Création et mise à jour guidées d'objets dans une base RDF(S)

Alice Hermann, Sébastien Ferré, Mireille Ducassé

Création et mise à jour guidées d'objets dans une base RDF(S)

Alice Hermann[1], Sébastien Ferré[2], and Mireille Ducassé[1]

[1] IRISA – INSA Rennes, Campus de Beaulieu, 35708 Rennes cedex 7, FRANCE
alice.hermann@irisa.fr, mireille.ducasse@irisa.fr

[2] IRISA – Université de Rennes 1, Campus de Beaulieu, 35042 Rennes cedex, FRANCE
sebastien.ferre@irisa.fr

Résumé : La mise à jour des bases de connaissances existantes est cruciale pour tenir compte des nouvelles informations, régulièrement découvertes. Toutefois, en pratique, les données actuelles du Web Sémantique sont rarement mises à jour par les utilisateurs. Nous proposons UTILIS, une méthode pour aider les utilisateurs à ajouter de nouveaux objets. Un objet est une ressource de la base. Sa description correspond aux propriétés qu'il possède. Pendant la création d'un nouvel objet *o*, UTILIS recherche les objets similaires. Les propriétés des objets similaires sont utilisées comme suggestions pour compléter la description de *o*. Les objets similaires sont trouvés en appliquant des règles de relaxation à la description de *o*, prise comme une requête. Comparé avec l'état de l'art, la contribution est qu'UTILIS est à la fois incrémental, chaque nouvelle propriété est utilisée pour la recherche, et interactif, l'utilisateur a un rôle actif dans le processus.

Mots-clés : Web sémantique, données RDF(S), édition, mise à jour

1. Introduction

De nouvelles informations sont perpétuellement découvertes. Mettre à jour les données existantes du Web Sémantique (WS) est crucial pour prendre en compte l'évolution des informations. Toutefois, en pratique, les données du WS sont rarement mises à jour par les utilisateurs. Or, dans le Web 2.0, les utilisateurs sont au centre de la production de données avec les tags par exemple. Ce n'est donc pas un problème de motivation des utilisateurs mais un problème de difficulté à rentrer l'information directement sous une forme compatible avec le WS. Des modèles existent pour tirer parti des avantages des deux Web, en reliant par exemple par les tags produits par les utilisateurs

avec des ressources du WS (Passant & Laublet, 2008; Limpens *et al.*, 2009). Toutefois, il reste des problèmes, d'ambiguité par exemple. Pour les résoudre, il faut permettre aux utilisateurs d'entrer directement des données du WS.

Ce papier présente UTILIS (Updating Through Interaction in Logical Information Systems), une méthode pour aider les utilisateurs à créer et mettre à jour des objets du WS. La description partielle du nouvel objet est utilisée pour trouver les objets similaires, c'est-à-dire ayant des propriétés communes. Les propriétés de ces objets sont proposées à l'utilisateur comme suggestions. Quand aucun objet n'a toutes les propriétés connues du nouvel objet, la description est généralisée pour trouver des objets plus distants. Cette généralisation est obtenue en utilisant des règles élémentaires de relâchement, inspirées du travail de Hurtado *et al.* (2008). Lorsque l'utilisateur décide que la description du nouvel objet est complète, il est ajouté à la base et utilisé pour les futurs guidages. UTILIS utilise les systèmes d'information logiques (Ferré & Ridoux, 2004), permettant un guidage interactif et incrémental pour la recherche d'information. UTILIS est interactif, l'utilisateur est intégré dans l'ensemble du processus de mise à jour, et incrémental, tout nouvel ajout dans la description d'un objet est utilisé pour le guidage.

Comme détaillé dans la section sur l'état de l'art, des outils existent pour aider l'utilisateur à créer des bases de connaissance (Kuhn, 2009; Haller, 2010), leur guidage permet généralement de créer des triplets corrects syntaxiquement et en accord avec les règles et les axiomes de la base, mais il ne s'adapte pas finement au nouvel objet. La contribution de notre méthode est qu'UTILIS fournit un guidage avec un grain plus fin qu'un guidage fondé uniquement sur la syntaxe ou sur une ontologie. En se focalisant sur les objets existants, UTILIS tient compte de leurs spécificités et n'a ainsi pas besoin *à priori* d'une ontologie pour calculer les suggestions, ce qui permet d'être générique, non spécialisé à un domaine particulier. Il s'appuie sur un graphe RDF(S), en fait 95% des documents du WS utilisent uniquement le vocabulaire RDF(S) (d'Aquin *et al.*, 2007). Ce qui indique que la création d'une ontologie est complexe. Du point de vue de l'utilisabilité, il est donc préférable que cette création ne soit pas un prérequis à l'entrée des données, mais plutôt qu'elle s'entrelace avec la création des données. Toutefois, si une ontologie est présente, on peut coupler UTILIS avec des travaux complémentaires, par exemple avec un raisonneur logique (Giacomo *et al.*, 2006).

Les sections 2. et 3. donnent des définitions préliminaires relatives au WS et au langage de requête utilisé dans la suite, LISQL. La section 4. spécifie UTILIS. La section 5. présente l'algorithme pour calculer les suggestions. La

section 6. décrit l'ajout d'un nouvel objet dans une base généalogique, réalisé avec notre prototype. Dans tout l'article, les exemples feront référence à la figure 5 de la section 6. La section 7. compare notre approche à l'état de l'art.

2. Le Web Sémantique

Le web sémantique est fondé sur de nombreux langages de répresentation, comme RDF, RDFS et OWL. Ces langages ont des pouvoirs d'inférence croissants (Hitzler *et al.*, 2009). Les deux éléments de base de ces langages sont les ressources et les triplets. Une ressource peut être soit une URI *(Uniform Resource Identifier)*, nom absolu d'une ressource, soit un littéral (par exemple, une chaîne de caractères ou une date) ou une ressource anonyme. Nous définissons un objet comme une ressource autre qu'une classe, une propriété ou un littéral. Un triplet *(s p o)* est composé de 3 ressources. Il peut être lu comme une phrase où *s* est le sujet, *p* est le verbe, appelé prédicat, et *o* est l'objet. Par exemple, dans la base généalogique de la figure 5, le triplet *(*:*Maya* :*mother* :*Zoé)* dit que "Maya a pour mère Zoé".

RDF(S) introduit un vocabulaire de ressources pour représenter l'appartenance à une classe de ressources (*rdf*:*type*), la hiérarchie entre les classes (rdfs :subClassOf) et entre les propriétés (*rdfs*:*subPropertyOf*). Par exemple, le triplet *(*:*Maya rdf*:*type* :*woman)* dit que "Maya est de type femme", ou plus simplement "Maya est une femme". rdf :type est une propriété et :woman est une classe. Le triplet *(*:*woman rdfs*:*subClassOf* :*person)* dit que "La classe femme est une sous classe de personne", ou plus simplement que "Toute femme est une personne". Par inférence, ces deux triplets permettent de déduire que Maya est une personne *(*:*Maya rdf*:*type* :*person)*. L'ensemble des triplets d'une base de connaissances forme un graphe RDF(S).

3. Les Systèmes d'Information Logiques et le langage de requête LISQL

Les Systèmes d'Infomation Logiques (LIS) (Ferré & Ridoux, 2004) appartiennent à un paradigme de recherche d'information qui combine l'interrogation et la navigation. Les LIS sont formellement fondés sur une généralisation logique de l'Analyse Formelle de Concepts (FCA) (Ganter & Wille, 1999). Ils implémentent aussi le paradigme de la recherche par facettes (Sacco & Tzitzikas, 2009).

Le langage de requête LISQL (Ferré, 2010; Ferré *et al.*, 2011) offre une syntaxe de haut niveau reposant sur SPARQL (Prud'hommeaux & Seaborne,

Question	Quelles sont les femmes dont un des parents est né à Metz ?
LISQL	a woman and parent: birth :place :Metz
SPARQL	SELECT ?x WHERE { ?x rdf:type :woman. ?x parent ?y. ?y birth ?z. ?z place ?a. FILTER (?a = Metz)}

FIGURE 1: Traduction d'une question en LISQL et en SPARQL

2008) pour interroger les graphes RDF(S). Un exemple de question et sa traduction en une requête LISQL ainsi qu'en SPARQL est montré dans la figure 1. Une requête LISQL spécifie un objet par combinaison, au moyen des trois connecteurs booléens (*and, or, not*) : de noms (ex, *Metz*), d'appartenance à une classe (ex, *a woman*), et de propriétés. La valeur d'une propriété peut être un objet (ex, *place*: *Metz*) ou une requête LISQL (ex, *birth*: *(year*: *2000 and place*: *Metz)*). Une requête LISQL et chacune de ses sous-requêtes correspondent à une variable distincte dans la requête SPARQL. L'appartenance à une classe et les propriétés correspondent aux triplets dans la requête SPARQL, tandis que les noms correspondent aux contraintes de filtre. La variable dans la clause SELECT, le *focus*, est montrée dans la requête LISQL en soulignant la sous-requête correspondante. Dans la figure 1, le focus est sur *a woman*, c'est-à-dire les femmes. Si la partie soulignée était birth: place: Metz, le focus serait sur les parents de ces femmes.

LISQL est un langage de requête dont l'expressivité est proche de SPARQL, et qui est plus concis et plus convivial. Un sous-ensemble de LISQL, où seules les conjonctions sont autorisées, peut être utilisé pour les descriptions d'objets (voir la section 4.1.). LISQL couvre certains aspects de SPARQL 1.1 [1] en particulier, la négation et certaines expressions de chemins, comme la sequence et l'inverse. Dans la suite, LISQL est utilisé par les utilisateurs, alors que le graph pattern de SPARQL est utilisé au niveau de l'implémentation, par exemple dans l'algorithme qui calcule les requêtes généralisées.

4. Un guidage incrémental et interactif

Cette section décrit UTILIS, la méthode de guidage proposée pour aider les utilisateur à créer et à mettre à jour des objets dans un graphe RDF(S). Comme mentionné dans l'introduction, elle utilise la description partielle du nouvel objet pour trouver les objets similaires, c'est-à-dire ayant des propriétés com-

1. http://www.w3.org/TR/sparql11-query/

munes. La section 4.1. décrit comment la description peut être utilisée comme une requête afin de trouver ces objets, la 4.2. comment relâcher la requête quand elle n'a pas de réponse et la 4.3. les interactions avec l'utilisateur.

4.1. La description utilisée comme une requête

La description courante du nouvel objet est utilisée comme une requête pour trouver des objets similaires. Les résultats et leurs propriétés sont utilisés comme liens de navigation. Par exemple, le nouvel objet a pour description courante : *Maya D. is a woman*. La requête correspondante est *a woman*. Les résultats sont les femmes, et leurs propriétés sont utilisées comme liens de navigation. Dans la figure 5 étape 2, la propriété suggérée *(name*: *?)* est sélectionnée pour spécifier que Maya D. a un nom, comme les autres femmes connues. La nouvelle description est *Maya D. is a woman and name*: *?*, où le focus est sur le nom (pour être spécifié au prochain pas de navigation). La requête est maintenant *woman and name*: *?*, soit "Les femmes et leurs noms, c'est quoi ?". En reformulant par rapport au focus, on obtient la requête équivalente : *name of a woman*, soit "Quels sont les noms des femmes ?".

4.2. Le relâchement de la description

Toutefois, rapidement, la description du nouvel objet est unique dans la base et ne correspond à aucun autre objet de la base, de ce fait le résultat de la requête est vide. Il n'y a donc pas d'objet similaire, ni de suggestion. Dans notre exemple de base généalogique, deux personnes n'ont jamais toutes leurs caractéristiques en commun (nom, prénom, ...). Une fois que l'utilisateur aura spécifié le nom et le prénom de Maya D., il y a de grandes chances pour qu'aucun autre individu n'ait ces deux propriétés. C'est pour cette raison que nous proposons de généraliser la requête correspondant à la description et d'utiliser les résultats des requêtes généralisées comme liens de navigation.

Pour calculer les requêtes généralisées, nous prenons comme base de travail, les travaux de Hurtado *et al.* (2008) sur le relâchement de requêtes effectuées sur une base de connaissances. Le relâchement est une généralisation faite en combinant des règles élémentaires de relâchement. Les résultats d'une requête initiale est un sous-ensemble de la requête relâchée. Chaque règle remplace un triplet par un triplet vide ou par un triplet qui le subsume. La figure 2 présente notre ensemble de règles. La première colonne montre le nom de la règle, la deuxième le triplet avant relâchement, la troisième les possibles conditions pour l'application de la règle et la quatrième le triplet relâché. Par

exemple, la règle *SuperProperty* s'applique sur un triplet ayant une variable à la place du sujet et de l'objet et une ressource p_1 à la place du prédicat. La condition d'application de cette règle est que p_1 soit une sous-propriété d'une autre propriété p_2. Après l'application de cette règle, les éléments en sujet et en objet restent identiques mais p_1 est remplacé par p_2. Par exemple, le triplet (`?x` *mother* `?y`) peut être relâché en (`?x` *parent* `?y`) en appliquant la règle *SuperProperty* s'il existe le triplet (*mother rdfs*:*subPropertyOf parent*). Le triplet vide correspond à un triplet n'apportant aucune restriction. Par exemple, pour la règle *Resource* le triplet vide correspond à ($?r_1$ rdf:type rdfs:Resource), la variable r_1 est une ressource, or chaque élément de la base est une ressource.

La figure 3 montre l'ensemble du processus interactif mis en œuvre pour raffiner la description du nouvel objet. L'élément de départ du processus est la description partielle du nouvel objet. Cette description est utilisée comme requête initiale pour élargir l'ensemble des objets similaires autant que nécessaire. Les résultats des requêtes initiale et généralisées et leurs propriétés sont utilisés comme liens de navigation. Ces liens de navigation permettent de compléter la description du nouvel objet. L'utilisateur intervient dans le raffinement de la description, avec le choix de la suggestion à ajouter à la description. Cette nouvelle description complétée est ensuite utilisée comme nouvelle requête initiale et ainsi de suite.

Par exemple, dans la figure 5 étape 7, le nouvel objet Maya D. a la description suivante, *Maya D. is a woman and name*: *Maya and mother*: *Zoé D. and father*: *?*. Nous rappellons que la partie soulignée pointe le focus, l'élément que l'utilisateur recherche, ici le père de Maya D. Pour trouver les suggestions, la requête correspondant à la description est reformulée par rapport au

Règle	Triplet initial	Condition	Triplet relâché
Resource	$?r_1 = \mathbf{r}_2$		triplet vide
SuperProperty	$?r_1\ \mathbf{p}_1\ ?r_2$	p_1 rdfs:subPropertyOf p_2	$?r_1\ \mathbf{p}_2\ ?r_2$
Property	$?r_1\ \mathbf{p}_1\ ?r_2$	$\neg$(p_1 rdfs:subPropertyOf p_2)	triplet vide
SuperClass	$?r_1$ rdf:type $\mathbf{c}_1$	c_1 rdfs:subClassOf c_2	$?r_1$ rdf:type $\mathbf{c}_2$
Class	$?r_1$ rdf:type $\mathbf{c}_1$	$\neg$(c_1 rdfs:subClassOf c)	triplet vide

FIGURE 2: Règles élémentaires pour relâcher un triplet. Les variables sont précédées par un point d'interrogation. r_1 et r_2 sont des objets, p_1 et p_2 des propriétés et c_1 et c_2 des classes.

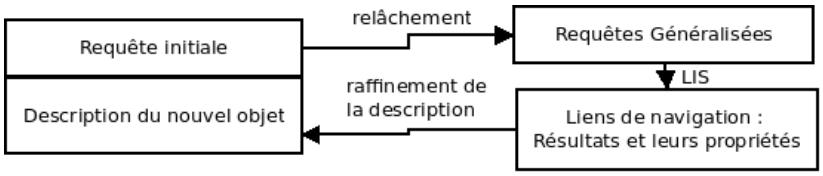

FIGURE 3: Raffinement interactif de la description du nouvel objet.

focus. Sans relâchement et avec reformulation, la nouvelle requête est *father of (a woman and name*: *Maya and mother*: *Zoé D.)*. Cette requête n'a pas de réponse [2]. Une des requêtes généralisées possible est : *father of (a person and name*: *? and mother*: *Zoé D.)*. Le résultat de cette requête relâchée est le père de la sœur du nouvel individu, et donc le père de Maya. Après sélection de cette suggestion, la description est *Maya D. is a woman and name*:*Maya and mother*: *Zoé D. and father*: *Guillaume D.* Cette nouvelle description est utilisée comme requête, pour calculer de nouvelles suggestions et ainsi de suite.

4.3. L'interaction avec l'utilisateur

Notre approche est implémentée dans Camelis 2 [3] (Ferré, 2010), une implémentation des systèmes d'information logiques pour le Web Sémantique. Il supporte la recherche exploratoire en guidant les utilisateurs de requête en requête, en suivant des liens de navigation. À chaque pas de navigation, l'ensemble des liens de navigation permet de raffiner la requête, en utilisant les résultats de celle-ci. Nous pouvons utiliser les mêmes mécanismes pour mettre à jour les descriptions, car le langage de description est un sous-ensemble du language de requête. Les liens de navigation permettent aux utilisateurs de compléter la description, en utilisant les résultats aux requêtes généralisées ou non. Chaque généralisation est associée à une distance, qui correspond au nombre de relâchements effectués. Ces résultats sont regroupés par distance. Par défaut, seuls ceux correspondant à la plus petite distance sont présentés, mais l'ensemble peut être élargi aux distances supérieures à la demande de l'utilisateur. La requête initiale peut être généralisée jusqu'à ce que tous les objets soient proposés comme suggestions. L'utilisateur peut à tout moment saisir ce qu'il souhaite ajouter. Cette saisie est soutenue par un mécanisme d'auto-complétion, similaire à QuiKey (Haller, 2010), pour per-

2. A moins que la mère du nouvel individu ait déjà une autre fille avec le même prénom !
3. `http://www.irisa.fr/LIS/ferre/camelis/camelis2.html`

mettre une réutilisation de l'existant. La possibilité de saisie permet à l'utilisateur de passer outre les suggestions s'il a déjà une idée très précise en tête.

5. L'algorithme de relâchement

Les structures de données utilisées pour l'implémentation sont le triplet, la requête, une conjonction de triplets et la distance, un entier positif. La requête initiale est une conjonction de triplets. Un relâchement d'une requête est la conjonction d'un relâchement de chacun de ses triplets. Pour calculer tous les relâchements d'une requête, l'implémentation utilise l'algorithme 1, la fonction relacherRequete appelée récursivement pour décomposer la requête en triplets qui sont relâchés avec la fonction relacherTriplet. Les requêtes relâchées sont ensuite reconstituées avec la fonction ConjonctionResultat.

La fonction relacherTriplet calcule l'ensemble des triplets relâchés et leur distance à partir du triplet initial. Chaque triplet est relâché par application d'une règle de la figure 2. Un triplet relaché peut encore être relâché pour obtenir un autre triplet. Après un ou plusieurs relâchements, chaque triplet devient le triplet vide : cela est équivalent à la suppression du triplet. Par exemple, l'appel de relacherTriplet avec les paramètres 0 pour la distance et ($?r_1$, mother, $?r_2$) pour le triplet va entraîner l'appel de relacherTriplet (1, ($?r_1$, parent, $?r_2$)) avec la règle *SuperProperty*. Cet appel va provoquer d'autres appels, jusqu'à obtenir le triplet vide. Le résultat sera l'ensemble {(0, ($?r_1$, mother, $?r_2$)), (1, ($?r_1$, parent, $?r_2$)), (2, triplet vide)}. La fonction ConjonctionResultat permet de recombiner des triplets et des requêtes relâchées par conjonction. À partir de deux ensembles de couples (distance, triplet) et (distance, requete), cette fonction retourne un seul ensemble. Les éléments composant ce nouvel ensemble correspondent à la conjonction deux à deux de tous les éléments des deux ensembles précédents. La distance associée à un élément est la somme des distances des deux éléments qui la composent.

La distance de relâchement est le nombre de relâchements appliqués sur les triplets de la requête initiale. Pour les suggestions, UTILIS ne conserve que les résultats des requêtes qui forment un graphe connexe, contenant le focus et avec un résultat non vide. Avoir un graphe connexe est important pour prendre en compte l'ensemble de la requête. La présence du focus est importante pour faire des suggestions adéquates. Les résultats des requêtes généralisées sont calculés par les LIS. Seuls les résultats non vides sont conservés, sinon aucune suggestion ne serait possible. La requête la plus générale _?_ est toujours accessible à une distance finie, donnant accès par suggestion à tous les ob-

Algorithm 1 relacherRequete(Requete requete)

Require: Prend en paramètre une requête : Requete = List (Triplet)
Ensure: Retourne l'ensemble des paires (distance, requeteRelachee)

```
1: if query.isEmpty then
2:     return {(0, List())}
3: else
4:     st = relacherTriplet(0, query.head)
5:     sq = relacherRequete(query.tail)
6:     return ConjonctionResultat(st, sq)
7: end if
```

jets. Pour réutiliser l'existant, nous avons créé des fonctions pour convertir un ensemble de triplets en requête LISQL et inversement.

6. Un exemple d'application

Pour illustrer UTILIS, cette section détaille la création de la description d'une nouvelle personne dans une base de généalogie (figure 4), réalisée avec notre prototype. Par exemple, Patrick D. est un homme prénommé Patrick, né à Metz en 1970, marié à Mirabelle D. et le père de Jeanne D. et Jade D.

Imaginons qu'un utilisateur souhaite ajouter Maya D, une femme, prénommée Maya, née à Metz en 2010 et fille de Guillaume D. et Zoé D. Les différentes étapes de la création sont présentées dans la figure 5. À chaque étape, la première boîte contient la description courante de l'objet, la partie soulignée représente le focus. Si aucune partie de la requête n'est soulignée, le focus est sur l'ensemble de la requête. La seconde boîte contient les suggestions avec devant chacune d'elles la distance minimale entre la description et la requête généralisée ayant permis cette réponse. Les suggestions proposées sont du type du focus, soit des propriétés soit des ressources. Pour chaque étape, dans notre exemple, cinq éléments sont proposés. Toutefois, pour des questions de place, nous avons parfois réduit ce nombre. L'élément en gras correspond au choix de l'utilisateur. À l'étape 1, UTILIS n'a aucune information sur le nouvel individu, les réponses sont toutes les propriétés de la base. L'utilisateur sélectionne une réponse, ici *a woman*, pour compléter la description. À l'étape 2, les résultats s'adaptent à la description courante : les propriétés proposées sont celles que possède au moins une femme. Jusqu'à l'étape 3, l'utilisateur choisit un des résultats de la requête initiale. À l'étape 4,

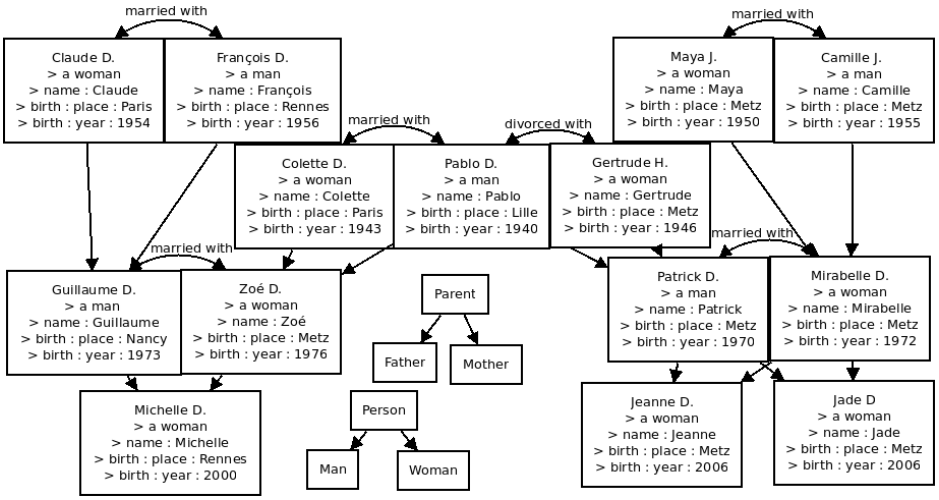

FIGURE 4: Répresentation graphique de la base généalogique

l'élément que souhaite ajouter l'utilisateur n'est pas dans les réponses de la requête initiale. L'utilisateur peut généraliser la requête et utiliser les réponses pour compléter la description. À partir de l'étape 5, la requête initiale n'a aucun résultat. La description courante ne correspond à aucun autre objet. Les réponses des plus petites généralisations servent de suggestions. À l'étape 7, Guillaume D. est proposé en premier comme père du nouvel individu, car Guillaume D. et Zoé D. sont déjà parents de Michelle D. À l'étape 10, la première proposition correspond au lieu de naissance de Michelle D, qui n'est pas le lieu de naissance de Maya D. L'utilisateur regarde les résultats avec une plus grande distance pour trouver le lieu de naissance de Maya D. À l'étape 12, pour l'année de naissance de Maya D, l'utilisateur ne cherche pas l'année dans les suggestions d'UTILIS mais la saisit. À l'étape 13, l'utilisateur décide que la description de Maya D. est complète et l'ajoute à la base.

Notre approche UTILIS permet avec un graphe RDF(S) d'avoir un guidage fin en utilisant la description partielle du nouvel objet et les objets présents dans la base.

7. Comparaison avec l'état de l'art

Dans cette section, nous comparons notre approche avec l'état de l'art. Les éditeurs existants de base de connaissances utilisent différents types de connaissance pour aider les utilisateurs à ajouter de l'information.

Étape 1	Étape 2	Étape 3	Étape 4
?	**a woman**	**a woman and name** : **?**	**a woman and name**:**Maya**
0 a person 0 **a woman** 0 a man 0 mother of ? 0 father : ?	0 birth : ? 0 married with ? 0 **name**: **?** 0 mother : ? 0 mother of ?	0 Claude 0 Colette 0 Jade 0 **Maya**	0 birth : ? 0 married with ? 0 mother of ? 1 **mother** : **?**
Étape 5	Étape 6	Étape 7	Étape 8
a woman and name:**Maya and mother** : **?**	**a woman and name**:**Maya and mother**:**ZoéD**	**a woman and name**:**Maya and mother**: **ZoéD and father**: **?**	**a woman and name**:**Maya and mother**: **ZoéD and father**: **GuillaumeD**
1 MayaJ 1 ColetteD 1 MirabelleD 1 **ZoéD** 2 ClaudeD	1 birth : ? 1 **father** : **?** 2 married with ? 2 mother of ? 3 father of : ?	1 **GuillaumeD** 2 PabloD 2 PatrickD 2 CamilleJ	1 **birth** : **?** 3 mother of ? 3 married with ?
Étape 9	Étape 10	Étape 11	Étape 12
a woman and name:**Maya and mother**:**ZoéD and father**: **GuillaumeD and birth**: **?**	**a woman and name**:**Maya and mother**:**ZoéD and father**: **GuillaumeD and birth**: **place**: **?**	**a woman and name**:**Maya and mother**:**ZoéD and father**: **GuillaumeD and birth**: **place**:**Metz**	**a woman and name**:**Maya and mother**:**ZoéD and father**: **GuillaumeD and birth**: **(place**:**Metz and year**: **?)**
1 **place** : **?** 1 year : ? 1 birth of ?	1 Rennes 3 **Metz**	2 **year** : **?** 2 birth of ?	2 : 2000

Étape 13

a woman and name:**Maya and mother**:**Zoé D and father**: **Guillaume D and birth**:**(place**:**Metz and year**:**2010)**
4 : mother of . . .

FIGURE 5: Étapes de création du nouvel individu Maya D., dont la mère est Zoé D., le père Guillaume D. et née à Metz en 2010.

L'éditeur Protégé [4] utilise le vocabulaire et les axiomes de l'ontologie. À

4. `http://protege.stanford.edu`

l'étape 2 de notre scénario, toutes les propriétés seraient proposées, car aucune connaissance sur les domaines et co-domaine n'est connue dans notre base. À l'étape 7, tous les objets seraient proposés par ordre alphabétique.

L'éditeur Gino (Bernstein & Kaufmann, 2006) utilise un guidage s'appuyant sur la syntaxe et le vocabulaire, pour la création de triplets RDF(S). À l'étape 2, l'utilisateur devrait saisir des caractères pour que Gino lui propose des propriétés. À l'étape 7, le même mécanisme s'appliquerait avec les objets.

Le système QuiKey (Haller, 2010) utilise un guidage s'appuyant également sur la syntaxe et le vocabulaire, pour la création de triplets RDF(S). À l'étape 2, toutes les propriétés connues seraient proposées, avec réduction de cette liste par saisie. A l'étape 7, le même mécanisme s'appliquerait avec les objets.

Le wiki sémantique ACEWiki (Kuhn, 2009) propose un éditeur de triplets RDF(S), en utilisant des listes de mots-clés et de ressources, filtrées par saisie, s'appuyant sur la syntaxe et le vocabulaire du graphe. À l'étape 2, la liste des propriétés serait proposée. À l'étape 7, la liste des objets serait proposée.

Le wiki sémantique KiWI (Knowledge In WIki), anciennement IkeWiki (Schaffert, 2006) propose un éditeur de contenu dans lequel des annotations sémantiques peuvent être insérées et un éditeur d'annotations sémantiques. À l'étape 2, l'utilisateur aurait une liste déroulante de propriétés. À l'étape 7, il devrait saisir des caractères pour obtenir une suggestion.

Semantic MediaWiki (Völkel *et al.*, 2006) et son extension Semantic Form [5] sont des extensions de MediaWiki, le moteur de Wikipédia. Elles proposent de créer manuellement des formulaires pour aider les utilisateurs à créer des pages wikis avec du contenu sémantique. À l'étape 2, toutes les propriétés sélectionnées par le créateur du formulaire seraient proposées. À l'étape 7, l'utilisateur devrait saisir des caractères pour avoir des suggestions.

Avec UTILIS, l'utilisateur n'a pas besoin de saisir l'élément à ajouter et une liste réduite de propositions lui est faite. Pour certains éléments comme les dates ou les identifiants, la saisie est moins coûteuse pour l'utilisateur que de les chercher dans la liste de suggestions. Même si des suggestions sont toujours proposées, l'utilisateur peut à tout moment saisir l'élément à ajouter. Un mécanisme d'auto-complétion, similaire à QuiKey, est mis en place pour la réutilisation de l'existant. La saisie permet de ne pas avoir à chercher dans les suggestions si l'élément à ajouter n'est pas en début de liste et que l'utilisateur connaît déjà l'élément. Il est probable que pour certains objets, très différents de tous ceux présents, les suggestions ne correspondront pas à

5. `http://www.mediawiki.org/wiki/Extension:Semantic_Forms`

l'ajout de l'utilisateur. Nous souhaitons faire une évaluation avec les utilisateurs pour vérifier que les suggestions sont un avantage pour eux, quelque soit le type d'objet à ajouter. Nous pensons faire évoluer les règles pour prendre en compte d'autres éléments si un langage plus riche que RDF(S) est utilisé. Pour le passage à échelle, l'algorithme de relâchement va être optimisé à partir de travaux sur la programmation dynamique (Ilyas *et al.*, 2004).

8. Conclusion

Ce papier présente UTILIS, une méthode pour guider les utilisateurs lors de la création et la mise à jour d'objets dans un graphe RDF(S). Le guidage tire profit des faits existants et de la description courante du nouvel objet. UTILIS est incrémental. Dès qu'un élément est ajouté à la description, il est utilisé pour calculer les suggestions servant au guidage. Pour calculer les suggestions, la description est utilisée comme requête. Toutefois, la description de l'objet devenant rapidement unique, la requête initiale n'a plus de réponses. Cette requête est alors généralisée à l'aide de règles de relâchement. Ces généralisations permettent de proposer en suggestions des objets similaires au nouvel objet. UTILIS est interactif, l'utilisateur est placé au centre du processus de mise à jour. Les utilisateurs n'ont plus besoin de saisir et de connaître les ressources existantes. Ils ont moins de risques d'oublier de remplir les propriétés importantes. En plus de ce confort, notre méthode garantit que le vocabulaire utilisé dans toute la base est cohérent. La création des premières ressources est la phase la plus coûteuse pour l'utilisateur car il faut inventer les classes et les propriétés. Toutefois, cela est vrai pour toutes les approches.

Références

BERNSTEIN A. & KAUFMANN E. (2006). Gino - a guided input natural language ontology editor. In *Proceedings of the International Semantic Web Conference (ISWC)* : Springer.

D'AQUIN M., BALDASSARRE C., GRIDINOC L., ANGELETOU S., SABOU M. & MOTTA E. (2007). Characterizing knowledge on the semantic web with watson. In *Evaluation of Ontologies and Ontology-based tools Workshop*, volume 329, p. 1–10.

FERRÉ S. (2010). Conceptual navigation in RDF graphs with SPARQL-like queries. In *International Conference on Formal Concept Analysis (ICFCA)*, volume 5986 of *Lecture Notes in Computer Science*, p. 193–208 : Springer.

FERRÉ S., HERMANN A. & DUCASSÉ M. (2011). *Semantic Faceted Search : Safe and Expressive Navigation in RDF Graphs*. Rapport interne, IRISA - PI1964. http ://hal.inria.fr/inria-00554093/PDF/PI-1964.pdf.

FERRÉ S. & RIDOUX O. (2004). An introduction to logical information systems. *Information Processing & Management*, **40**(3), 383–419.

GANTER B. & WILLE R. (1999). *Formal concept analysis mathematic foundations*. Berlin-Heidelberg : Springer.

GIACOMO G. D., LENZERINI M., POGGI A. & ROSATI R. (2006). On the update of description logic ontologies at the instance level. In *National Conference on Artificial Intelligence and the Innovative Applications of Artificial Intelligence Conference* : AAAI Press.

HALLER H. (2010). QuiKey – an efficient semantic command line. In *Knowledge Engineering and Management by the Masses*, p. 473–482 : Springer.

HITZLER P., KRÖTZSCH M. & RUDOLPH S. (2009). *Foundations of Semantic Web Technologies*. CRC Press.

HURTADO C. A., POULOVASSILIS A. & WOOD P. T. (2008). Query relaxation in RDF. *J. Data Semantics*, **10**, 31–61.

ILYAS I. F., AREF W. G. & ELMAGARMID A. K. (2004). Supporting top-k join queries in relational databases. *J. Very Large Databases*, **13**, 207–221.

KUHN T. (2009). How controlled english can improve semantic wikis. In *Semantic Wiki Workshop at the European Semantic Web Conference (ESWC)*, volume 464 : CEUR-WS.org.

LIMPENS F., GANDON F. & BUFFA M. (2009). Sémantique des folksonomies : structuration collaborative et assistée. In *Ingénierie des Connaissances (IC)*, p. 37–48 : Presses Universitaires de Grenoble.

PASSANT A. & LAUBLET P. (2008). Meaning of a tag : A collaborative approach to bridge the gap between tagging and linked data. In *Workshop Linked Data on the Web (LDOW)* : CEUR-WS.

PRUD'HOMMEAUX E. & SEABORNE A. (2008). *SPARQL Query Language for RDF*. Rapport interne. World Wide Web Consortium recommendation.

G. SACCO & Y. TZITZIKAS, Eds. (2009). *Dynamic Taxonomies and Faceted Search : Theory, Practice, and Experience*, volume 25 of *The Information Retrieval Series*. Berlin : Springer.

SCHAFFERT S. (2006). IkeWiki : A semantic wiki for collaborative knowledge management. In *Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises*, p. 388–396 : IEEE Computer Society.

VÖLKEL M., KRÖTZSCH M., VRANDECIC D., HALLER H. & STUDER R. (2006). Semantic Wikipedia. In *International conference on World Wide Web (WWW)*, p. 585–594 : ACM Press.