Détection de liens d'identité contextuels dans une base de connaissances

Joe Raad, Nathalie Pernelle, Fatiha Saïs

LRI, Paris Sud University, Bât. 650, F-91405 Orsay, France
firstname.lastname@lri.fr

Résumé : De nombreuses applications du Web de données exploitent des liens d'identités déclarés à l'aide du constructeur $owl:sameAs$. Cependant, différentes études ont montré qu'une utilisation abusive de ces liens peut conduire à des inférences erronées ou contradictoires. Dans ce papier nous proposons de calculer des liens d'identités contextuels qui permettent d'expliciter les contextes dans lesquels ces liens sont valides. La notion de contexte que nous proposons est représentée en se basant sur l'ontologie de domaine dans laquelle les instances sont représentées. Nous avons expérimenté cette approche dans le domaine des données scientifiques où les éléments décrivant les expériences partagent rarement un lien d'identité tel que défini par $owl:sameAs$.

Mots-clés : Liage de données, Contextes, Ontologies, Bases de connaissances, Enrichissement.

1 Introduction

Le Linked Open Data cloud est une initiative du W3C, qui définit un ensemble de bonnes pratiques pour publier et lier des données structurées sur le Web. En utilisant des technologies du Web sémantique, des applications peuvent partager, extraire, interroger ou raisonner sur les données publiées. Le Linked Open Data (LOD) a récemment pris une nouvelle dimension avec la publication de grandes quantités de données (le LOD est passé de 500 millions de triplets RDF en 2007 à 130 milliards de triplets en 2016). Ces données sont encyclopédiques telles que DBpedia, Yago ou encore Google Knowledge Vault ou concernent différents domaines d'application comme les sciences du vivant, la culture ou encore l'économie. Toutefois, si ces données se retrouvent isolées, leur utilité reste très limitée. En effet, un des points angulaires du Web des données est le fait que les données soient liées entre elles par des liens sémantiques tels que les liens d'identité *owl :sameAs* qui expriment que deux ressources différentes réfèrent à la même entité (e.g., même personne, même article, même gène).

Ce type de liens est défini dans (Patel-Schneider *et al.* (2004)) avec une sémantique très stricte qui exprime le fait que déclarer un fait *owl :sameAs* entre deux objets indique que toutes les valeurs de propriétés déclarées pour l'un peuvent aussi être déclarées pour l'autre. Ainsi, si des faits *owl :sameAs* erronés sont déclarés dans les bases de connaissances cela peut conduire à inférer des informations erronées et même contradictoires. Dans (Jaffri *et al.* (2008)), les auteurs ont évalué la qualité du résultat du liage de données obtenu entre des données DBLP et des données de DBpedia. Pour cela, ils ont mesuré la correction des nouveaux faits inférés en exploitant la sémantique logique des liens *owl*:*sameAs*. En choisissant arbitrairement 49 noms parmi les 491 796 auteurs disponibles dans DBLP 2006, ils ont montré que 92 % de ces 49 auteurs ont eu des publications affiliées incorrectement. Par ailleurs, l'absence d'autres types de liens alternatifs avec une sémantique claire, renforce l'utilisation abusive du *owl*:*sameAs*. Souvent, la relation d'identité entre objets est plus faible et dépend du contexte dans lequel on

veut utiliser l'identité. Prenons l'exemple de deux éditions distinctes du même livre. Dans le cas où l'on s'intéresse à identifier s'il s'agit de la même édition de livre alors ces deux livres seront différents. Cependant, dans un contexte où l'on cherche à savoir s'il s'agit de la même oeuvre artistique alors ces deux livres seront identiques.

Dans cet article, nous proposons une approche de détection de *liens d'identité contextuels* dans des bases de connaissances RDF [1]. Un lien d'identité contextuel exprime une relation d'identité entre deux instances, valide dans un contexte défini par rapport à une ontologie de domaine. Pour cela, nous avons défini une notion de contexte global composé d'un ensemble de contextes locaux. Nous avons développé un algorithme de détection de liens d'identité contextuels qui permet de déterminer pour chaque couple d'instances les contextes globaux les plus spécifiques dans lesquels ces instances peuvent être considérées comme étant identiques. Nous avons testé notre approche sur des données scientifiques issues d'un projet dans le domaine de l'agro-alimentaire.

Dans ce qui suit, nous présentons en section 2 un ensemble de travaux connexes. Puis, en section 3, nous fournissons les objectifs ainsi que les définitions utilisées dans notre approche. En section 4 nous présentons l'algorithme DECIDE qui calcule les liens d'identité contextuels. Enfin, nous présentons les premiers résultats d'expérimentation en section 5.

2 Travaux connexes

Les approches de découverte de liens d'identité permettent de détecter que deux descriptions se réfèrent au même objet du monde réel (e.g. même personne, même lieu, même gène). Avec l'initiative du Linked Open Data cloud (LOD) proposée par Tim-Berners Lee en 2007, un fort engouement a été constaté pour le développement d'approches de liage de données RDF, dans le domaine du Web sémantique (voir Ferrara *et al.* (2011) pour un état de l'art). Pour représenter les liens d'identités générés par des approches (semi)-automatiques, le constructeur $owl{:}sameAs$ défini dans le langage OWL2 (Patel-Schneider *et al.* (2004)) est utilisé, mais sa sémantique stricte exige que, si deux objets sont liés par un lien *owl*:*sameAs*, ces derniers doivent avoir les mêmes valeurs pour toutes les propriétés ($(owl{:}sameAs(i_1, i_2) \wedge p(i_1, v) \Rightarrow p(i_2, v))$). Certaines approches se sont focalisé sur la détection de liens erronés. Dans (de Melo (2013)), les auteurs ont exploité l'hypothèse du nom unique pour détecter des liens d'identité erronés. Dans le même esprit, le travail de Ding *et al.* (2010) propose une approche globale qui par analyse de la topologie du graphe des liens d'identité, détecte des liens d'identité invalides. Enfin, l'approche logique proposée dans (Papaleo *et al.* (2014)) permet de détecter des liens d'identité invalides en exploitant un sous-graphe RDF construit en exploitant des axiomes de fonctionnalité et de complétude locale des propriétés.

D'autres approches ont caractérisé les différentes situations où le lien *owl*:*sameAs* serait utilisé à mauvais escient (Halpin *et al.* (2010); de Melo (2013); Ding *et al.* (2010)). Halpin et al. (Halpin *et al.* (2010)) ont fait état de quatre cas où le lien *owl*:*sameAs* serait utilisé de façon inappropriée. Les auteurs citent en particulier le cas où un lien d'identité est déclaré entre deux objets dont les informations décrivent une entité dans deux contextes différents. En d'autres termes, il ne s'agit pas des mêmes informations qui sont renseignées dans tous les contextes (e.g. contexte social vs professionnel). Dans ce type de cas, il est nécessaire de pouvoir distinguer et repré-

1. https ://www.w3.org/RDF/

senter les contextes dans lesquels un lien d'identité serait valide. Il existe quelques propositions pour la représentation de liens d'identité faible tels que les prédicats SKOS (Miles & Bechhofer (2009)) *skos*:*exactMatch*, *skos*:*closeMatch*, *skos*:*broadMatch* ou encore *skos*:*narrowMatch*. Ces prédicats ne permettent néanmoins pas de répondre au problème de l'identité contextuelle. De plus, les prédicats SKOS ne peuvent être utilisés que pour des URI dont le type est un concept SKOS ce qui limite les cas d'utilisation possibles dans le LOD. Dans (Halpin *et al.* (2010)), les auteurs ont développé une ontologie de l'identité dans laquelle treize prédicats ont été hiérarchisés par la relation *rdfs*:*subPropertyOf* et caractérisés par les propriétés de réflexivité, de transitivité et de symétrie. Parmi ces prédicats, on trouve les prédicats SKOS cités précédemment mais également de nouveaux prédicats comme *id*:*claimsIndentical*, *id*:*matches* et *id*:*similar*. Le prédicat *owl*:*sameAs*, qui est réflexif, symétrique et transitif, est présenté au niveau le plus spécifique (i.e., le plus strict) et au niveau le plus général on trouve le prédicat *id*:*claimsSimilar* (réflexif, non-symétrique et non-transitif) et le prédicat *id*:*claimsRelated* (non-réflexif, non-symétrique et non-transitif). Les prédicats préfixés par le mot *claims* expriment une relation d'identité ou de similarité subjective dont la véracité dépend du contexte et/ou de l'interprétation du décideur humain. Bien que cette ontologie soit précise du point de vue structuration des relations d'identité et de similarité, elle ne permet néanmoins pas d'expliciter les contextes dans lesquels une relation d'identité serait valide. Dans le but d'utiliser des liens d'identité dans des contextes spécifiques, les auteurs de (Halpin *et al.* (2015)) ont proposé l'utilisation de graphes nommés pour représenter les contextes. Les différents contextes pertinents ne sont pas toujours faciles à expliciter, même pour un expert de domaine. Aussi, plus récemment, une approche développée par (Beek *et al.* (2016)) permet de représenter l'ensemble des contextes possibles dans lesquels un lien d'identité pourrait être valide. Un contexte associé à un lien d'identité entre deux instances i_1 et i_2 correspond à un sous-ensemble de propriétés pour lesquelles i_1 et i_2 doivent avoir les mêmes valeurs. Cependant, dans cette approche, un contexte est un ensemble non structuré de propriétés qui ne tient pas compte de l'organisation multi-classes de données RDF associées à une ontologie. Par ailleurs, les auteurs ne présentent pas d'algorithme permettant de calculer les liens d'identité contextuels.

3 Une relation d'identité contextuelle guidée par l'ontologie et des connaissances expertes

L'objectif de notre approche est de découvrir des liens d'identité valides dans un contexte qui peut être défini comme une sous-partie d'une ontologie de domaine. Dans cette section, nous présentons tout d'abord le modèle de données sur lequel s'appuie notre approche. Puis, nous définissons les notions de contextes locaux et globaux utilisés pour représenter la relation d'identité proposée. Enfin, nous donnons la définition de la relation d'identité contextuelle que l'approche que nous avons développée permet de découvrir.

3.1 Modèle de données

Notre approche de détection de liens d'identité contextuelle s'appuie sur une base de connaissances où l'ontologie est représentée en OWL [2] et les données sont représentées en RDF. Une

2. https ://www.w3.org/OWL/

base de connaissances $\mathcal{B}$ est définie par un couple $(\mathcal{O}, \mathcal{F})$ où :

— $\mathcal{O} = (\mathcal{C}, \mathcal{DP}, \mathcal{OP}, \mathcal{A})$ représente la partie conceptuelle de la base de connaissances définie par un ensemble de classes $\mathcal{C}$, un ensemble $\mathcal{DP}$ de propriétés de type *owl*:*DataTypeProperty*, un ensemble $\mathcal{OP}$ de propriétés de type *owl*:*ObjectProperty* et par un ensemble d'axiomes $\mathcal{A}$ tels que la relation de subsomption entre classes, la disjonction entre classes ou encore la fonctionnalité des propriétés.

— $\mathcal{F} = \{(s, p, o)\}$ est une collection de triplets (faits) de la forme *(sujet, propriété, objet)*, exprimant des liens entre deux instances de classe ou une instance et une valeur littérale. On notera $\mathcal{I}$ l'ensemble des instances de $\mathcal{C}$ correspondant à l'ensemble des sujets s et des objets o tels que $\exists(s, p, o) \in \mathcal{F}$.

Une ontologie OWL peut être représentée par un graphe $\mathcal{G} = (V, E)$ où l'ensemble de sommets V sont les classes et les types de données élémentaires (e.g. String, Date, Integer), et les arcs E sont les propriétés liant les classes entre elles ou liant les classes à des types de données élémentaires.

3.2 Présentation du problème de détection de liens d'identité contextuels

Le problème de détection de liens d'identité contextuels auquel nous nous intéressons peut être défini comme suit : étant donnée une base de connaissances $\mathcal{B} = (\mathcal{O}, \mathcal{F})$ et l'ensemble $\mathcal{I}^c$ des instances d'une classe cible c de l'ontologie $\mathcal{O}$, trouver pour l'ensemble des paires d'instances $(i_1, i_2) \in (\mathcal{I}^c \times \mathcal{I}^c)$ les contextes les plus spécifiques pour lesquels (i_1, i_2) sont identiques.

Par exemple, on s'intéresse en Figure 1 à l'identité des individus de la classe cible Jus pour laquelle deux instances jus1 et jus2 sont représentées. L'un des contextes dans lequel les

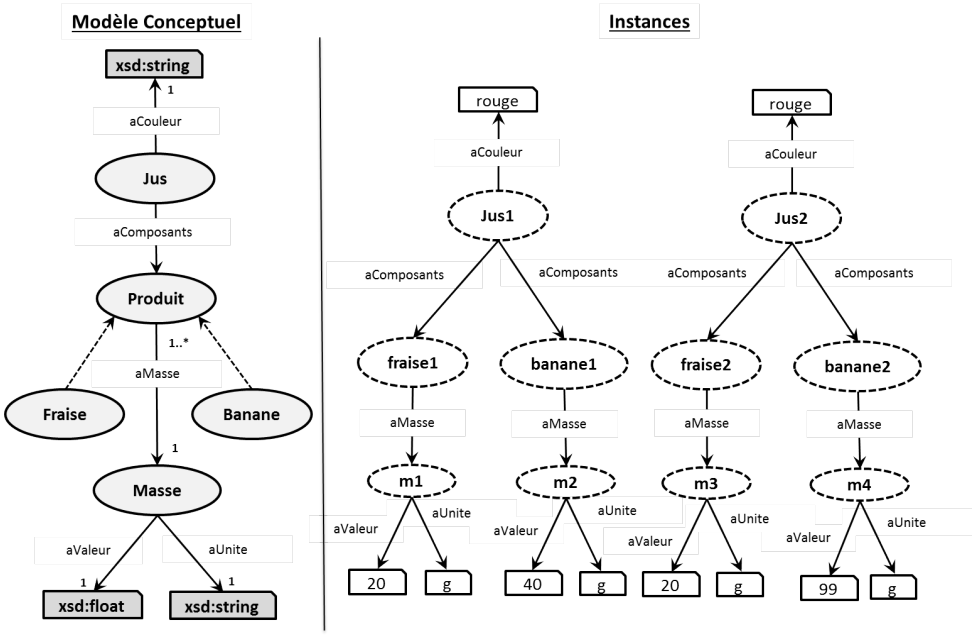

FIGURE 1 – Un extrait d'ontologie O, deux instances jus1 et jus2 de la classe cible *Jus*.

instances de jus de fruit $jus1$ et $jus2$ sont identiques est celui où l'on considère tous les produits

composant le jus, et pour chaque produit une masse dont la description est réduite à l'unité de mesure, mais sans considérer leur quantité. Dans un deuxième contexte, la masse de Fraise est considérée (avec sa valeur et son unité) et l'on observera la présence de banane mais on ne considérera pas la masse pour la banane.
Nous nous intéressons à la recherche de contextes communs correspondant à un sous-ensemble de classes et où pour chaque classe un sous-ensemble de propriétés de l'ontologie peut être pris en compte [3]. Enfin, certains contextes sont clairement plus pertinents que d'autres (e.g. une valeur de masse sans unité n'a pas de sens). Nous considérons également une recherche de liens d'identités contextuels qui puisse être guidée par certaines connaissances expertes.

3.3 Contextes

L'approche de détection de liens d'identité contextuelle que nous avons développée utilise une notion de contexte global fondée sur la notion de contexte local qui peut être définie pour une classe quelconque de l'ontologie.

Définition (**Contexte Local**). Étant donnée une classe c, un contexte local peut être défini par $\pi(c) = (c, DP^c, OP^c)$ où DP^c est un ensemble de propriétés de type *owl*:*DataTypeProperty* telles que c apparaît en domaine, ensemble auquel peuvent s'ajouter la propriété rdf:Type ainsi que des propriétés d'annotations (e.g., *rdfs*:*label*). OP^c est un ensemble de propriétés de type *owl*:*ObjectProperty* telles que c apparaît en domaine ou en co-domaine (notée alors op^{-1}).

Par exemple, un des contextes locaux de la classe $Masse$ présenté en Figure 1 est :
$\pi(Masse) = (Masse, \{rdf\ :type, aValeur, aUnite\}, \{aMasse^{-1}\})$

Définition (**Relation d'ordre sur les contextes locaux**). Soient $\pi_1(c)$ et $\pi_2(c)$ deux contextes locaux pour la classe c. Le contexte $\pi_1(c)$ est dit plus spécifique que $\pi_2(c)$, noté $\pi_1(c) \leqslant \pi_2(c)$, si et seulement si : $DP_2^c \subseteq DP_1^c$ et $OP_2^c \subseteq OP_1^c$.

Par exemple, soient deux contextes locaux de la classe Masse π_1 et π_2 tels que :
$\pi_1(Masse) = (Masse, \{rdf\ :type, aValeur, aUnite\}, \{aMasse^{-1}\})$ et
$\pi_2(Masse) = (Masse, \{rdf\ :type, aValeur\}, \emptyset)$, on a alors $\pi_1(Masse) \leqslant \pi_2(Masse)$.
L'ensemble des contextes locaux d'une classe c peut être représenté par un treillis de contextes locaux noté $T(c)$ qui est composé de 2^n contextes locaux, où n est le nombre de propriétés.

Pour réduire le nombre de contextes locaux à considérer (et donc le nombre de contextes globaux), nous prenons en compte des connaissances expertes, quand elles sont disponibles, concernant l'inutilité ou la nécessité de certaines propriétés ainsi que des informations sur l'importance de la cooccurrence de certaines d'entre elles. Plus précisément, pour chaque classe, un expert peut alimenter trois types de listes :

– $unwantedProps$: la liste des propriétés non pertinentes, qui n'interviendront pas dans

3. Il ne s'agit pas ici de calculer le graphe le plus spécifique partagé par deux instances de Jus dans lequel les descriptions de classes pourraient varier selon les instances considérées (e.g. les propriétés prises en compte pour comparer les instances de la classe Masse ne doivent pas varier selon que l'on compare la masse de la banane ou celle de la fraise)

le calcul des liens d'identité, soit parce qu'il s'agit d'une propriété non structurée (textuelle), ou parce que ses valeurs sont très hétérogènes ou encore parce que ses variations ne sont pas significatives compte tenu de l'utilisation des liens d'identité par l'expert (e.g. la couleur du jus de fruit). Ainsi, si l'on a ajouté la propriété $rdf:type$ à $unwantedProps$ avant la construction de $T(Masse)$, le contexte le plus spécifique sera $\pi_k(Masse) = (Masse, \{aValeur, aUnite\}, \{aMasse^{-1}\})$ et $T(Masse)$ contiendra seulement 8 contextes locaux.

– $coProps$: ensembles de propriétés devant être prises en compte ensemble, si elles sont considérées. Par exemple, une valeur de masse n'ayant pas de sens sans son unité de mesure, le couple $\{aValeur, aUnit\}$ peut être ajouté par l'expert à $coProps$. Dans ce cas, $T(Masse)$ se réduira à quatre contextes locaux possibles.

– $necProps$: ensemble des propriétés essentielles pour la comparaison des instances quel que soit le contexte considéré. Ainsi, si on ajoute $\{rdfType, aValeur, aUnit\}$ à $necProps$, $\pi_k(Masse) = (Masse, \{rdf:type, aValeur, aUnite\}, \{\emptyset\})$ sera le contexte le moins spécifique de $T(Masse)$.

Définition (**Contexte Global**). Etant donnée une classe cible $cbl \in \mathcal{C}$, un contexte global $\Pi(cbl)$ est un sous-graphe connexe G de $\mathcal{G}$ contenant cbl, tel que : $\Pi(cbl) = \bigcup_{c_k \in C_G} \pi(c_k)$

Dans l'exemple présenté en figure 1, il existe différents contextes globaux dans lesquels les deux individus de la classe cible ***Jus*** sont identiques. On en citera deux :

– $\Pi_a(Jus) = \{(Jus, \{rdf:type\}, \{aComposants\}),$
$(Fraise, \{rdf:type\}, \{aComposants^{-1}\}), (Banane, \{rdf:type\}, \{aComposants^{-1}\})\}$

– $\Pi_b(Jus) = \{(Jus, \{rdf:type, aCouleur\}, \{aComposants\}),$
$(Fraise, \{rdf:type\}, \{aComposants^{-1}, aMasse\},$
$(Banane, \{rdf:type\}, \{aComposants^{-1}\}, (Masse, \{rdf:type, aValeur, aUnite\}, \emptyset)\}$

Les contextes globaux que nous considérons ne contiendront pas deux classes c_1 et c_2, telles que $c_1 \subseteq c_2$. Cette contrainte permet d'éviter de construire des contextes globaux ne respectant pas les mécanismes d'héritage des propriétés. Pour cela nous sélectionnons les classes les plus générales pour lesquelles des instances ont été typées directement. Par exemple, s'il existe des instances directes de la classe $Produit$, les contextes globaux ne contiendront plus de contextes locaux définis pour les classes $Fraise$ et $Banane$ qui sont plus spécifiques. Les instances de ces classes seront représentées par le contexte local de la classe $Produit$. Le contexte global est alors plus abstrait et plus contraint puisqu'on ne pourra pas distinguer les contextes locaux de la classe $Fraise$ et de la classe $Banane$, comme pour le contexte global Π_b.

Définition (**Relation d'ordre sur les contextes globaux**). Soient $\Pi_1(cbl)$ et $\Pi_2(cbl)$ deux contextes globaux pour la classe cbl. Le contexte global $\Pi_1(cbl)$ est dit plus spécifique que $\Pi_2(cbl)$, noté $\Pi_1(cbl) \leqslant \Pi_2(cbl)$, si et seulement si : $\forall \pi_i(c) \in \Pi_2(cbl), \exists \pi_j(c) \in \Pi_1(cbl)$ tel que $\pi_j(c) \leqslant \pi_i(c)$.

Dans l'exemple ci-dessus, $\Pi_b(jus) \leqslant \Pi_a(jus)$, puisque $\pi_b(Jus) \leqslant \pi_a(Jus)$, $\pi_b(Fraise) \leqslant \pi_a(Fraise)$ et $\pi_b(Banane) \leqslant \pi_a(Banane)$.

3.4 Relation d'identité contextuelle

Une relation d'identité contextuelle est valide dans un contexte global si toutes les valeurs littérales (à une équivalence près) et toutes les instances appartenant à ce contexte sont identiques. Cette relation peut être exprimée de la façon suivante :

Définition (**Relation d'identité contextuelle**). Soient (i_1, i_2) deux instances d'une classe cible cbl. Soit $\Pi(cbl) = \{\pi_1(c_1), \ldots, \pi_n(c_n)\}$ un contexte global pour la classe cbl. Deux instances (i_1, i_2) sont liées par une relation d'identité dans le contexte global $\Pi(cbl)$, notée $identiConTo_{<\Pi(cbl)>}(i_1, i_2)$, si et seulement si les sous-graphes RDF décrivant i_1 et i_2 et obtenus en utilisant la partie de l'ontologie représentée dans le contexte global, sont identiques, au renommage d'URI près et à une réécriture de valeurs littérales près.

Les relations d'identités ne seront représentées que pour les contextes les plus spécifiques et pourront être générées pour des contextes plus généraux si besoin. Soient $\Pi_1(cbl)$ et $\Pi_2(cbl)$ deux contextes globaux d'une classe cible cbl. Si $\Pi_1(cbl) \leqslant \Pi_2(cbl)$ alors $identiConTo_{<\Pi_1(cbl)>}(i_1, i_2) \Rightarrow identiConTo_{<\Pi_2(cbl)>}(i_1, i_2)$

4 DECIDE - Un algorithme de détection de liens d'identité contextuels

Le but de l'algorithme $DECIDE$ (DEtection of Contextual IDEntity) est de déterminer pour chaque couple d'instances (i_1,i_2) $\in I \times I$ d'une classe cible cbl fixée par l'utilisateur, l'ensemble des contextes globaux les plus spécifiques pour lesquels la relation $identiConTo$ est vraie. $DECIDE$ prend en paramètres la base de connaissances $\mathcal{B}$, la classe cible cbl, et les trois listes de contraintes des experts $unwantedProps$, $coProps$ et $necProps$ si elles existent. La construction des contextes globaux se déroulent en trois étapes :

- Construction de la liste $Cdep$ des classes les plus générales du graphe connexe maximal de cbl qui comportent des instances directement typées par ces classes.
- Pour chaque classe $c \in Cdep$, construction des treillis de contextes locaux $T(c)$ pertinents en tenant compte des listes $unwantedProps$, $coProps$ et $necProps$ définies par les experts.
- Pour chaque couple d'instances (i_1,i_2) de cbl, appel de la fonction $IdentiConMax$ détaillée dans l'algorithme 1 qui calcule l'ensemble des contextes globaux les plus spécifiques ($CGSet$).

Pour chaque couple d'instances (i_1,i_2), $IdentiConMax$ retourne l'ensemble des contextes globaux les plus spécifiques, dans lesquels les instances du couple (i_1,i_2) sont identiques. Cette fonction effectue un parcours en profondeur d'abord des propriétés décrivant les instances à comparer et construit au fur et à mesure les contextes globaux les plus spécifiques.

Algorithme 1 : identiConToMax

Inputs :
- cbl : la classe cible
- $Cdep$: l'ensemble des classes du graphe connexe maximal de cbl
- (i_1, i_2) : une paire d'instances de la classe c

Output : $CGSet$: l'ensemble des contextes globaux les plus spécifiques concernant le couple (i_1, i_2)

$CGSet \leftarrow \emptyset$; $\Pi(cbl) \leftarrow \emptyset$; $fileCP.add(\Pi(cbl))$

while $fileCP$ *is not empty* **do**

  $\Pi(cbl)^{cour} \leftarrow fileCP.getNextElement()$

  $dejaVu \leftarrow \emptyset$; $propSrc \leftarrow nil$; $cSrc \leftarrow nil$

  $\Pi(cbl)^{cour} = compareCouple(i_1, i_2, dejaVu, propSrc, cSrc, \Pi(cbl)^{cour})$

  $CGSet \leftarrow CGSet \cup \Pi(cbl)^{cour}$

Plus précisément, l'algorithme commence avec un contexte global courant vide $\Pi(cbl)^{cour}$ qui sera enrichi par les contextes locaux au fur et à mesure de l'exploration. Pour ce contexte, on appelle la fonction $compareCouple$ qui permet de comparer un couple d'instances d'une même classe c. Une file de contextes globaux partiels, notée $fileCP$, contient les contextes globaux alternatifs qui seront détectés lors de l'exploration et qui ne sont pas encore traités.

La fonction $compareCouple$, décrite dans l'algorithme 2, permet d'enrichir un contexte courant pour trouver un contexte plus spécifique dans lequel les instances du couple (i_1,i_2) sont identiques. Les données étant représentées sous la forme d'un graphe pouvant comporter des cycles, une liste de paires d'instances déjà explorées nommée $dejaVu$ est maintenue afin d'éviter de recalculer les contextes identiques pour ces paires. La fonction $comparerCouple$ prend en paramètre la liste $dejaVu$, la propriété objet source $propSrc$ qui a permis d'atteindre le couple (i_1,i_2) à comparer, la classe source $cSrc$ et le contexte global en cours $\Pi(cbl)^{cour}$ ($dejaVu$, $propSrc$ et $cSrc$ seront réinitialisés à nil pour chaque nouveau contexte courant exploré). Cette fonction détermine les propriétés de type *DataTypeProperty* et les propriétés d'annotation dont les valeurs sont identiques (noté $DPeg$)[4]. Elle détermine également l'ensemble des propriétés objets $OPcom$ instanciées pour i_1, et i_2. La fonction $possibleContext$ va ensuite rechercher dans le treillis $T(c)$ si ce contexte local $\pi(c)^{cour}$ est possible (compte tenu des connaissances expert). S'il n'existe pas un tel contexte, le contexte local de $T(c)$ le plus spécifique généralisant $\pi(c)^{cour}$ est retourné. Si ce contexte retourné est le contexte vide, la fonction s'arrête et retourne le contexte global en cours $\Pi(cbl)^{cour}$. Si le contexte retourné de la fonction $possibleContext$ n'est pas vide, 4 cas sont possibles :

1 – Il n'existe pas de contexte local déjà défini pour c dans $\Pi(cbl)^{cour}$, on ajoute alors $\pi(c)^{cour}$ à $\Pi(cbl)^{cour}$. Ensuite, la fonction $compareObjProperties$, est appelée pour comparer tous les ressources reliées au couple (i_1, i_2). Par exemple, si i_1=$Jus1$ et i_2=$Jus2$, la fonction $compareObjProperties$ va comparer les couples $(fraise1, fraise2)$ puis $(banane1, banane2)$.

2 – Il existe un contexte local identique $\pi(c)^{existant}$ déjà défini pour c. On appelle alors directement $compareObjProperties$.

3 – Il existe un contexte local pour c dans $\Pi(cbl)^{cour}$ qui est plus général que $\pi(c)^{cour}$. Dans

4. Une mesure de similarité peut être utilisée et différer suivant chaque propriété

ce cas, on crée un nouveau contexte global $\Pi(cbl)^{nouv}$ contenant le contexte local $\pi(c)^{cour}$, et on l'ajoute à la file $fileCP$ qu'il restera à explorer. Ensuite, comme $\pi(c)^{cour} \leqslant \pi(c)^{existant}$, on doit vérifier si les instances ont bien les mêmes valeurs pour les *objectProperties* de $\pi(c)^{existant}$ en appelant la fonction $compareObjProperties$.

4 – Dans tous les autres cas, on crée un nouveau contexte global $\Pi(cbl)^{nouv}$ contenant le contexte local $\pi(c)^{cour}$, et on l'ajoute à la file $fileCP$ pour qu'il soit exploré dans une prochaine itération de l'algorithme 1. Dans le contexte courant, comme on est sûr que le couple n'est pas identique en considérant $\pi(c)^{existant}$, le contexte de la classe source src $\pi(src)^{existant}$ est remplacé par un nouveau contexte local $\pi(src)^{nouv}$ ne contenant pas la propriété $propSrc$ dans $\Pi(cbl)^{cour}$.

Algorithme 2 : compareCouple

Inputs :
- (i_1, i_2) : une paire d'instances
- $dejaVu$: la liste des couples d'instances déjà traités
- $propSrc$: l'object property source
- $cSrc$: la classe source (i.e., le domaine ou le co-domaine de $propSrc$)
- $\Pi(cbl)^{cour}$: le contexte global courant de la classe cible.

Output : $\Pi(cbl)^{cour}$: le contexte global courant de la classe cible, mis à jour.

$dejaVu \leftarrow dejaVu \cup (i_1, i_2)$;
$c \leftarrow$ la classe commune de i_1 et i_2 tel que $c \in Cdep$;
$OPcom \leftarrow$ les propriétés objets communes de i_1 et i_2;
$DPeg \leftarrow$ les data properties, rdftype et propriétés d'annotation ayant mêmes valeurs pour i_1 et i_2;
$\pi(c)^{cour} \leftarrow possibleContext(c, DPeg, OPcom)$
if $\pi(c)^{cour}$ *!=* $\emptyset$ **then**
  if *(Il n'existe pas de contexte local de c dans* $\Pi(cbl)^{cour}$*)* **then**
    $\Pi(cbl)^{cour}.add(\pi(c)^{cour})$
    $compareObjProperties(i_1, i_2, c, dejaVu, propSrc, cSrc, \Pi(cbl)^{cour})$;
  else
    $\pi(c)^{existant} \leftarrow getLocalContext(c, \Pi(cbl)^{cour})$
    if $\pi(c)^{existant} == \pi(c)^{cour}$ **then**
      $compareObjProperties(i_1, i_2, c, dejaVu, propSrc, cSrc, \Pi(cbl)^{cour})$;
    else
      $\Pi(cbl)^{nouv} \leftarrow \emptyset$
      $\Pi(cbl)^{nouv}.add(\pi(c)^{cour})$
      if $\Pi(cbl)^{nouv}$ *n'existe pas dans* $fileCP$ **then**
        $fileCP \leftarrow fileCP.add(\Pi(cbl)^{nouv})$
      if $\pi(c)^{cour} \leqslant \pi(c)^{existant}$ **then**
        $compareObjProperties(i_1, i_2, c, dejaVu, propSrc, cSrc, \Pi(cbl)^{cour})$;
      else
        $\pi(cSrc) \leftarrow getLocalContext(cSrc, \Pi(cbl)^{cour})$
        $\pi(cSrc)^{nouv} \leftarrow possibleContext(cSrc, \pi(cSrc).DPeg, \pi(cSrc).OPcom - propSrc)$
        $\Pi(cbl)^{cour} \leftarrow replaceLocalContext(cSrc, \Pi(cbl)^{cour}, \pi(cSrc)^{nouv})$
return $\Pi(cbl)^{cour}$;

La fonction $compareObjProperties$ appelée dans $identiConToMax$ permet de comparer l'ensemble des instances reliées au couple (i_1, i_2) pour chacune des propriétés p du contexte local. Les listes des instances liées à i_1 et i_2 sont examinées classe par classe et les propriétés sont supposée être localement complètes pour chacune des classes (e.g. si l'on connait certains fruits composants un jus, nous supposons qu'ils sont tous représentés). Pour chacune des classes, afin

	Echantillon	***D. Réelles***
#Faits	24 458	1 269 624
#Classes	4 635	4 743
#Instances	876	272 724
#ObjectProperties	50	50
#DataProperties	11	11
#Annot.Properties	25	25

	Echantillon	***D. Réelles***
# Instances (cible)	22	220
# Paires d'instances	231	24 090
# Classes (graphe)	128	306
# Instances (graphe)	606	90 384
# Contextes Globaux	8	30
# Liens d'identité	252	25 189
Temps d'exécution	9 s	372 s

TABLE 1 – Taille des Datasets et Résultats

de limiter le nombre de paires à examiner en cas de multivaluation, on vérifie tout d'abord que le nombre de valeurs de p est identique, puis on utilise une heuristique, qui permet de ne former que les couples (i'_1, i'_2) qui partagent le plus de valeurs de *data properties*. Après avoir formé la liste des meilleures paires, la fonction rappelle la fonction $compareCouple$ pour chacune de ces paires afin de trouver les contextes les plus spécifiques dans lesquels ces paires sont identiques.

L'algorithme $DECIDE$ appliqué sur l'exemple de la figure 1 permet de trouver les contextes globaux les plus spécifiques suivants pour $Jus1$ et $Jus2$:
$\Pi_a(Jus) = \{(Jus, \{rdf : type, aCouleur\}, \{aComposants\}), (Fraise, \{rdf : type\}, \{aComposants^{-1}, aMasse\}), (Banane, \{rdf : type\}, \{aComposants^{-1}\}), (Masse, \{rdf : type, aValeur, aUnite\}, \{aMasse^{-1}\})\}$ et $\Pi_b(Jus) = \{(Jus, \{rdf : type, aCouleur\}, \{aComposants\}), (Fraise, \{rdf : type\}, \{aComposants^{-1}, aMasse\}), (Banane, \{rdf : type\}, \{aComposants^{-1}, aMasse\}), (Masse, \{rdf : type, aUnite\}, \{aMasse^{-1}\})\}$

5 Expérimentations

Notre approche a été évaluée sur des données scientifiques relatives au domaine de l'agro-alimentaire. Nous avons exploité la version 1.4 de l'ontologie PO^2 (Ibanescu *et al.*, 2016) qui décrit des processus de transformations [5]. Dans le cadre du projet INRA CellExtraDry, PO^2 a été enrichie par une partie d'Agrovoc et la version de PO^2 a été peuplée par 10 processus de transformation de micro-organismes (levures) suivant 20 itinéraires où chaque itinéraire représente des séquences d'étapes de transformation (séchage, chauffage ...). Les données réelles ne pouvant être mises à disposition, pour des raisons de confidentialité, un échantillon de ces données, décrivant un seul processus de stabilisation et dont certaines valeurs ont été modifiées, est accessible avec le code en $Java$ de l'algorithme et ses résultats sur `https://github.com/raadjoe/DECIDE`. La taille des données réelles du projet et celle de l'échantillon sont décrites dans la Table 1. Cette première expérimentation a eu pour but d'observer le nombre de contextes distincts et le nombre de liens d'identité contextuels pouvant être générés dans une ontologie décrivant des données scientifiques. Dans cette expérimentation, une égalité stricte des valeurs littérales a été utilisée (à une normalisation près).

L'algorithme $DECIDE$ a été appliqué pour chacun de ces datasets afin de découvrir les contextes les plus spécifiques pour lesquels les individus de la classe cible $Mixture$ sont identiques. Une instance de cette classe représente un ensemble de produits sur lequel est appliqué une étape d'un processus de transformation. La taille de la classe cible pour chaque dataset, la

5. L'ontologie core de PO^2 est accessible sur `http://agroportal.lirmm.fr/ontologies/PO2`

taille du graphe connexe maximal (i.e. classes et individus pouvant être atteints depuis la classe cible), et les résultats de l'algorithme sont présentés dans la Table 1. L'algorithme a été exécuté sur une machine de 8GB de RAM, ayant un processeur Intel Core 4× 2.6GHz (Windows 10).

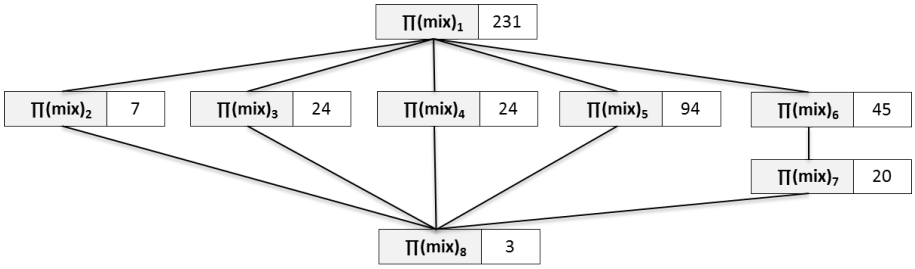

FIGURE 2 – La hiérarchie des 8 contextes globaux les plus spécifiques de la classe Mixture.

Les 220 instances de la classe cible $Mixture$ des données réelles, forment 24 090 couples d'individus candidats. 30 contextes globaux générés par $DECIDE$ suffisent pour représenter les contextes les plus spécifiques dans lesquels ces couples sont identiques (8 pour l'échantillon). Le nombre de liens d'identité (25 189), plus élevé que le nombre de couples d'individus, montre que certains couples sont identiques dans plusieurs contextes globaux non comparables. La hiérarchie des 8 contextes globaux créés dans l'échantillon est représentée dans la Figure 2. Le nombre qui apparait à droite de chaque contexte global dans cette figure représente le nombre de couples qui sont identiques dans ce contexte (après inférence). Le contexte $\Pi(mix)_1 = \{(Mixture, \{rdf : type\})\}$, qui est le contexte global le moins spécifique de tous les contextes générés, est un contexte dans lequel tous les couples sont identiques (231 pour l'échantillon). Le contexte le plus spécifique qui a été généré ($\Pi(mix)_8$), contient 11 contextes locaux, et peut garantir une identité plus forte. Un couple de mixtures identiques dans ce contexte signifie qu'elles sont de même nature (e.g humides) et partagent les mêmes quantités et unités de mesure de leurs composants : eau, silicone, levure, glucose, et cystéine.

En ré-appliquant l'algorithme $DECIDE$, et en prenant compte cette fois d'une seule connaissance experte qui indique qu'une valeur de masse n'a pas de sens sans son unité de mesure et vice-versa, les contextes locaux de 17 contextes globaux parmi 30 des données réelles, et 5 parmi 8 contextes globaux de l'échantillon étaient affectés. Ce qui indique que l'ajout d'une seule connaissance au début de l'algorithme a permis d'améliorer la sémantiques d'environ 60% des contextes globaux générés, sans affectant le nombre des contextes pour les deux datasets.

6 Conclusion

Nous proposons une approche de détection de liens d'identité contextuels permettant de générer pour chaque couple instance d'une classe cible, l'ensemble des contextes les plus spécifiques dans lesquels ces instances sont identiques. Une première expérimentation a été réalisée pour des données réelles décrivant des expérimentations scientifiques sur la levure, menées à l'INRA. Cette expérimentation a montré que différents contextes à différents niveaux d'abstraction pouvaient être découverts en un temps raisonnable.

Nous souhaitons maintenant montrer les résultats obtenus aux experts de domaine pour connaître l'intérêt des différents contextes et recueillir le maximum de contraintes permettant

d'élaguer l'espace de recherche. Cette collecte pourrait être facilité par la création d'un outil permettant d'interagir avec les experts du domaine. Nous souhaitons ensuite utiliser ces liens pour faire prédire des valeurs manquantes à différents niveaux de confiance selon le niveau de spécificité du contexte d'identité. Il nous envisageons également de découvrir relations de causalité entre certains faits décrivant les expériences et ceux décrivant les résultats des observations en considérant l'identité contextuelle des autres facteurs expérimentaux.

Remerciements

Ce travail est soutenu par le Center for Data Science, financé par IDEX Paris-Saclay, ANR-11-IDEX-0003-02.

Références

BEEK W., SCHLOBACH S. & HARMELEN F. (2016). A contextualised semantics for owl : Sameas. In *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*, p. 405–419, New York, NY, USA : Springer-Verlag New York, Inc.

DE MELO G. (2013). Not quite the same : Identity constraints for the web of linked data. In M. DESJARDINS & M. L. LITTMAN, Eds., *AAAI* : AAAI Press.

DING L., FININ T., SHINAVIER J. & MCGUINNESS D. L. (2010). owl :sameAs and linked data : An empirical study. In *In The Semantic Web - ISWC*, p. 145–160.

FERRARA A., NIKOLOV A. & SCHARFFE F. (2011). Data linking for the semantic web. *Int. J. Semantic Web Inf. Syst.*, **7**(3), 46–76.

HALPIN H., HAYES P. J., MCCUSKER J. P., MCGUINNESS D. L. & THOMPSON H. S. (2010). When owl :sameas isn't the same : An analysis of identity in linked data. In P. F. PATEL-SCHNEIDER, Y. PAN, P. HITZLER, P. MIKA, L. ZHANG, J. Z. PAN, I. HORROCKS & B. GLIMM, Eds., *The Semantic Web – ISWC 2010 : 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, p. 305–320, Berlin, Heidelberg : Springer Berlin Heidelberg.

HALPIN H., HAYES P. J. & THOMPSON H. S. (2015). When owl : sameas isn't the same redux : towards a theory of identity, context, and inference on the semantic web. In *International and Interdisciplinary Conference on Modeling and Using Context*, p. 47–60 : Springer.

IBANESCU L., DIBIE J., DERVAUX S., GUICHARD E. & RAAD J. (2016). Poˆ 2-a process and observation ontology in food science. application to dairy gels. In *Metadata and Semantics Research : 10th International Conference, MTSR 2016, Göttingen, Germany, November 22-25, 2016, Proceedings*, p. 155–165 : Springer.

JAFFRI A., GLASER H. & MILLARD I. (2008). Uri disambiguation in the context of linked data. In C. BIZER, T. HEATH, K. IDEHEN & T. BERNERS-LEE, Eds., *Linked Data on the Web - LDOW*, volume 369 of *CEUR Workshop Proceedings* : CEUR-WS.org.

MILES A. & BECHHOFER S. (2009). Skos simple knowledge organization system reference. w3c recommendation 18 august 2009.

PAPALEO L., PERNELLE N., SAÏS F. & DUMONT C. (2014). Logical detection of invalid sameas statements in RDF data. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, p. 373–384.

PATEL-SCHNEIDER P. F., HAYES P. & HORROCKS I. (2004). *OWL Web Ontology Language Semantics and Abstract Syntax Section 5. RDF-Compatible Model-Theoretic Semantics*. Rapport interne, W3C.