de 0,6 puisque des travaux [Soergel et al., 2004] ont montré qu'elle contient un certain nombreux d'erreurs. Elle reste néanmoins une source intéressante, notamment par l'apport de ses labels.

Les autres paramètres à définir pour l'utilisation de l'intégrale de choquet sont le $x_0$ et le $\gamma$. Ces paramètres sont utilisés dans la fonction $\mu$ qui permet de représenter l'intérêt des sources dans un candidat (c.f. la section 4.4.3 du chapitre III). Le paramètre $x_0$ représente le point d'explosion de la courbe d'intérêt et le paramètre $\gamma$ la linéarité de la courbe de la fonction $\mu$. Nous définissons ici les valeurs $x_0 = \gamma = \frac{\sum_{i=1}^{N} Q(S_i)}{2} = \frac{0,6+0,9+0,8}{2} = 1.15$. Cette valeur permet une représentation standard de la fonction $\mu$.

Les résultats obtenus avec la fonction de calcul $trust_{choquet}$ montrent que cette fonction permet de prendre en considération les avantages des deux autres fonctions de calculs. Lorsque nous analysons les résultats présentés dans le tableau 22 nous pouvons observer une augmentation de la précision, ce qui permet de montrer que la confiance consensuelle

| **Seuil** | **Précision** | **Rappel** | **F-Mesure** |
|---|---|---|---|
| **0,1** | 0,87 | 0,98 | 0,92 |
| **0,2** | 0,99 | 0,63 | 0,77 |
| **0,3** | 0,99 | 0,63 | 0,77 |
| **0,4** | 1 | 0,62 | 0,77 |
| **0,5** | 1 | 0,60 | 0,75 |
| **0,6** | 1 | 0,60 | 0,75 |
| **0,7** | 1 | 0,26 | 0,41 |
| **0,8** | 1 | 0,26 | 0,41 |
| **0,9** | 1 | 0,26 | 0,41 |

TABLE 22 – Expérimentations $trust_{choquet}$ : individus

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,74 | 0,99 | 0,85 |
| **0,2** | 0,70 | 0,96 | 0,81 |
| **0,3** | 0,583 | 0,43 | 0,50 |
| **0,4** | 0,58 | 0,43 | 0,50 |
| **0,5** | 0,58 | 0,43 | 0,50 |
| **0,6** | 0,58 | 0,43 | 0,50 |
| **0,7** | 0,58 | 0,43 | 0,50 |
| **0,8** | 0,30 | 0,42 | 0,351 |
| **0,9** | 1 | 0,21 | 0,35 |

TABLE 23 – Expérimentations $trust_{choquet}$ : relations

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,45 | 1 | 0,62 |
| **0,2** | 0,39 | 0,98 | 0,56 |
| **0,3** | 0,48 | 0,45 | 0,46 |
| **0,4** | 0,48 | 0,45 | 0,46 |
| **0,5** | 0,48 | 0,45 | 0,46 |
| **0,6** | 0,48 | 0,45 | 0,46 |
| **0,7** | 0,48 | 0,45 | 0,46 |
| **0,8** | 0,23 | 0,43 | 0,30 |
| **0,9** | 0,56 | 0,37 | 0,44 |

TABLE 24 – Expérimentations $trust_{choquet}$ : types

| Seuil | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **0,1** | 0,74 | 0,79 | 0,77 |
| **0,2** | 0,69 | 0,78 | 0,73 |
| **0,3** | 0,49 | 0,60 | 0,54 |
| **0,4** | 0,49 | 0,60 | 0,54 |
| **0,5** | 0,49 | 0,60 | 0,54 |
| **0,6** | 0,49 | 0,60 | 0,54 |
| **0,7** | 0,49 | 0,60 | 0,54 |
| **0,8** | 0,63 | 0,54 | 0,58 |
| **0,9** | 0,45 | 0,18 | 0,26 |

TABLE 25 – Expérimentations $trust_{choquet}$ : labels

est là aussi bien représentée. La diminution du rappel nous permet d'observer l'effet discriminatoire de la considération des correspondances dans la fonction de calcul. Néanmoins cet effet est diminué ici puisque compensé par une implication de l'intérêt des sources. Cet effet est particulièrement remarquable sur le tableau 25 puisque la précision des candidats labels augmente au seuil 0.8. Cette augmentation n'était pas présente sur le tableau 18. Ceci nous permet de faire la distinction entre des labels présents dans Agrovoc et TaxRef par exemple et des labels présents dans TaxRef et NCBI. Ces deux dernières sources ayant un score de qualité plus important, elles apportent alors un intérêt supérieur. La source Agrovoc est néanmoins réputé pour ses labels. Il aurait été interessant ici de pouvoir spécifier l'implication d'une source en fonction du type pour pouvoir favoriser les labels provenant d'Agrovoc, et donc exploiter la spécificité de cette source. Nous reviendrons sur ce point lors des perspectives.

### 1.2.3. Analyse

L'analyse des résultats obtenus en utilisant les différentes fonction de calcul de la confiance nous a permis d'observer majoritairement trois aspects de cette confiance.

Tout d'abord l'évaluation de la fonction $trust_{likelihood}$ nous a permis de vérifier que notre hypothèse de départ, qui stipule que plus un candidat implique de sources plus il est susceptible d'être validé, est une bonne intuition. Nous pouvons ici affirmer que la représentation de la confiance consensuelle est un aspect important dans notre processus et permet d'obtenir un résultat de qualité. Le deuxième aspect important est que l'utilisation des correspondances dans le calcul de la confiance a un effet discriminant assez fort sur le score des candidats. Effectivement les candidats, même s'ils impliquent plusieurs sources, n'impliquent pas forcément beaucoup de correspondances. Ces correspondances apportent par contre un aspect de qualité supplémentaire au candidat. Nous pouvons observer ce phénomène dans les tableaux de résultats utilisant la fonction $trust_{degree}$. La fonction de calcul $trust_{choquet}$ considère les deux aspects présentés précédemment en donnant en plus la possibilité de paramétrer l'importance des sources. De cette manière la considération des correspondances est toujours forte mais elle est compensée par l'intérêt des sources impliquées dans le processus. L'utilisation de la fonction de calcul $trust_{degree_r}$ considérant le rayonnement permet d'améliorer les résultats sur certains types de candidats arc par rapport à la fonction $trust_{degree}$. Ce phénomène n'ayant pas été pris en compte dans $trust_{choquet}$ il serait intéressant de pouvoir appliquer ce rayonnement sur cette fonction de calcul.

Nous l'avons vu dans les résultats, les candidats labels ont des résultats assez différents des autres types de candidats. Ceci s'explique par le peu de consensus sur ces candidats n'apparaissant que dans peu de sources. C'est notamment le cas d'Agrovoc qui contient beaucoup de labels qu'on ne retrouve pas dans les autres sources. Il serait alors intéressant de faire varier la fonction d'intérêt des sources suivant le type de candidat considéré. Par exemple l'intérêt d'Agrovoc pour un candidat label est beaucoup plus important que pour un candidat relation. Nous verrons cet aspect plus en détail dans les perspectives.

## 1.3. Évaluation du score de confiance pour la génération d'une extension

Nous avons présenté dans la section précédente la qualité des candidats générés et l'impact de la fonction de calcul de la confiance consensuelle sur le filtrage des candidats. Nous n'avons pas considéré dans ces calculs la génération d'une extension et l'intérêt de l'utilisation d'une fonction de confiance consensuelle dans un processus de validation des candidats.

Nous allons présenté ici nos expérimentations concernant la création d'une extension sur le domaine de la taxonomie des blés. Pour cela nous générons une extension et nous demandons à un expert de valider chaque candidat sommet contenu dans l'extension. Si un des candidats sommets de l'extension n'est pas validé par l'expert alors une nouvelle extension est recherchée en imposant d'inclure tous les candidats déjà validés et en excluant tous les candidats invalidés. Le processus s'arrête lorsque l'expert a validé tous les candidats d'une extension. Nous allons montrer qu'un expert a besoin de moins d'interactions (action de valider ou invalider un candidat) pour trouver l'extension optimale (extension dont tous les candidats sont validés) lorsque nous prenons en compte le calcul de confiance des candidats. Cette confiance est utilisée dans la fonction objective pour trouver une extension mais aussi pour trier les candidats à valider dans une extension. Ce tri permet de présenter à l'expert en premier les candidats le plus susceptibles d'être validés. Dans ces expérimentations nous allons considérer uniquement la fonction de calcul $trust_{choquet}$.

La génération d'une extension étant un processus qui prend d'autant plus de temps qu'il y a d'incompatibilités il ne nous a pas été possible d'effectuer cette évaluation sur la totalité des candidats sommets générés. Effectivement nous avons générés au total 212 candidats sommets, avec environ 1500 incompatibilités entre ces candidats. La génération d'une extension a duré 6h sur une machine standard [105] et a permis d'obtenir une extension contenant 17 candidats. Néanmoins ce processus devant être exécuté chaque fois qu'un candidat n'est pas validé par l'expert, il n'était pas envisageable d'évaluer la génération d'une extension sur la totalité des candidats sommets générés.

Nous avons fais le choix de ne considérer que les candidats sommets ayant un score de confiance supérieur à 0.45. De cette manière nous obtenons 100 candidats avec 1150 incompatibilités et une extension est calculée en moins de 3 minutes.

Le tableau 26 permet d'observer les résultats obtenus lors de nos expérimentations. La première ligne considère la génération d'une extension avec comme fonction objective la maximisation du nombre de candidats. La deuxième ligne présente les résultats en utilisant la fonction objective maximisant la somme des scores de confiances des candidats comme présenté dans la section 4.5.4 du chapitre III.

Le tableau 26 présente, pour les deux fonctions objectives définies, d'abord le nombre de candidats dans l'extension maximum $ext_{max}$. Cette extension est la première extension trouvée par le système, elle représente l'extension qui maximise la fonction objective sans considération de validation de candidats. En d'autres termes, si nous ne faisions pas l'étape de validation avec l'expert, c'est cette extension que nous aurions considérée. Ce

105. Core I7, 8Go DDR3

| Fnct Obj. | $\|ext_{max}\|$ | $\|ext_{opti}\|$ | Nb interactions |
|---|---|---|---|
| Nb. Cands | 10 | 7 | 55 |
| Confiance | 10 | 7 | 25 |

TABLE 26 – Résultats des expérimentations sur la recherche d'une extension optimale

tableau présente aussi le nombre de candidats présents dans l'extension optimale $ext_{opti}$. Cette extension est celle obtenue à la fin du processus de validation avec l'expert. Nous savons que cette extension contient uniquement des candidats validés par l'expert. Enfin le tableau présente le nombre d'interactions qui ont été nécessaire à l'expert pour obtenir l'extension optimale. Nous considérons ici une interaction comme l'action de définir qu'un candidat est valide ou invalide.

Nous pouvons observer qu'en utilisant les deux fonctions objectives nous obtenons la même extension maximale, contenant 10 candidats. Nous pouvons remarquer aussi que l'extension optimale est aussi la même avec 7 candidats. Cette similarité est moins étonnante puisque ce sont les candidats validés par l'expert. Par contre nous pouvons voir que l'extension optimale contient 3 candidats de moins que l'extension maximale. Ceci est due au fait que 3 des candidats de l'extension maximale et tous les candidats incompatibles à ces candidats n'ont pas été validés par l'expert. Ce qui signifie qu'un filtre sur le score de confiance et la considération des incompatibilités ne sont pas suffisant pour garantir des candidats valides. Il est toujours nécessaire de passer par une phase de validation manuelle par un expert pour s'assurer de la qualité du résultat.

Ce qui nous intéresse particulièrement dans le tableau 26 ce sont les nombres d'interactions qui ont été nécessaire à l'expert pour découvrir l'extension optimale. Nous considérons ici un ensemble de 100 candidats avant la recherche d'extension. Si nous avions demandé à un expert de valider ces candidats sans considérer les incompatibilités, celui-ci aurait eu besoin de 100 interactions pour tous les valider. En considérant les incompatibilités et une fonction objective maximisant le nombre de candidats dans l'extension, l'expert n'a eu besoin que de 55 interactions. Ce sont donc 45% des candidats qui n'ont pas eu besoin d'être validé puisqu'ils sont incompatibles avec les candidats validés. La phase de validation demandée à l'expert est donc particulièrement simplifiée. Si nous observons ensuite le nombre d'interactions nécessaire lors d'une utilisation d'une fonction objective maximisant la somme des scores de confiances, ce nombre d'interactions passe à 25. En ne validant que 25% des candidats, l'extension optimale a été découverte. Cette extension étant la même que l'extension obtenue avec une fonction objective maximisant le nombre de candidats, nous ne perdons pas en qualité du résultat obtenu.

En observant les résultats obtenus lors de ces évaluations nous pouvons en conclure trois aspects importants sur la qualité des candidats générés. Tout d'abord nous avons observé que le filtrage des candidats sur le score de confiance et la considération des incompatibilités ne sont pas suffisants pour valider les candidats. Il est nécessaire de passer par une validation manuelle pour obtenir un bon résultat. Ensuite nous avons vu que l'utilisation des incompatibilités a permis de diminuer le nombre d'interactions nécessaires pour obtenir un ensemble de candidats compatibles tous validés. La considération des

incompatibilités permet ici de simplifier le travail de validation de l'expert. Le dernier aspect est l'utilisation des scores de confiance dans la découverte de l'extension. Cette considération a permis de diminuer de moitié le nombre d'interactions qui ont été nécessaire à l'expert pour trouver l'extension optimale. Ceci permet de montrer que le score de confiance calculé permet de donner une bonne intuition sur la validité d'un candidat.

## 2. Expérimentations sur un jeu de données issu du Web de données liées

Après avoir expérimenté notre approche sur le domaine de la taxonomie des blés, nous avons souhaité montrer la généricité de notre approche. Pour cela, nous avons appliqué notre méthodologie à un domaine totalement différent. Nous avons voulu analyser les résultats de notre approche sur un jeu de données plus volumineux que la taxonomie des blés et qui n'aurait pas été construit dans l'objectif d'être utilisé dans notre approche. De plus, nous considérons des sources qui sont déjà des bases de connaissances et non des sources non-ontologiques.

Pour cela, nous avons utilisé le jeu de données fourni pour la campagne d'évaluation des systèmes d'alignement OAEI. Cette campagne d'évaluation comprend une tâche dont l'objectif est d'évaluer et de comparer les systèmes d'alignement en analysant la pertinence des correspondances générées par les systèmes pour reformuler des requêtes SPARQL. L'idée est de s'appuyer sur les correspondances générées par le système pour traduire une requête SPARQL posée initialement sur une première base en une requête SPARQL équivalente posée sur une deuxième base. Notre objectif n'est pas de nous positionner par rapport aux systèmes d'alignements car notre méthodologie a un but différent. Nous cherchons à montrer que notre méthodologie permet de fusionner les bases de connaissances considérées et de retrouver en une requête SPARQL sur la base résultant de la fusion l'ensemble des réponses présentes dans les différentes bases initiale. Dans un premier temps, nous allons présenter la campagne d'évaluation des systèmes d'alignement. Nous présenterons ensuite comment nous avons pu appliquer notre proposition au jeu de données de cette campagne. Nous finirons cette section en présentant la stratégie d'évaluation et les résultats obtenus par Muskca dans ce contexte.

### 2.1. Présentation de la tâche QA4OA de l'OAEI

Nous avons considéré la campagne d'évaluation de systèmes d'alignements OAEI [106]. Cette campagne propose, depuis 2005, une méthode pour évaluer les systèmes d'alignement d'ontologies. Plusieurs tâches permettent de tester plusieurs aspects d'un système d'alignement. Nous retrouvons par exemple des tâches spécialisées dans les systèmes d'alignements multilingues (Multifarm) ou pour l'alignement d'individus (Instance Matching). Nous nous sommes particulièrement intéressé à la tâche QA4OA (Question Answering for Ontology Alignment - https://www.cs.ox.ac.uk/isg/projects/Optique/oaei/oa4qa/).

106. Ontology Alignment Evaluation Initiative - http://oaei.ontologymatching.org/

Cette tâche propose une nouvelle forme d'évaluation depuis 2014. Les autres tâches proposant de comparer les correspondances générées par les systèmes avec des correspondances de références (définies manuellement), celle-ci préfère passer par un système d'interrogation. La tâche considère sept sources ou bases de connaissances et 18 requêtes SPARQL chacune écrite pour au moins une des sept sources. Les organisateurs de la tâche ont développé un système de reformulation de requêtes prenant en entrée une requête SPARQL écrite pour une base de connaissances et un ensemble de correspondances établies entre cette base et une deuxième (ces correspondances étant générées par les systèmes d'alignement concourant dans la tâche). Le système fournit en sortie une nouvelle requête à exécuter sur la deuxième base, le contenu de cette requête ayant été obtenu automatiquement à partir des correspondances fournies en entrée. Les résultats retournés par chacune des requêtes sont ensuite comparés aux résultats de référence. Ces résultats de référence sont obtenus en utilisant ce même système de reformulation, mais en prenant cette fois-ci des correspondances de références générées manuellement.

Cette tâche permet de prendre en considération des correspondances qui ne seraient pas identiques aux correspondances de référence, mais qui permettraient tout de même d'obtenir des résultats équivalents par des requêtes.

Prenons par exemple la première requête SPARQL proposée dans la tâche :

```
SELECT DISTINCT ?x
WHERE {
 ?x rdf:type [
   a owl:Class;
owl:unionOf (
         confof:Trip
         confof:Banquet
         confof:Reception
)
   ] .
  ?x rdf:type owl:NamedIndividual.
}
```

Cette requête propose de récupérer tous les individus de type "Trip", "Banquet" ou "Reception" de la base de connaissances "confof".

Considérons qu'un système d'alignement établit la correspondance suivante entre la base "confof et la base "ekaw".

(au format RDF Alignment [107]) :

```
<map>
  <Cell>
    <entity1 rdf:resource="http://confOf#Banquet"/>
    <entity2 rdf:resource="http://ekaw#Conference_Banquet"/>
    <measure rdf:datatype="xsd:float">1.0</measure>
```

107. http://alignapi.gforge.inria.fr/format.html

```
    <relation>=</relation>
  </Cell>
</map>
```

La requête réécrite pour être adaptée à la source "ekaw" sera alors :

```
SELECT DISTINCT ?x
WHERE {
 ?x rdf:type [
   a owl:Class;
owl:unionOf (
         confof:Trip
         ekaw:Conference_Banquet
         confof:Reception
)
   ] .
  ?x rdf:type owl:NamedIndividual.
}
```

Afin d'évaluer les résultats obtenus pour un système d'alignement, la campagne propose d'interroger chaque couple de sources en exploitant les alignements entre ces deux sources. Le résultat de la requête obtenu en utilisant les correspondances de référence sont comparés aux résultats obtenus avec les correspondances générées. Un calcul de précision, rappel et f-mesure est effectué pour chaque couple de source et pour chaque requête.

Il est possible d'accéder aux résultats des systèmes d'alignement ayant participé à cette tâche en 2014 à l'adresse suivante : https://www.cs.ox.ac.uk/isg/projects/Optique/oaei/oa4qa/2014/results.html. Nous pouvons remarquer que LogMap (qui est le système que nous utilisons dans Muskca) est celui qui a obtenu les meilleurs résultats.

## 2.2. Mise en place du cadre expérimental

Pour utiliser cette campagne d'évaluation avec notre méthodologie, nous avons besoin de définir plusieurs éléments. Il est nécessaire de proposer la partie haute d'un module ontologique permettant de définir les bornes du domaine et la modélisation souhaitée. Cette partie haute doit pouvoir couvrir les éléments qui sont interrogés dans les requêtes utilisées dans la campagne. Après avoir présenté cette partie haute d'un module ontologique, nous présenterons la transformation des sept sources de la campagne en fonction de ce module pour obtenir les SKB associées. Enfin, nous présenterons le paramétrage de Muskca pour l'utiliser sur ce jeu de données ainsi que les résultats obtenus

### 2.2.1. Module ontologique pour la tâche QA4OA - OAEI

Pour construire la partie haute du module ontologique, nous avons utilisé les 18 requêtes de la campagne comme spécifications. Ces questions représentent les besoins de la base

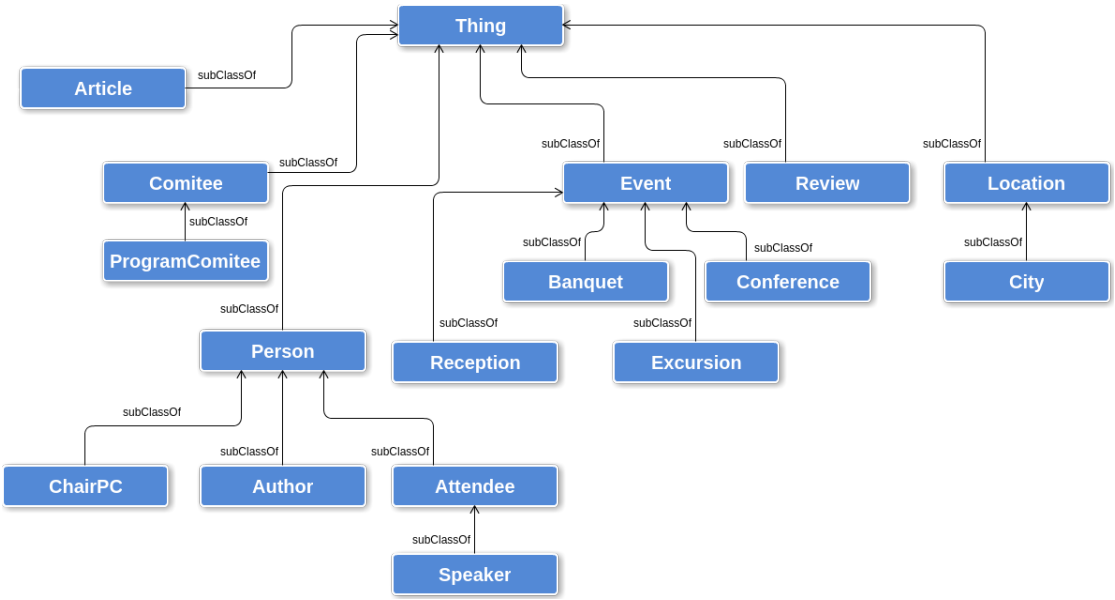


FIGURE 55 – Partie haute du module ontologique pour la tâche QA4OA - OAEI

de connaissances finale sous la forme de questions auxquelles la base de connaissances devra répondre (Cf. section 3 du chapitre I). La partie haute du module ontologique est présentée dans la figure 55.

Nous n'avons pas représenté les propriétés sur ce schéma mais, d'après les requêtes considérées, uniquement deux sont nécessaires :

**hasAuthor :** Permet de spécifier qu'un article est écrit par un auteur

**reviewWrittenBy :** Permet de spécifier qu'une review a été écrite par une personne

### 2.2.2. Sources

Les sept sources proposées dans cette campagne concernent toutes le même domaine : les conférences. Chacune des sources répertorie les connaissances liées à une conférence ou un ensemble de conférences sur l'informatique.

Voici la liste des sept sources présentes dans la campagne :

**Cmt :** http://msrcmt.research.microsoft.com/cmt

**ConfOf :** http://www.conftool.net/

**Edas :** http://edas.info/

**Ekaw :** http://ekaw.vse.cz/

**Iasted :** http://iasted.com/conferences/2005/cancun/ms.htm

**Sigkdd :** http://www.acm.org/sigs/sigkdd/kdd2006

**Conference :** Ontologie créee à la main et peuplée à partir de DBLP [108]

Nous n'avons pas besoin d'effectuer une transformation syntaxique puisque les sources sont déjà en RDF. Néanmoins, il est nécessaire d'effectuer une extraction et un changement de la modélisation pour que ces sources soient adaptées à la partie haute du module ontologique défini précédemment. Pour cela, nous avons défini un patron de transformation générique pour toutes les sources. Celui-ci est disponible à l'adresse suivante : https://github.com/Murloc6/OAEIConf2RKB. Cet algorithme, applicable aux bases de connaissances, prend un ensemble de correspondances entre la source et le module et extrait toutes les spécialisations de ces correspondances. En d'autres termes, seront extraits grâce à ce patron toutes les classes qui sont des spécialisations des classes mises en correspondance, tous les individus de ces classes, tous les labels associés aux classes et aux individus extraits et toutes les relations sélectionnées qui relient les individus extraits.

Le tableau 27 présente la quantité d'éléments extraits de chaque source.

| Source | Classes | Individus | Relations | Labels |
|---|---|---|---|---|
| Cmt | 31 | 1581 | 293 | 1612 |
| ConfOf | 17 | 960 | 96 | 977 |
| Edas | 16 | 1049 | 69 | 1065 |
| Ekaw | 46 | 1157 | 108 | 1202 |
| Iasted | 57 | 1445 | 290 | 1502 |
| Sigkdd | 21 | 1248 | 567 | 1269 |
| Conference | 24 | 1350 | 69 | 1374 |

TABLE 27 – Nombre d'éléments extraits des sources de la campagne QA4OA

Nous pouvons observer qu'il y a autant de labels dans chaque source que d'éléments ontologiques (classes + individus). Ceci s'explique par une action supplémentaire qui a été effectuée par le patron de transformation. Aucun label n'était défini a priori dans les sources de la campagne QA4OA. Comme les systèmes d'alignement utilisent ces labels pour faciliter l'alignement, nous avons récupéré le fragment d'URI de chaque élément ontologique que nous avons ajouté comme label de l'élément. Considérer ces labels nous permet depuis de pouvoir générer des candidats labels par notre approche.

Prenons par exemple l'URI d'un individu de type "Person" provenant de la source Ekaw :

```
http://xmlns.com/foaf/0.1#AnastasiaAnalyti
```

Nous retrouvons ici la base URI qui est : "xmlns.com/foaf/0.1#". Cette base permet de définir l'ontologie FOAF qui a été utilisée ici pour définir une personne. Le fragment d'URI de cette URI est : "AnastasiaAnalyti". Si nous appliquons notre extraction de terme fondée sur la syntaxe du "Camel Case" nous pouvons ajouter le label suivant :

108. http://dblp.uni-trier.de/db/

| Candidats sommets | | Candidats arcs | | |
|---|---|---|---|---|
| 4818 | | 18330 | | |
| **Individus** | **Classes** | **Labels** | **Relations** | **Types** |
| 4669 | 149 | 9461 | 2933 | 5936 |

TABLE 28 – Nombre de candidats générés QA4OA

"Anastasia Analyti". De cette manière, nous ajoutons un label pour chaque élément ontologique extrait des sources.

Nous pouvons également constater, à partir du tableau 27, que les individus sont le type d'éléments le plus présent dans ces bases. Nous avons donc repris LogMap comme système d'alignement utilisé par notre approche. Afin d'analyser a priori les éléments partagés par les sources, nous avons analysé le nombre de correspondances établies par cet outil entre les différentes source. Nous avons constaté que peu de correspondances sont établies entre les individus des différentes sources alors que la quasi-totalité des classes sont alignées. C'est par exemple le cas entre les sources EKAW et Iasted dont LogMap n'a découvert que 297 correspondances entre individus et 36 correspondances entre classes. Si nous comparons ce nombre de correspondances par rapport au nombre d'éléments extraits des sources dans le tableau 27, le ratio d'individus communs entre les deux sources est très faible. Ceci met en évidence que sur ce jeu de données, peu d'individus sont partagés par les sources. Ceci s'explique par le fait que chaque source est consacrée à la représentations des données liées à des conférences différentes ou à la même conférence mais ayant eu lieu sur des années différentes. A priori, notre approche qui exploite les éléments communs des sources ne sera pas dans ce cadre-là testée sur un jeu de données idéal. En revanche, nous pensons que cette évaluation a un intérêt car à l'échelle du LOD, ce cas de figure est fréquent puisque les jeux de données publiés reposent sur des vocabulaires similaires mais qui ont chacun pour objectif de représenter des jeux de données spécifiques.

### 2.2.3. Paramétrage de Muskca

Grâce à ces bases de connaissances sources générées précédemment, nous pouvons utiliser l'outil Muskca pour générer des candidats. Pour cela, il est nécessaire de définir les paramètres pour l'utilisation de la fonction de calcul du score de confiance $trust_{choquet}$. Les sources présentent toutes le même intérêt. Nous définissons donc le score de qualité de toutes les sources à 1. De la même manière, nous définissons les paramètres $x_0$ et $\gamma$ avec des valeurs standards. Ces valeurs permettent une représentation standard de la fonction d'intérêt des sources $\mu$. Toutes les sources ayant un intérêt égal, l'évolution de l'intérêt des sources en fonction du nombre de sources impliquées est elle aussi standard. Pour cela, nous utilisons la valeur médiane de la somme des qualité des sources. Nous obtenons alors : $x_0 = \gamma = \frac{7}{2} = 3,5$.

Le tableau 28 présente le nombre de candidats générés à partir du jeu de données proposé dans la campagne d'évaluation QA4OA. Nous pouvons observer que ce jeu de

données nous permet de générer un plus grand nombre de candidats que le jeu de données sur la taxonomie des blés.

Dans ce paragraphe, nous ne considérerons pas la génération d'extension car ce processus est particulièrement long lorsqu'il y a beaucoup d'incompatibilités. Sur les 4818 candidats sommets générés sur ce jeu de données, 241.951 incompatibilités entre candidats ont été détectées. Notre processus de génération d'extension ne permet pas de traiter une quantité aussi importante d'incompatibilités dans un temps raisonnable.

## 2.3. Stratégie de validation

Afin d'utiliser la campagne d'évaluation QA4OA pour valider notre approche sur un grand jeu de données, nous devons adapter les méthodes de calculs de cette campagne pour l'appliquer à notre résultat qui est ici un ensemble de candidats. La campagne propose un calcul de précision, rappel et f-mesure sur les résultats obtenus pour les 18 requêtes proposées. Ces requêtes permettent d'interroger les bases de connaissances en exploitant les correspondances. Nous verrons dans cette section comment nous avons adapté cette interrogation à l'interrogation de candidats. Nous présenterons ensuite la façon dont nous avons adapté le jeu de données de références pour prendre en considération toutes les sources et plus seulement un couple de source. Enfin, nous présenterons la modification apportée aux fonctions de calcul de précision et rappel pour les adapter à l'étude des candidats.

### 2.3.1. Adaptation des requêtes

Les requêtes proposées dans la campagne sont spécifiques aux éléments présents dans les sources. Nous souhaitons utiliser ces requêtes pour interroger la base de connaissances finale générée par notre approche. Il est alors nécessaire d'adapter ces requêtes pour qu'elles soient exécutables sur notre base.

Prenons l'exemple de la première requête proposée dans la campagne :

```
SELECT DISTINCT ?x
WHERE {
 ?x rdf:type [
   a owl:Class;
owl:unionOf (
         confof:Trip
         confof:Banquet
         confof:Reception
)
   ] .
  ?x rdf:type owl:NamedIndividual.
}
```

Cette requête permet d'obtenir l'ensemble des individus qui sont de type "confof :Trip" ou "confof :Banquet" ou encore "confof :Reception". Dans le cadre de notre approche,

des classes équivalentes ont été créées dans la partie haute du module ontologique et sont liées par des relations d'équivalence "owl :sameAs" aux classes des sources. Si les classes utilisées dans une requête ne sont pas présentes dans le module, elles peuvent apparaître dans un candidat classe. Dans les deux cas, ces classes sont liées avec la relation "owl :sameAs" vers la classe du module [109]. Nous devons donc modifier la requête SPARQL pour changer les liens vers les classes recherchées en exploitant ces relations "owl :sameAs". La première requête de la campagne devient alors :

```
SELECT DISTINCT ?x
WHERE {
 ?x rdf:type [
   a owl:Class; [owl:sameAs
             owl:unionOf (
              confof:Trip
              confof:Banquet
              confof:Reception
             )]
   ] .
  ?x rdf:type owl:NamedIndividual.
}
```

Cette requête permet d'obtenir tous les individus dont le type est une classe qui a la relation "owl :sameAs" avec une des trois classes de la source "confOf". De cette manière, nous pouvons obtenir l'ensemble des résultats contenus dans la base de connaissances finale.

Ces requêtes permettent d'obtenir un ensemble de candidats. Par exemple ici, l'ensemble des individus de ce type sont des candidats individus, résultats de la fusion des individus des différentes sources. Nous exploitons, là aussi, les relations "owl :sameAs" présentes sur ces candidats individus pour obtenir les URI des éléments qui étaient présents dans les sources. De cette manière, nous obtenons l'ensemble $V_{CandS}$ des candidats sommets retournés. Ces éléments ontologiques nous serviront pour comparer les résultats avec les résultats de référence.

### 2.3.2. Adaptation du jeu de données de référence

La campagne est prévue pour comparer les performances des systèmes d'alignement entre deux sources. Le résultat d'une requête est donc donné en fonction du couple de sources étudiées et des correspondances entre ces sources. Le jeu de données de référence est prévu pour évaluer ce type de résultat. Pour adapter ce jeu de données de référence, nous fusionnons les résultats attendus entre tous les couples de sources et nous supprimons les doublons qui apparaissent.

Si nous considérons par exemple la première requête de la campagne présentée précédemment, nous récupérons les résultats attendus par le système pour le couple de

109. Nous utilisons ici la sortie au format OWL présenté dans la section 5.2 du chapitre III

source Cmt et ConfOf. Nous faisons la même chose pour Cmt et Edas et nous fusionnons les résultats en supprimant les doublons. Nous appliquons ce processus jusqu'à avoir considéré tous les couples de sources présents dans cette campagne.

Ce processus permet d'obtenir les résultats attendus pour chaque requête pour toutes les sources simultanément. Ce jeu de données de référence est donc adapté aux résultats obtenus par Muskca.

### 2.3.3. Adaptation des fonctions de calculs

Nous avons vu comment nous adaptons les requêtes pour qu'elles permettent d'interroger notre base de connaissances finale. Nous avons ensuite présenté l'adaptation du jeu de données de référence pour qu'il permette la considération de toutes les sources simultanément. Nous allons présenter maintenant les fonctions de calculs que nous avons utilisées pour calculer la précision et le rappel sur ces requêtes adaptées et en fonction du jeu de données de référence adapté.

La précision permet d'évaluer la pertinence des résultats à partir du ratio entre le nombre d'éléments générés par le système qui sont aussi présents dans le jeu de données de référence et le nombre d'éléments dans le résultat. Nous étendons ce calcul de la précision en ajoutant une considération de la redondance des résultats. Chaque candidat retourné par la requête adaptée à Muskca est composé de plusieurs éléments ontologiques. Lors de l'extraction de ces éléments ontologiques pour obtenir la liste des URI à comparer avec le jeu de données de référence, il est possible qu'il y ait des URI redondantes. En effet, des éléments ontologiques peuvent apparaître dans plusieurs candidats (dans le cas d'incompatibilités). Afin de prendre en compte cette redondance dans le calcul de la précision, nous considérons le ratio entre les URI distinctes du résultat présents dans le jeu de données de référence sur le nombre total d'URI du résultat. De cette manière, si un élément ontologique est présent dans plusieurs candidats, il ne sera comptabilisé qu'une seule fois au numérateur mais plusieurs fois au dénominateur.

Le rappel et la f-mesure sont calculés de façon standard.

L'ensemble du processus d'expérimentation est schématisé dans la figure 56.

## 2.4. Résultats

Le tableau 29 présente les résultats obtenus avec les différentes requêtes de la campagne d'évaluation QA4OA. Nous pouvons observer sur ce tableau que la plupart des requêtes ont des scores très elevés. Ceci s'explique par le fait que nous avons réutilisé les requêtes de la campagne pour générer notre module et que nous avons ensuite effectué des alignements manuellement entre ce module et les sources considérées. Il est donc normal que nous obtenions des résultats aussi élevés. Néanmoins, nous pouvons aussi observer qu'en utilisant une source intermédiaire (ici la partie haute du module ontologique) et un alignement manuel uniquement sur cette source, nous obtenons de très bons résultats. Cet aspect est particulièrement intéressant pour une application comme le Web de données liées qui se développe rapidement au fil des années. Ce Web de données liées encourage la réutilisation de vocabulaires pour définir de nombreuses données factuelles. L'ontologie

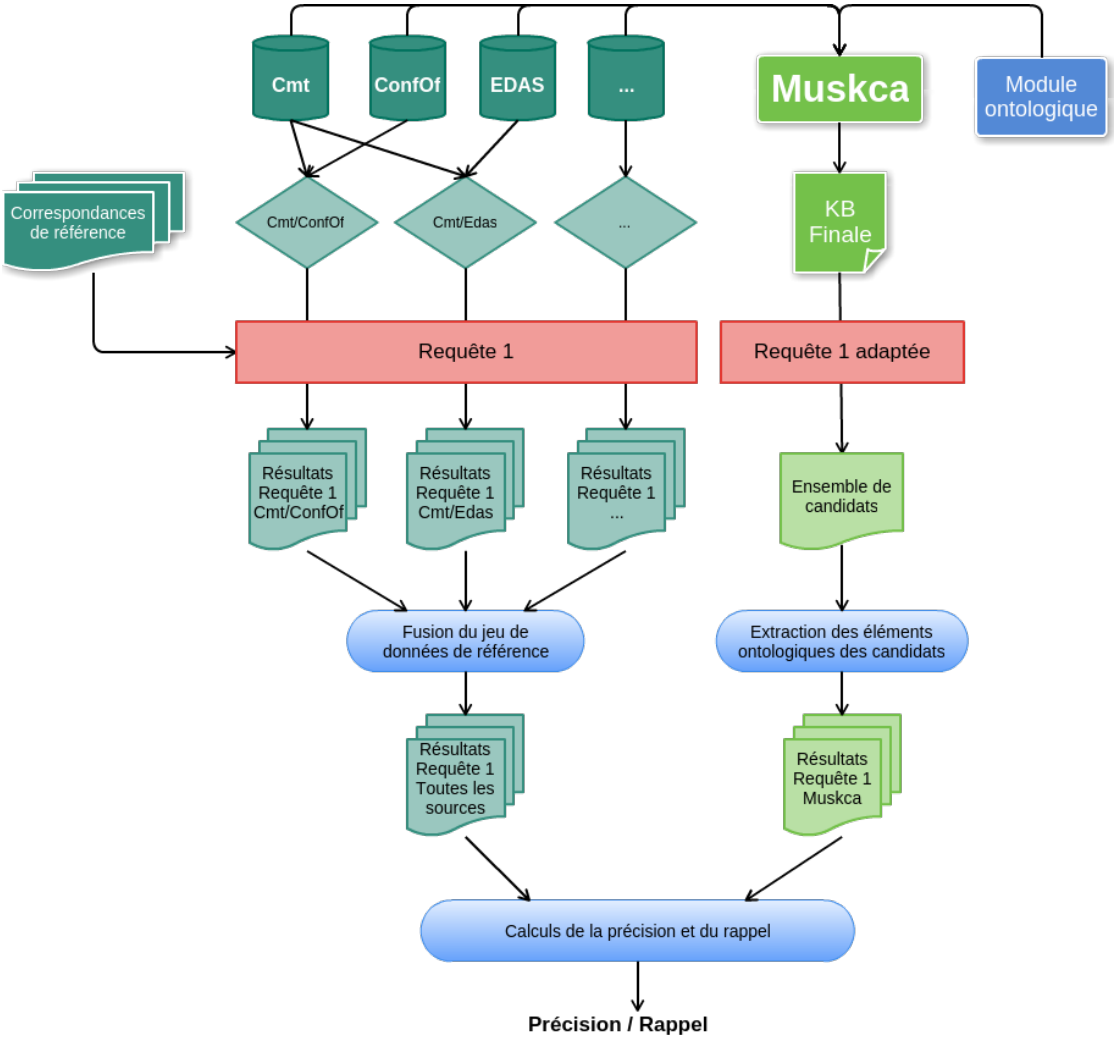


FIGURE 56 – Stratégie d'expérimentation de Muskca sur la campagne d'évaluation QA4OA

| Requête | Précision | Rappel | F-Mesure |
|---|---|---|---|
| **qa1** | 0,99 | 1 | 0,99 |
| **qa2** | 1 | **0,33** | 0,5 |
| **qa3** | **0,04** | 1 | 0,08 |
| **qa4** | 1 | 1 | 1 |
| **qa5** | 1 | 0,99 | 0,99 |
| **qb1** | 1 | 1 | 1 |
| **qb2** | 1 | 1 | 1 |
| **qb3** | **0,08** | 1 | 0,16 |
| **qb4** | 1 | 1 | 1 |
| **qb5** | 1 | 1 | 1 |
| **qb6** | **0,5** | 1 | 1 |
| **qv1** | **0,75** | 0,99 | 0,85 |
| **qv2** | 1 | 0,99 | 0,99 |
| **qv3** | 1 | 1 | 1 |
| **qv4** | 1 | 1 | 1 |
| **qv5** | 1 | 0,99 | 0,99 |
| **qv6** | 1 | 0,99 | 0,99 |
| **qv7** | 1 | 0,99 | 0,99 |

TABLE 29 – Résultats de l'évaluation de Muskca sur la campagne d'évaluation QA4OA

Foaf est par exemple largement utilisée sur le Web de données liées. Il est donc de plus en plus courant de trouver sur le Web de données liées des sources ayant des vocabulaires (ou ontologies) similaires mais des données factuelles différentes.

Un second fait notable sur ce tableau réside dans les résultats des requêtes qa3 et qb3 qui ont des précisions particulièrement basses bien que les rappels soient à 1. En observant les résultats, nous avons pu remarquer que cette précision basse est en grande partie due à la redondance des URI récupérées. En d'autres termes, les candidats récupérés par la requête qa3 et qb3 partagent un grand nombre d'éléments ontologiques. Ces deux requêtes permettent de récupérer des "Reviews", l'une par son type (qb3) et l'autre par la relation "reviewWrittenBy" (qa3). Comme pour les autres éléments ontologiques des différentes sources, les reviews n'ont pas de labels définis dans les sources. De plus, les fragments d'URI que nous avons utilisés pour générer des labels sont de type "review1", "review2", etc. De ce fait, le système d'alignement génère beaucoup de correspondances erronées, ce qui implique la génération de beaucoup de candidats qui partagent des éléments ontologiques. Si nous observons les résultats pour ces deux requêtes en utilisant des seuils pour filtrer les candidats, nous obtenons les résultats présentés dans le tableau 30.

Ce tableau permet d'observer que la précision augmente significativement lorsque nous filtrons les candidats. La requête qa3 utilise la relation "reviewWrittenBy" pour récupérer l'ensemble des Reviews des sources. Dans certaines sources, cette relation n'est pas définie aussi précisément. L'auteur d'une Review est représenté par une relation du

| | qa3 | | | qb3 | | |
|---|---|---|---|---|---|---|
| Seuils | Précision | Rappel | F-Mesure | Précision | Rappel | F-Mesure |
| **0,05** | 0,04 | 0,95 | 0,08 | 0,08 | 0,97 | 0,15 |
| **0,1** | 0,04 | 0,95 | 0,08 | 0,08 | 0,97 | 0,15 |
| **0,15** | 0,15 | 0,88 | 0,26 | 0,29 | 0,84 | 0,43 |
| **0,2** | 0,33 | 0,19 | 0,24 | 0,6 | 0,17 | 0,27 |
| **0,25** | **0,5** | 0,015 | 0,02 | **1** | 0,015 | 0,03 |
| **0,3** | 0 | 0 | 0 | 0 | 0 | 0 |

TABLE 30 – Résultats pour les requêtes qa3 et qb3 avec seuils de filtrage

type "hasAuthor" qui est plus générique et qui peut être aussi utilisée pour représenter l'auteur d'un article. De ce fait, l'ensemble des résultats récupérés par la requête qa3 ne sont pas bons en raison de cette ambiguïté dans certaines sources, ce qui explique qu'avec un seuil de 0,25, la précision n'est que de 0,5 alors qu'elle est de 1 pour la requête qb3. Cette requête qb3, quant à elle, ne contient aucun mauvais résultat, même sans filtrage des candidats. La précision basse est uniquement due aux doublons présents dans le partage des éléments ontologiques entre candidats, comme nous l'avons expliqué précédemment. L'utilisation des incompatibilités entre candidats peut ici permettre une augmentation significative de la précision pour ces deux requêtes.

Nous pouvons aussi remarquer qu'avec un seuil à 0,3, il n'y a plus aucun résultat pour aucune des deux requêtes. Ce phénomène est observable pour toutes les autres requêtes dès le seuil est à 0,05. Cette absence de résultat peut s'expliquer par le fait que les sources ne partagent que très peu, voire pas du tout, d'éléments.

## 2.5. Analyses

Nous avons évalué les résultats obtenus par Muskca sur la campagne d'évaluation QA4OA. Cette campagne d'évaluation des systèmes d'alignement a été adaptée pour pouvoir évaluer les résultats obtenus par Muskca. Nous avons pu observer de bons résultats pour un grand nombre de requêtes grâce à l'utilisation du module ontologique. L'utilisation d'un module ontologique permet de simplifier le processus de fusion en guidant les alignements grâce au module. Le jeu de données de cette campagne d'évaluation, de par son objectif d'évaluer les systèmes d'alignement, ne comporte pas beaucoup d'éléments communs entre les différentes sources. Néanmoins certains éléments communs à toutes les sources ont pu être fusionnés et nous avons pu observer l'intérêt du score de confiance sur ces candidats. Cette évaluation nous a permis de montrer la généricité de Muskca, qui peut être utilisé dans d'autres domaines que l'agriculture mais aussi dans un processus de fusion de bases de connaissances. Nous avons également pu observer que Muskca peut être utilisé sur de gros jeux de données, bien que la génération des extensions nécessite encore une optimisation. Nous avons pu observer dans ces évaluations que, même dans un cas où peu d'éléments sont partagés entre les sources, la recherche dans les bases reste de bonne qualité. En effet, nos résultats avec des seuils relativement bas obtiennent une

précision et un rappel élevés.

## 3. Conclusion

Nous avons dans ce chapitre évalué notre approche. Nous avons tout d'abord pu valider l'approche sur un processus de génération d'une base de connaissances à partir de plusieurs sources non-ontologiques. En effet, nous avons pu, grâce à l'aide de trois experts du domaine, générer une base de connaissances sur la taxonomie des blés. Notre approche peut être appliquée pour toute la taxonomie des plantes si nous changeons l'élément limitant dans la transformation des sources. Nous avons ensuite validé notre hypothèse de départ qui stipule qu'un candidat impliquant plusieurs sources est de meilleure qualité qu'un candidat présent dans une seule source. L'utilisation d'un score de confiance représentant le consensus sur ce candidat a permis de valider cette hypothèse. Nous avons aussi montré que l'utilisation des correspondances dans le calcul de la confiance consensuelle peut être très discriminatoire dans des jeux de données pour lesquels peu de correspondances sont établies. Cette discrimination peut être atténuée en considérant l'intérêt des sources dans le processus. L'utilisation d'un score de confiance dans un processus de validation des candidats d'une extension a permis de diminuer de moitié le nombre d'interactions nécessaires par un expert pour obtenir une extension optimale. Ceci permet de montrer l'intérêt du calcul de confiance d'un candidat et la considération des incompatibilités dans le processus de validation. Finalement, nous avons montré que notre approche peut être appliquée dans d'autres contextes, notamment pour la fusion d'un jeu de données d'une taille plus conséquente et sur un autre domaine que l'agriculture. Mais elle peut aussi être utilisée dans le cas de la fusion de plusieurs sources ontologiques. Nous avons aussi pu observer que le résultat obtenu par notre approche permet de garder suffisamment d'éléments pour assurer une recherche de qualité, même dans le cas où peu d'éléments sont partagés entre les sources.

Plusieurs perspectives ressortent de ces évaluations. Nous avons tout d'abord vu que l'utilisation de l'intérêt des sources dans le calcul de la confiance est particulièrement intéressante. Néanmoins, cet intérêt peut être variable suivant les types de candidats considérés. Nous pouvons par exemple considérer que la source Agrovoc apporte un intérêt plus important lors du calcul de la confiance d'un candidat label plutôt que d'un candidat d'une autre relation. De cette manière, nous pourrions exploiter les spécificités des sources. Ensuite, nous avons pu observer que notre algorithme de génération d'extensions n'est pas optimisé et ne permet pas de générer des extensions pour un grand nombre de données. Il serait intéressant de pouvoir améliorer notre modèle de génération d'extension pour qu'il passe à l'échelle. Nous avons aussi observé que les extensions générées contiennent peu de candidats lorsqu'il y a beaucoup d'incompatibilités. Ceci s'explique par notre hypothèse forte de ne pas considérer les correspondances complexes. Il serait alors intéressant d'étudier comment désambiguïser ces incompatibilités pour ne pas rejeter un candidat mais considérer un sous-ensemble de ses éléments ontologiques.

# Conclusion

## Conclusion et Perspectives

Le Web de données liées permet, grâce à l'utilisation des technologies du Web sémantique, de décloisonner les données disponibles sur Internet. Ces données sont alors représentées dans un format standardisé et directement interprétables par une machine. De plus, des liens de correspondance entre les vocabulaires utilisés dans les jeux de données permettent l'intéropérabilité de ces données. Ce format de représentation des données ouvertes suscite aujourd'hui un engouement certain des producteurs de jeux de données. Le nuage des données liées grossit d'année en année.

D'un autre côté, un grand nombre de sources de données présentes sur le Web ne sont pas disponibles sur ce Web de données liées. L'évolution fulgurante qu'a connue le Web depuis sa création a apporté de nombreuses applications et donc de nombreux jeux de données. Un travail important a d'ores et déjà été fait pour modéliser, stocker et rendre accessible un certain nombre de ces jeux de données, sans que ceux-ci ne soient pour autant accessibles sur le Web de données liées. Il devient particulièrement intéressant de pouvoir réutiliser ces sources existantes pour les rendre disponibles sur le Web de données liées. Une des problématique est le niveau d'expressivité nécessaire pour la représentation des données en utilisant les technologies du Web sémantique. Ce niveau d'expressivité permet une représentation explicite des concepts manipulés que nous appelons connaissances. Le passage à un niveau d'expressivité supérieure implique nécessairement de faire certaines hypothèses de désambiguïsation.

Ce phénomène d'apparition de nombreuses sources non-ontologiques présentes sur le Web est particulièrement vrai dans le domaine de l'agriculture. Comme nous l'avons vu dans le chapitre contexte, ce domaine génère de nombreux jeux de données, mais peu d'entre-eux sont présents sur le Web de données liées. La préoccupation de l'ouverture des données et leur interopérabilité est devenue décisive depuis le Grenelle de l'environnement, qui a préconisé une transition vers une agriculture durable. Celle-ci favorise l'utilisation de produits phytosanitaires uniquement lorsque cela est nécessaire, et non plus par anticipation. Pour cela, il est nécessaire de pouvoir analyser les données présentes sur le Web pour pouvoir définir le niveau de risques d'une attaque de bio-agresseur sur une culture. Ce niveau de risque indique s'il est alors nécessaire pour l'agriculteur d'agir ou non. Les différentes possibilités d'action sont soit de traiter la culture avec des produits phytosanitaires si cela est nécessaire, soit d'utiliser une technique alternative n'utilisant pas de produits. Pour cela, il est encore une fois nécessaire de pouvoir analyser différents jeux de données pour déterminer quelle est la nature de l'attaque, les risques liés à une utilisation d'un produit phytosanitaire et les différentes techniques alternatives existantes. L'interopérabilité des données est donc au cœur des préoccupations de cette transition des pratiques agricoles. À l'heure actuelle, cette interopérabilité est encore complexe et les jeux de données encore trop cloisonnés.

C'est dans ce contexte que nous avons mené notre étude. Nous transformons simul-

tanément différentes sources de données non-ontologiques afin d'obtenir une base de connaissances finale. Cette base est le résultat de la fusion des différentes transformations des sources. L'originalité de notre approche réside dans le fait que nous procédons à une transformation simultanée des sources afin de pouvoir détecter, parmi elles, des éléments de consensus, alors que les approches de la littérature utilisent séquentiellement les sources pour enrichir la base.

Un processus de transformation ne peut être efficace que s'il est orienté vers un objectif défini. Pour cela, nous organisons notre méthodologie autour de la partie haute d'un module ontologique. Ce module permet tout d'abord de définir les bornes du domaine étudié. Il permet aussi d'harmoniser la modélisation des éléments, lors d'une première étape de transformation. Cette transformation est faite sur toutes les sources considérées d'après un module défini.

Nous cherchons ensuite à fusionner les sources transformées pour créer des candidats d'éléments pouvant apparaître dans la base de connaissances finale. Pour cela, nous utilisons un système d'alignement qui nous permet de détecter les correspondances existant entre ces sources transformées. À partir de ces correspondances, nous générons des candidats d'éléments ontologiques. Ces candidats représentent les classes ou individus potentiels pouvant apparaître dans la base de connaissances finale. Nous générons ensuite des candidats arcs représentant les relations potentielles apparaissant entre ces candidats d'éléments ontologiques.

Pour chaque candidat est défini un score de confiance dans le calcul duquel nous considérons plusieurs facteurs :

— Le premier est le consensus existant entre les différentes sources sur ce candidat. Nous considérons que plus un élément apparaît dans plusieurs sources, plus il est digne de confiance. Cet intérêt est calculé en fonction de la définition de la qualité de chacune des sources.

— Le second aspect considéré dans le calcul de la confiance est la connexité des éléments fusionnés. Plus il y a de correspondances utilisées pour générer le candidat, plus la confiance sera élevée.

Nous avons proposé trois fonctions de calcul de la confiance. La première fonction appelée $trust_{likelihood}$ permet de représenter le consensus entre les sources. La deuxième fonction appelée $trust_{degree}$ permet de représenter la connexité des éléments fusionnés. Nous avons enfin proposé $trust_{choquet}$ qui prend en compte ces deux aspects.

À l'heure actuelle, les systèmes d'alignement ne permettent d'obtenir que des correspondances de type équivalence. Néanmoins, il apparaît qu'un élément d'une source peut être équivalent à plusieurs éléments d'une deuxième source. Ce type de correspondances fait partie des correspondances dites complexes. Ces correspondances nécessitent une désambiguïsation afin de déterminer la nature exacte de la relation liant ces éléments. Nous faisons l'hypothèse de considérer ces correspondances multiples comme un ensemble d'erreurs provenant du système d'alignement. Cette hypothèse nous amène à établir des incompatibilités entre les candidats qui partagent un même élément, puisqu'ils ont été générés à partir de ce type de correspondances. En considérant ces incompatibilités, nous générons une extension qui est le sous-ensemble de candidats compatibles maximisant la

somme des confiances. La génération de cette extension permet de faciliter le travail de validation nécessaire pour obtenir la base de connaissances finale.

Un outil a été développé afin de mettre en œuvre la méthodologie présentée dans ce manuscrit : Muskca. Cet outil génère des candidats à partir de différentes sources issues de la première transformation. Il génère également l'extension maximisant la confiance de ces candidats. Muskca exporte ensuite ce résultat en maintenant la provenance des candidats pour pouvoir justifier la confiance attribuée.

Nous avons expérimenté notre approche grâce à l'outil Muskca sur deux jeux de données différents. Le premier, concernant la taxonomie des blés, nous a permis de justifier l'utilisation d'une fonction de confiance d'après le consensus des éléments fusionnés. Pour cela, nous avons considéré trois sources sur ce domaine, choisies avec l'aide d'experts. Trois experts ont ensuite validé les éléments des sources qu'ils souhaitaient retrouver dans la base de connaissances finale. D'après ces validations, nous avons pu comparer les résultats obtenus en utilisant trois fonctions de confiance différentes. Nous avons aussi montré l'intérêt de la génération d'une extension pour le processus de validation avec ce jeu de données d'évaluation. Le deuxième jeu de données sur lequel nous avons évalué notre approche provient d'une campagne d'évaluation consacrée aux systèmes d'alignement : OAEI. Nous avons pu montrer la généricité de notre approche en l'appliquant sur un grand jeu de données déjà sous forme de bases de connaissances. Pour cela, nous avons considéré sept sources concernant les données de conférences. Ces sources ne traitant pas des mêmes conférences, elles ne contiennent que peu d'éléments en commun. Nous avons pu montrer que l'utilisation de Muskca n'altère pas la qualité de la base de connaissances finale, même lorsqu'il n'y a pas beaucoup d'éléments en commun.

Plusieurs pistes de perspectives sont envisagées pour faire évoluer nos travaux. Nous souhaitons tout d'abord expérimenter notre approche sur d'autres jeux de données, afin d'expérimenter l'intérêt d'un consensus entre de nombreuses sources. L'expérimentation sur la taxonomie des blés portait seulement sur trois sources et celle de l'OAEI ne comportait que peu d'éléments réellement consensuels. Nous envisageons quatre évolutions possibles sur notre approche :

**Propagation de la confiance des candidats :**
Nous avons montré que l'utilisation de la propagation de la confiance des candidats sommets sur les candidats arcs associés permet d'améliorer les résultats. La propagation que nous avons défini avec la fonction $trust_{degree_r}$ ne permet que de considérer la propagation de la confiance des candidats sommets sur les candidats arcs. Alors que cette propagation peut être considérée entre deux candidats sommets, mais aussi entre candidats arcs et candidats sommets. De la même façon que la probabilité est propagée dans un réseau Bayésien, il serait intéressant de pouvoir définir le même phénomène ici.

**Nouveaux types de candidats :**
Nous avons considéré ici uniquement des candidats sommets de type classes ou individus et labels et des candidats arcs. Ce choix a été fait dans un soucis de simplification de l'étude, mais également pour nous adapter aux besoins réels de notre domaine d'application. Néanmoins, il est tout à fait possible de considérer

l'ajout de nouvelles propriétés d'objets ou de nouvelles propriétés de types de données. L'algorithme de création de candidats arcs serait alors modifié puisqu'il faudrait prendre en considération les candidats sommets propriétés comme étiquettes potentielles des candidats arcs. En plus des candidats sommets et des candidats arcs, il serait également intéressant de considérer des candidats de type règles. Ces candidats permettraient de spécifier la définition formelle d'une classe ou d'une propriété suivant les contraintes associées. Ce type de candidat serait très difficile à identifier dans des sources structurées telles que des thésaurus ou des taxonomies qui n'ont pas une expressivité suffisante. Bien que certaines contraintes puissent être extraites des bases de données, comme nous l'avons vu dans l'état de l'art, elles restent très spécifiques aux bases de données et difficilement transposables pour la définition formelle. Les sources qui seraient particulièrement adaptées à l'extraction de ces candidats règles seraient les textes en langage naturel. Prenons par exemple la base de connaissances finale pour l'annotation des BSV. Nous pouvons envisager la création d'une première version de la base de connaissances grâce à la transformation des sources non-ontologiques structurées. Cette base pourrait être enrichie en utilisant des candidats règles extraits à partir des annotations des BSV.

**Extension cohérente :**

Les incompatibilités utilisées pour générer les extensions ne prennent pas en compte la cohérence logique des candidats dans l'extension. Il serait intéressant de pouvoir déterminer les incompatibilités logiques entre candidats d'après les contraintes définies dans le module. De cette manière, une extension ne prendrait pas uniquement en considération les correspondances erronées comme nous l'avons présenté dans ce manuscrit, mais également la cohérence logique de la base de connaissances finale. Il serait alors nécessaire de pouvoir déterminer les sous-ensembles maximaux de candidats qui n'impliquent pas d'incohérences logiques. Les incompatibilités logiques sont plus complexes à déterminer que les incompatibilités que nous avons présentées dans notre étude. En effet, les incompatibilités présentées dans notre étude ne considèrent les candidats incompatibles que deux à deux, alors qu'une incompatibilité logique intervient entre deux sous-ensembles.

**Traitement des alignements complexes :**

Nous avons posé l'hypothèse de ne pas considérer les alignements complexes. Cette hypothèse est particulièrement forte puisqu'elle implique des incompatibilités entre candidats, ce qui réduit les possibilités de désambiguïsation. En effet, les correspondances multiples impliquant les correspondances ne sont pas nécessairement des erreurs du système d'alignement. Une première perspective de traitement serait de ne pas supprimer totalement le candidat mais de pouvoir enlever l'élément ontologique créant l'incompatibilité avec un autre candidat de l'extension. Il serait également intéressant de pouvoir déterminer la nature des correspondances, ce qui nous permettrait d'ajuster les relations de dépendances entre candidats. En effet, si la correspondance entre un élément d'une source et plusieurs éléments d'une deuxième source représente une relation de méronymie, alors nous pouvons ajouter cette relation à notre base de connaissances finale. Il est nécessaire de pouvoir

déterminer le candidat représentant réellement le concept principal et les candidats représentant les composants de ce premier candidat.

**Amélioration de la définition de l'implication des sources :**
Lors du calcul de la confiance d'un candidat, nous avons considéré la notion d'intérêt de l'implication des sources. Pour cela, un score de qualité des sources a été défini. Ce score de qualité représente la confiance que l'on porte aux éléments présents dans cette source. Cette notion peut être spécialisée puisque certaines sources impliquent un intérêt variable suivant le type d'élément considéré. Le thésaurus Agrovoc présente par exemple un intérêt majeur pour les candidats de type labels. Il est intéressant de pouvoir spécialiser la qualité d'une source en fonction du type de candidats. De plus, la fonction d'intérêt de l'implication des sources utilisée dans le calcul de la confiance des candidats considère les sources comme indépendantes, c'est à dire que l'apport d'intérêt d'une source dans cette fonction sera le même, quelles que soient les autres sources considérées. Il serait néanmoins particulièrement intéressant de pouvoir préciser l'intérêt d'une source en fonction de son implication. Nous aurions par exemple pu vouloir modéliser que l'intérêt de la source Agrovoc est moins important si la source TaxRef est déjà impliquée. Ce phénomène représente la dépendance entre les sources.

Dans ce manuscrit, nous avons présenté notre proposition pour établir la base d'une méthodologie pour la transformation de plusieurs sources non-ontologiques. Cette méthodologie reste perfectible, notamment en travaillant sur les perspectives que nous venons d'évoquer. Cette méthodologie trouve pleinement sa place dans le Web de données liées et plus généralement le Web sémantique en intégrant une introduction à la confiance en fonction de la provenance des ressources d'une base de connaissances. Avec l'évolution certaine du nombre de données sur le Web de données liées, cette problématique de la confiance sur les données va devenir centrale dans les prochaines années.

# Sixième partie .
# Annexes

## Table des figures

## Liste des tableaux

# Table des définitions

# Septième partie .
# Bibliographie

## Références


[25964-1, 2011] 25964-1, I. (2011). Information and documentation – Thesauri and interoperability with other vocabularies – Part 1 : Thesauri for information retrieval. *Thesauri and interoperability with other vocabularies. Part*, 1. bibtex : 25964-1_information_2011.

[Abbas and Berio, 2013] Abbas, M. and Berio, G. (2013). Creating Ontologies Using Ontology Mappings : Compatible and Incompatible Ontology Mappings. *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, pages 143–146. bibtex : abbas_creating_2013.

[Agrawal et al., 1989] Agrawal, R., Borgida, A., and Jagadish, H. V. (1989). *Efficient management of transitive relationships in large data and knowledge bases*, volume 18. ACM. bibtex : agrawal_efficient_1989.

[Alexander, 1979] Alexander, C. (1979). *The timeless way of building*, volume 1. New York : Oxford University Press. bibtex : alexander_timeless_1979.

[APG, 2009] APG (2009). An update of the Angiosperm Phylogeny Group classification for the orders and families of flowering plants : APG III : APG III. *Botanical Journal of the Linnean Society*, 161(2) :105–121. bibtex : apg_update_2009.

[Arenas et al., 2012] Arenas, M., Bertails, A., Prud, E., and Sequeda, J. (2012). A Direct Mapping of Relational Data to RDF. bibtex : arenas_direct_2012.

[Artz and Gil, 2007] Artz, D. and Gil, Y. (2007). A survey of trust in computer science and the semantic web. *Web Semantics : Science, Services and Agents on the World Wide Web*, 5(2) :58–71. bibtex : artz_survey_2007.

[Astrova, 2004] Astrova, I. (2004). Reverse engineering of relational databases to ontologies. In *The Semantic Web : Research and Applications*, pages 327–341. Springer. bibtex : astrova_reverse_2004.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). *Dbpedia : A nucleus for a web of open data*. Springer. bibtex : auer_dbpedia_2007.

[Auer et al., 2009] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., and Aumueller, D. (2009). Triplify : light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 621–630, New York, NY, USA. ACM. bibtex : auer_triplify_2009.

[Avraham et al., 2008] Avraham, S., Tung, C.-W., Ilic, K., Jaiswal, P., Kellogg, E. A., McCouch, S., Pujar, A., Reiser, L., Rhee, S. Y., Sachs, M. M., and others (2008). The Plant Ontology Database : a community resource for plant structure and developmental

stages controlled vocabulary and annotations. *Nucleic acids research*, 36(suppl 1) :D449–D454. bibtex : avraham_plant_2008.

[Berners-Lee, 1989] Berners-Lee, T. (1989). Information management : A proposal. bibtex : berners-lee_information_1989.

[Berners-Lee and Connolly, 1995] Berners-Lee, T. and Connolly, D. (1995). *Hypertext markup language-2.0*. RFC 1866, November. bibtex : berners-lee_hypertext_1995.

[Berners-Lee et al., 2000] Berners-Lee, T., Fischetti, M., and Foreword By-Dertouzos, M. L. (2000). *Weaving the Web : The original design and ultimate destiny of the World Wide Web by its inventor*. HarperInformation. bibtex : berners-lee_weaving_2000.

[Borst, 1997] Borst, W. N. (1997). *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente. bibtex : borst1997construction.

[Brickley and Miller, 2005] Brickley, D. and Miller, L. (2005). Foaf vocabulary specification. *RDF Web Namespace Document*. bibtex : brickley2005foaf.

[Bron and Kerbosch, 1973] Bron, C. and Kerbosch, J. (1973). Algorithm 457 : finding all cliques of an undirected graph. *Communications of the ACM*, pages 575–577. bibtex : bron_algorithm_1973.

[Caracciolo et al., 2013] Caracciolo, C., Stellato, A., Morshed, A., Johannsen, G., Rajbahndari, S., Jaques, Y., and Keizer, J. (2013). The agrovoc linked dataset. *Semantic Web*, 4(3) :341–348. bibtex : caracciolo_agrovoc_2013.

[Cerbah, 2008] Cerbah, F. (2008). Learning highly structured semantic repositories from relational databases. In *The Semantic Web : Research and Applications*, pages 777–781. Springer. bibtex : cerbah_learning_2008.

[Charlet et al., 2012] Charlet, J., Declerck, G., Dhombres, F., Gayet, P., Miroux, P., Vandenbussche, P., et al. (2012). Construire une ontologie médicale pour la recherche d'information : problématiques terminologiques et de modélisation. *Actes des 23es journées francophones d'Ingénierie des connaissances*, 1 :33textendash48. bibtex : charlet_construire_2012.

[Chrisment et al., 2008] Chrisment, C., Haemmerlé, O., Hernandez, N., and Mothe, J. (2008). Méthodologie de transformation d'un thesaurus en une ontologie de domaine. *Revue d'Intelligence Artificielle*, 22(1) :7–37. bibtex : chrisment_methodologie_2008.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6) :377–387. bibtex : codd_relational_1970.

[Cronquist, 1981] Cronquist, A. (1981). *An Integrated System of Classification of Flowering Plants*. Columbia University Press. bibtex : cronquist_integrated_1981.

[d'Aquin et al., 2006] d'Aquin, M., Sabou, M., and Motta, E. (2006). Modularization : a key for the dynamic selection of relevant knowledge components. bibtex : d2006modularization.

[Das et al., 2012] Das, S., Sundara, S., and Cyganiak, R. (2012). R2rml : RDB to RDF Mapping Language. bibtex : das_r2rml_2012.

[de Candolle, 1813] de Candolle, A.-P. (1813). *Théorie élémentaire de la botanique*. Deterviile. bibtex : decandolle_theorie_1813.

[Dhombres et al., 2012] Dhombres, F., Vandenbussche, P.-Y., Rath, A., Hanaeur, M., Olry, A., Urbero, B., Choquet, R., and Charlet, J. (2012). Projet OrphaOnto-Première étape de l'ontologisation des bases de connaissances d'Orphanet. *Actes de IC2011*, pages 573–588. bibtex : dhombres_projet_2012.

[Dong et al., 2013] Dong, X. L., Berti-Equille, L., and Srivastava, D. (2013). Data fusion : resolving conflicts from multiple sources. In *Handbook of Data Quality*, pages 293–318. Springer. bibtex : dong_data_2013.

[Dragisic et al., 2014] Dragisic, Z., Eckert, K., Euzenat, J., Faria, D., Ferrara, A., Granada, R., Ivanova, V., Jiménez-Ruiz, E., Kempf, A. O., Lambrix, P., et al. (2014). Results of the ontology alignment evaluation initiative 2014. In *Proceedings of the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference (ISWC 2014)*. bibtex : dragisic_results_2014-1.

[Dubois and Prade, 1985] Dubois, D. and Prade, H. (1985). Théorie des possibilités. bibtex : dubois1985theorie.

[Dubois et al., 1988] Dubois, D., Prade, H., Farreny, H., Martin-Clouaire, R., and Testemale, C. (1988). *Théorie des possibilités : applications à la représentation des connaissances en informatique*, volume 1. Masson Paris. bibtex : dubois_theorie_1988.

[Eppstein and Strash, 2011] Eppstein, D. and Strash, D. (2011). Listing all maximal cliques in large sparse real-world graphs. *Experimental Algorithms*, pages 364–375. bibtex : eppstein_listing_2011.

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*. bibtex : euzenat_ontology_2007.

[Fleiss and Cohen, 1973] Fleiss, J. L. and Cohen, J. (1973). The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*. bibtex : fleiss1973equivalence.

[Gangemi, 2005] Gangemi, A. (2005). Ontology design patterns for semantic web content. In *The Semantic Web–ISWC 2005*, pages 262–276. Springer. bibtex : gangemi_ontology_2005.

[Gangemi et al., 2003] Gangemi, A., Navigli, R., and Velardi, P. (2003). The OntoWordNet Project : extension and axiomatization of conceptual relations in WordNet. *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*, 1 :820–838. bibtex : gangemi_ontowordnet_2003.

[Gangemi and Presutti, 2009] Gangemi, A. and Presutti, V. (2009). Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer. bibtex : gangemi_ontology_2009.

[Gargominy et al., 2014] Gargominy, O., Tercerie, S., Régnier, C., Ramage, T., Schoelinck, C., Dupont, P., Vandel, E., Daszkiewicz, P., and Poncet, L. (2014). TAXREF v8. 0, référentiel taxonomique pour la France : Méthodologie, mise en oeuvre et diffusion. *Rapport SPN*, 42 :2014. bibtex : gargominy_taxref_2014.

[Gil and Artz, 2007] Gil, Y. and Artz, D. (2007). Towards content trust of web resources. *Web Semantics : Science, Services and Agents on the World Wide Web*, 5(4) :227–239. bibtex : gil_content_2007.

[Gillet et al., 2013] Gillet, P., Trojahn, C., Haemmerlé, O., and Pradel, C. (2013). Complex correspondences for query patterns rewriting.

[Grau et al., 2007] Grau, B. C., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). Just the right amount : extracting modules from ontologies. In *Proceedings of the 16th international conference on World Wide Web*, pages 717–726. ACM. bibtex : grau_just_2007.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2) :199–220. bibtex : gruber1993translation.

[Hahn, 2003] Hahn, U. (2003). Turning informal thesauri into formal ontologies : a feasibility study on biomedical knowledge re-use. *Comparative and functional genomics*, 4(1) :94textendash97. bibtex : hahn_turning_2003.

[Hassine-Guetari, 2014] Hassine-Guetari, S. B. (2014). *Ecole Doctorale Informatique et Mathématiques*. PhD thesis, Université Rennes 1. bibtex : hassine-guetari_ecole_2014.

[Hepp and De Bruijn, 2007] Hepp, M. and De Bruijn, J. (2007). GenTax : A generic methodology for deriving OWL and RDF-S ontologies from hierarchical classifications, thesauri, and inconsistent taxonomies. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4519 LNCS, pages 129–144, Innsbruck. cited By (since 1996) 5 ; Conference of 4th European Semantic Web Conference, ESWC 2007 ; Conference Date : 3 June 2007 through 7 June 2007 ; Conference Code : 70113 bibtex : hepp_gentax_2007.

[ISO 2788, 1986] ISO 2788 (1986). Documentation – Guidelines for the establishment and development of monolingual thesauri. *Documentation – Principes directeurs pour l'établissement et le développement de thésaurus monolingues*, 1. bibtex : iso2788_documentation_1986.

[ISO 8879, 1986] ISO 8879 (1986). ISO 8879 : 1986, Information processing-Text and office systems-Standard Generalized Markup Language (SGML). *ISO, Geneva*. bibtex : iso8879_iso_1986.

[Jayawardene et al., 2013] Jayawardene, V., Sadiq, S., and Indulska, M. (2013). An analysis of data quality dimensions. Technical report. bibtex : jayawardene_analysis_2013.

[Jiménez-Ruiz and Grau, 2011] Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap : Logic-based and scalable ontology matching. *The Semantic Web–ISWC 2011*, pages 273–288. bibtex : jimenez-ruiz_logmap_2011.

[Jonquet et al., 2015a] Jonquet, C., Dzalé-Yeumo, E., Arnaud, E., and Larmande, P. (2015a). AgroPortal : a proposition for ontology-based services in the agronomic domain. In *IN-OVIVE*, Rennes. bibtex : jonquet_agroportal_2015.

[Jonquet et al., 2015b] Jonquet, C., Emonet, V., and Musen, M. A. (2015b). Roadmap for a multilingual BioPortal. bibtex : jonquet_roadmap_2015.

[Khaleghi et al., 2013] Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion : A review of the state-of-the-art. *Information Fusion*, 14(1) :28–44. bibtex : khaleghi_multisensor_2013.

[Kless et al., 2012] Kless, D., Jansen, L., Lindenthal, J., and Wiebensohn, J. (2012). A method for re-engineering a thesaurus into an ontology. page 133. Ios PressInc. bibtex : kless_method_2012.

[Krivine et al., 2009] Krivine, S., Nobécourt, J., Soualmia, L., Cerbah, F., and Duclos, C. (2009). Construction automatique d'ontologie à partir de bases de données relationnelles : application au médicament dans le domaine de la pharmacovigilance. *IC 2009*. bibtex : krivine_construction_2009.

[Labreuche et al., 2008] Labreuche, C., Grabisch, M., and Mayag, B. (2008). Un algorithme de détermination de la capacité pour l'intégrale de Choquet 2-additive. bibtex : labreuche_algorithme_2008.

[Lebo et al., 2013] Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2013). Prov-o : The prov ontology. *W3C Recommendation*, 30. bibtex : lebo_provo_2013.

[Li et al., 2009] Li, J., Tang, J., Li, Y., and Luo, Q. (2009). Rimom : A dynamic multistrategy ontology alignment framework. *Knowledge and Data Engineering, IEEE Transactions on*, 21(8) :1218–1232. bibtex : li_rimom_2009.

[Li and Li, 2012] Li, P. and Li, Y. (2012). On Transformation from The Thesaurus into Domain Ontology. 1. bibtex : li_transformation_2012.

[Masolo et al., 2002] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Oltramari, R., Schneider, L., Istc-cnr, L. P., and Horrocks, I. (2002). Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology. bibtex : masolo_wonderweb_2002.

[McCandless et al., 2010] McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). *Lucene in Action : Covers Apache Lucene 3.0*. Manning Publications Co. bibtex : mccandless_lucene_2010.

[Miles and Bechhofer, 2009] Miles, A. and Bechhofer, S. (2009). SKOS simple knowledge organization system reference. *W3C recommendation*, 18 :W3C. bibtex : miles_skos_2009.

[Murofushi and Sugeno, 1991] Murofushi, T. and Sugeno, M. (1991). A theory of fuzzy measures : representations, the Choquet integral, and null sets. *Journal of Mathematical Analysis and Applications*, 159(2) :532–549. bibtex : murofushi_theory_1991.

[Noy and Musen, 2003] Noy, N. F. and Musen, M. A. (2003). The PROMPT suite : interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6) :983–1024.

[Noy et al., 2009] Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and others (2009). BioPortal : ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, page gkp440. bibtex : noy_bioportal_2009.

[Otero-Cerdeira et al., 2015] Otero-Cerdeira, L., Rodriguez-Martinez, F. J., and Gomez-Rodriguez, A. (2015). Ontology matching : A literature review. *Expert Systems with Applications*, 42(2) :949–971. bibtex : otero2015ontology.

[Pottinger and Bernstein, 2003] Pottinger, R. A. and Bernstein, P. A. (2003). Merging models based on given correspondences. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 862–873. VLDB Endowment.

[Poveda-Villalon et al., 2012] Poveda-Villalon, M., Suárez-Figueroa, M. C., and Gomez-Pérez, A. (2012). Validating ontologies with OOPS ! In *Knowledge Engineering and Knowledge Management*, pages 267–281. Springer. bibtex : poveda-villalon_validating_2012.

[Poveda Villalon et al., 2010] Poveda Villalon, M., Suárez-Figueroa, M. C., and Gómez-Pérez, A. (2010). Reusing ontology design patterns in a context ontology network. bibtex : povedavillalon_reusing_2010.

[Pradel, 2013] Pradel, C. (2013). *D'un langage de haut niveau à des requ^etes graphes permettant d'interroger le web sémantique*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier. bibtex : pradel_dun_2013.

[Raunich and Rahm, 2014] Raunich, S. and Rahm, E. (2014). Target-driven merging of taxonomies with Atom. *Information Systems*, 42 :1–14.

[Rector, 2003] Rector, A. L. (2003). Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 121–128. ACM. bibtex : rector_modularisation_2003.

[Roussey, 2001] Roussey, C. (2001). *Une Méthode d'indexation sémantique adaptée aux corpus multilingues*. Thèse doctorat, Institut national des sciences appliquées, Lyon, France. bibtex : roussey_methode_2001.

[Roussey and Bernard, 2015] Roussey, C. and Bernard, S. (2015). Annotation des Bulletins de Santé du Végétal. In *7ème Atelier Recherche d'Information SEmantique*, pages 12–p. bibtex : roussey2015annotation.

[Roussey et al., 2013] Roussey, C., Chanet, J.-P., Cellier, V., and Amarger, F. (2013). Agronomic taxon. In *Proceedings of the 2nd International Workshop on Open Data*, page 5. ACM. bibtex : roussey_agronomic_2013.

[Sayers, 2009] Sayers, E. (2009). Entrez programming utilities help. *URL http ://www. ncbi. nlm. nih. gov/books/NBK25499*. bibtex : sayers_entrez_2009.

[Schmeidler, 1986] Schmeidler, D. (1986). Integral representation without additivity. *Proceedings of the American mathematical society*, 97(2) :255–261. bibtex : schmeidler_integral_1986.

[Sequeda et al., 2011a] Sequeda, J. F., Arenas, M., and Miranker, D. P. (2011a). A Completely Automatic Direct Mapping of Relational Databases to RDF and OWL. *ISWC*. bibtex : sequeda_completely_2011.

[Sequeda et al., 2012] Sequeda, J. F., Arenas, M., and Miranker, D. P. (2012). On directly mapping relational databases to rdf and owl (extended version). *arXiv preprint arXiv :1202.3667*. bibtex : sequeda_directly_2012.

[Sequeda et al., 2011b] Sequeda, J. F., Tirmizi, S. H., Corcho, O., and Miranker, D. P. (2011b). Survey of directly mapping SQL databases to the Semantic Web. *The Knowledge Engineering Review*, 26 :445–486. bibtex : sequeda_survey_2011.

[Shafranovich, 2005] Shafranovich, Y. (2005). Common format and MIME type for Comma-Separated Values (CSV) files. bibtex : shafranovich_common_2005.

[Shrestha et al., 2010] Shrestha, R., Arnaud, E., Mauleon, R., Senger, M., Davenport, G. F., Hancock, D., Morrison, N., Bruskiewich, R., and McLaren, G. (2010). Multifunctional crop trait ontology for breeders' data : field book, annotation, data discovery and semantic enrichment of the literature. *AoB plants*, 2010 :plq008. bibtex : shrestha_multifunctional_2010.

[Shvaiko and Euzenat, 2013] Shvaiko, P. and Euzenat, J. (2013). Ontology matching : state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1) :158–176. bibtex : shvaiko_ontology_2013.

[Soergel et al., 2004] Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., and Katz, S. (2004). Reengineering thesauri for new applications : The AGROVOC example. *Journal of Digital Information*, 4(4). cited By (since 1996) 51 bibtex : soergel_reengineering_2004.

[Strong et al., 1997] Strong, D. M., Lee, Y. W., and Wang, R. Y. (1997). Data quality in context. *Communications of the ACM*, 40(5) :103–110. bibtex : strong_data_1997.

[Studer et al., 1998] Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering : principles and methods. *Data & knowledge engineering*, 25(1) :161–197. bibtex : studer1998knowledge.

[Suominen and Hyvönen, 2012] Suominen, O. and Hyvönen, E. (2012). Improving the quality of SKOS vocabularies with Skosify. In *Knowledge Engineering and Knowledge Management*, pages 383–397. Springer. bibtex : suominen_improving_2012.

[Suárez-Figueroa, 2010] Suárez-Figueroa, M. C. (2010). *NeOn Methodology for building ontology networks : specification, scheduling and reuse*. PhD thesis, Informatica.

[Suárez-Figueroa et al., 2012] Suárez-Figueroa, M. C., Gómez-Pérez, A., Motta, E., and Gangemi, A. (2012). *Ontology engineering in a networked world*. Springer Science & Business Media. bibtex : suarez-figueroa_ontology_2012.

[Svatek, 2004] Svatek, V. (2004). Design patterns for semantic web ontologies : Motivation and discussion. In *Proceedings of the 7th Conference on Business Information Systems, Poznan*. bibtex : svatek_design_2004.

[Tirmizi et al., 2008] Tirmizi, S. H., Sequeda, J., and Miranker, D. (2008). Translating sql applications to the semantic web. In *Database and Expert Systems Applications*, pages 450–464. bibtex : tirmizi_translating_2008.

[Tomita et al., 2006] Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, pages 28–42. bibtex : tomita_worstcase_2006.

[Trojahn et al., 2011] Trojahn, C., Euzenat, J., Tamma, V., and Payne, T. R. (2011). Argumentation for reconciling agent ontologies. *Semantic Agent Systems*, pages 89–111. bibtex : trojahn_argumentation_2011.

[Turenne et al., 2015] Turenne, N., Andro, M., Corbière, R., and Phan, T. T. (2015). Open Data Platform for Knowledge Access in Plant Health Domain : VESPA Mining. *arXiv preprint arXiv :1504.06077*. bibtex : turenne2015open.

[Van Assem et al., 2004] Van Assem, M., Menken, M., Schreiber, G., Wielemaker, J., and Wielinga, B. (2004). A method for converting thesauri to RDF/OWL. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3298 :p17–31. cited By (since 1996) 10 bibtex : vanassem_method_2004.

[Villazon-Terrazas et al., 2010] Villazon-Terrazas, B., Carmen Suarez-Figueroa, M., and Gomez-Perez, A. (2010). A Pattern-Based Method for Re-Engineering Non-Ontological Resources into Ontologies. *International Journal on Semantic Web and Information Systems*, 6(4) :27–63. WOS :000286635000002 bibtex : villazon-terrazas_patternbased_2010.

[Wielinga et al., 2001] Wielinga, B. J., Schreiber, A. T., Wielemaker, J., and Sandberg, J. A. C. (2001). From thesaurus to ontology. pages 194–201, Victoria, British Columbia, Canada. ACM. bibtex : wielinga_thesaurus_2001.

[Winkler, 1990] Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. bibtex : winkler_string_1990.

[Winkler, 1999] Winkler, W. E. (1999). The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer. bibtex : winkler_state_1999.

[Zadeh et al., 1978] Zadeh, L., Negoita, C., and Zimmermann, H. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1 :3–28. bibtex : zadeh_fuzzy_1978.

[Zins, 2007] Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge. *Journal of the American society for information science and technology*, 58(4) :479–493. bibtex : zins_conceptual_2007.