Une plate-forme XML pour représenter des documents et leur contenu pour la mise en œuvre du Web sémantique

Rim Al-Hulou*, Olivier Corby†, Rose Dieng-Kuntz, Jérôme Euzenat‡, Carolina Medina Ramírez, Amedeo Napoli, Raphaël Troncy§

Résumé

Les documents accessibles à partir du Web ou d'une base documentaire quelconque constituent une source considérable de connaissances. Cet article présente le projet ESCRIRE, dont un des objectifs est de construire une plate-forme XML associée à un système de représentation des connaissances, pour annoter et consulter des documents, en tirant parti d'une ontologie du domaine construite à partir du contenu des documents. L'ensemble de ces fonctionnalités constitue l'un des préalables à la mise en œuvre du Web sémantique, et les problèmes liés à cette mise en œuvre sont discutés.

1 Introduction

Pour manipuler efficacement un ensemble de documents provenant du Web, que ce soit pour une recherche d'information, une exploration de données, ou une extraction d'information, il est intéressant de construire une représentation explicite du contenu des documents, et en conséquence, de les indexer et de les annoter. L'objectif du travail de recherche présenté dans cet article — dans le cadre du projet de recherche appelé ESCRIRE — est de proposer une méthode originale de construction d'annotations et de requêtes exploitant le contenu d'un ensemble de documents, ainsi qu'une base de connaissances relatives au domaine sur lequel portent les documents. Contrairement aux techniques d'indexation classiques, comme la recherche plein-texte, par mots clés, pondérée, ou par thésaurus, les annotations sur les documents mettent ici en jeu des éléments conceptuels présents dans les documents eux-mêmes. Ces éléments conceptuels sont en relation avec des éléments conceptuels associés au domaine sur lequel portent les documents, et qui sont représentés dans une ontologie. Une telle approche permet d'envisager une *recherche de documents dirigée par des connaissances* dans la cadre du Web sémantique [4, 10]. De plus, le contenu des documents est représenté en parallèle dans un formalisme de représentation des connaissances, et il est attaché aux documents eux-mêmes. Cela permet alors de mettre en œuvre des raisonnements sur les documents, comme par exemple des classifications de documents ou des recherches par analogie sur des éléments contenus dans les documents.

Les formalismes de représentation des connaissances sont de bons candidats pour représenter le contenu de documents, et plusieurs formalismes de RC peuvent être envisagés, sans

*LORIA, 615, rue du Jardin Botanique, 54610 Villers-lès-Nancy, {Rim.Al-Hulou, Amedeo.Napoli}@loria.fr

†INRIA Sophia Antipolis, 2004 route des Lucioles, 06902 Sophia Antipolis, {Olivier.Corby, Rose.Dieng, Carolina.Medina}@sophia.inria.fr

‡INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 Montbonnot St-Martin, Jerome.Euzenat@inrialpes.fr

§INRIA Rhône-Alpes et INA, 4, avenue de l'Europe, 94366 Bry sur Marne cedex, Raphael.Troncy@ina.fr

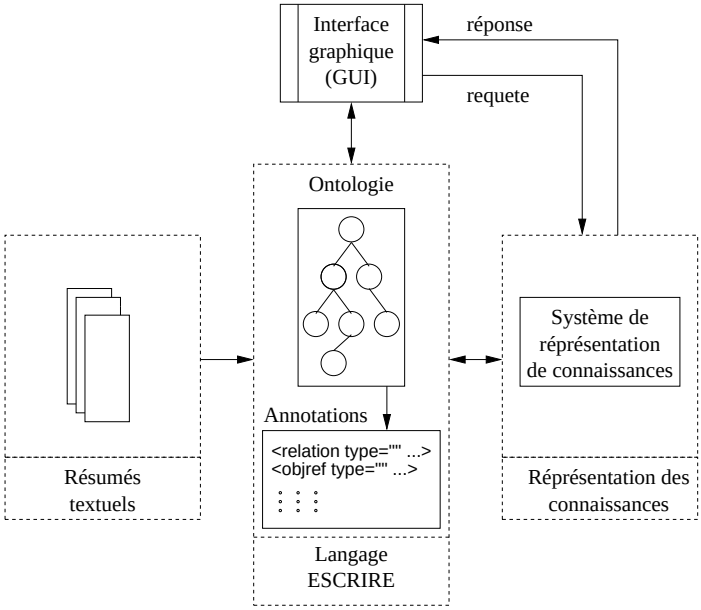

FIG. 1 – *Des documents aux systèmes de représentation : le cadre des expérimentations du projet* ESCRIRE.

qu'il n'existe de guide pratique permettant de choisir un formalisme plutôt qu'un autre. C'est un des autres objectifs du projet ESCRIRE que de mettre en œuvre une comparaison de trois formalismes de représentation des connaissances pour l'annotation et l'interrogation du contenu de documents, qui sont en l'occurrence : les graphes conceptuels (GC), les logiques de descriptions (LD), et les représentations par objets (RCO) [1]. Le protocole de comparaison repose sur un langage pivot s'appuyant sur XML— appelé le *langage pivot* ESCRIRE dans la suite — qui sert de *passerelle* entre les documents et les formalismes de représentation. La figure 1 montre le protocole de comparaison utilisé dans le projet ESCRIRE. Tout d'abord, l'ontologie, les annotations, mais aussi les requêtes, sont codées dans le langage pivot ESCRIRE, et ce code est ensuite traduit dans chaque formalisme de représentation. Les requêtes sont ensuite traitées dans chaque formalisme de RC. Les expérimentations ont été réalisées sur une base de résumés d'articles du domaine biologique extraits de la base publique NIH Medline[1]. Pour ces expérimentations, une centaine de résumés a été annotée manuellement, et le contenu extrait de ces résumés a été restreint à une liste de gènes et d'interactions entre ces gènes.

Une des ambitions du projet ESCRIRE est proposer une méthodologie générale pour représenter le contenu de documents en utilisant un langage pivot comme celui d'ESCRIRE. L'utilisation du tel langage pivot peut être comparée à des techniques d'intégration de bases de données reposant sur un médiateur et un formalisme de RC, où plusieurs bases de données peuvent être interrogées au moyen d'une interface commune [5]. La méthodologie présentée ici peut être généralisée à d'autres tâches telles que l'extraction d'information, l'analyse de textes, la fouille de données et le Web sémantique [10, 11]. Nous n'aborderons pas ici la comparaison des formalismes de représentation, mais seulement l'utilisation du langage pivot ESCRIRE et la justification des choix qui ont été faits.

1. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed

2 Le langage pivot ESCRIRE

Dans cet article, un *document* dénote le résumé d'un article en biologie présent dans une entrée Medline, où ont été supprimés le titre, le nom des auteurs et du journal. Un *contenu* correspond aux éléments jugés significatifs du document à représenter. Une *annotation* décrit ces éléments biologiques significatifs localisés dans le document, par exemple ici des gènes et des interactions entre gènes. La connaissance du domaine repose sur une *ontologie* qui est organisée en une taxonomie de classes représentant les concepts du domaine et un ensemble d'*assertions* ou faits impliquant les classes et les instances de ces classes.

Ainsi, dans le langage pivot ESCRIRE, un document est représenté par une structure XML composée du document lui-même et d'une liste d'annotations, elles-mêmes représentées par des structures XML. L'ontologie et les assertions sont représentées toutes les deux dans le langage pivot ESCRIRE et dans chaque formalisme de représentation de connaissances.

2.1 L'ontologie

L'ontologie repose sur une DTD[2] qui définit des classes, des classes de relations, et une relation de subsomption pour organiser ces classes en une hiérarchie. Une classe est décrite par un ensemble d'*attributs* – *rôles* pour les classes de relations – dont le co-domaine peut être une classe ou un type concret comme entier, chaîne de caractères, ou booléen. Une classe peut être *primitive* ou *définie* : les attributs de la classe sont considérés comme des conditions nécessaires — respectivement nécessaires et suffisantes — pour l'appartenance d'un individu à cette classe. La distinction primitive/ définie est la même qu'en logique de descriptions [7].

La figure 2 montre la déclaration de la classe primitive `gene` : la balise `descclass` introduit le nom de la classe primitive `gene` alors que la balise `defattribute` introduit l'attribut de nom `name`, dont le co-domaine (`string`) est donné par la balise `typeref`.

2.2 Les annotations

Dans un document, une annotation décrit des gènes et des relations entre gènes. Une annotation est représentée par une structure XML ayant deux types de balises principaux : des *rôles* et des *attributs*. Les rôles sont utilisés pour dénoter le domaine et le co-domaine des relations, alors que les attributs sont utilisés pour représenter les caractéristiques des relations.

Par exemple, l'annotation `interaction` donnée à la figure 2 décrit une interaction entre un gène source instigateur de type `gene` et appelé `mam`, avec un gène cible de type `gene` et appelé `N`. Cette interaction a un attribut `effect`, dont la valeur est `inhibition`. L'interprétation de cette annotation est la suivante : le gène `mam` inhibe le gène `N` dans ce type particulier d'interaction. Les individus présents dans une annotation correspondent à des instances de gènes existant dans l'ontologie.

2.3 Les requêtes

Une requête dans le langage pivot ESCRIRE s'appuie sur le triplet `SELECT-FROM-WHERE`. Les opérateurs logiques de conjonction, disjonction, et négation ainsi que les quantificateurs universel et existentiel peuvent être utilisés. Dans la requête de la figure 2, l'objectif est de rechercher des interactions (entre gènes) dont la cible ou l'instigateur sont le gène de type

2. `http://escrire.inrialpes.fr/dtd/escrire.dtd`

```
<descclass name='gene'>
   <defattribute name='name'>
      <typeref name='string'>
   </defattribute
</descclass>

<relation type='interaction'>
   <role name='promoter'>
      <objref type='gene' id='mam'>
   </role>
   <role name='target'>
      <objref type='gene' id='N'>
   </role>
   <attribute name='effect'>
      <value>inhibition</value>
   </attribute>
</relation>

SELECT I.effect, I.location
FROM I:interaction
WHERE I.target = OBJREF(gene,'gt')
      OR I.promoter = OBJREF(gene,'gt')
```

FIG. 2 – *Un exemple de classe primitive, de relation et de requête en langage pivot* ESCRIRE.

`gene` appelé `gt`, et de retourner la valeur des attributs `effect` et `location` des interactions composant le résultat.

En général, une requête fait référence à une interaction entre gènes ou à un gène en particulier, et elle est utilisée pour retourner les documents mentionnant cette interaction ou ce gène particulier. Dans notre cadre, une requête `Q` est interprétée dans le contexte d'un document particulier `D`. Le document `D` est une réponse à la requête `Q` si `D` satisfait `Q` en utilisant l'ontologie et les annotations associées au document `D`. Le résultat correspondant à une requête est composé par les `URL` des documents satisfaisant la requête.

3 Discussion et remarques

Remarques générales.

L'expérience ESCRIRE a mis en évidence l'intérêt d'exploiter, voire de combiner, plusieurs formalismes de RC avec XML dans des applications qui nécessitent de manipuler des documents sur la base de leur contenu, et pour le Web sémantique plus généralement, en confortant ainsi les discussions amorcées dans [2, 8]. Pour de telles tâches, il existe actuellement un noyau de constructeurs — correspondant à la base des constructions de RDF(S) — inclus dans la plupart des langages disponibles et dédiés au Web sémantique, comme DAML+OIL par exemple [9, 10] (voir aussi `http://www.daml.org/2001/03/daml+oil-index.html`). Ce noyau doit être étendu aussi bien pour prendre en compte la manipulation de documents par le contenu à grande échelle que la gestion du Web sémantique [12, 13].

Bien que brièvement, nous avons esquissé une stratégie générale pour implémenter une correspondance entre un langage pivot en XML et un formalisme de RC concret. Les stratégies

locales s'appuyant sur la construction de traducteurs spécifiques ne sont pas faciles à généraliser, et elles mènent à des problèmes lors de maintenance et du développement d'applications d'envergure. D'une certaine façon, l'infrastructure globale de la plate-forme ESCRIRE peut se voir comme un « gestionnaire de requêtes pour le Web sémantique », où les requêtes sur les documents peuvent être satisfaites simultanément et indépendamment par plusieurs moteurs, ce qui est très utile lorsque le pouvoir d'expression et l'efficacité des formalismes de RC sont différents, comme c'est le cas d'ailleurs.

Parmi les perspectives actuelles du travail de recherche sur la plate-forme ESCRIRE se pose la question importante suivante : « vaut-il mieux avoir un langage de représentation très expressif et un langage de requêtes très simple, ou bien l'inverse, ou bien les deux ? », sachant les problèmes de complexité qui sont posés par la richesse de l'ensemble des constructeurs d'un langage de représentation et d'un langage de requêtes [6, 3, 4]. De plus, nous étudions une question parallèle qui se pose dans le cadre de la combinaison de bases de connaissances et de bases de données pour la manipulation de documents, et plus généralement pour le Web sémantique : « quelles sont les tâches respectives à la charge du formalisme de RC et à la charge du gestionnaire de la base de données pour la satisfaction de requêtes ? », question qui n'a pas encore eu de réponse satisfaisante pour le moment.

Quelques problèmes de représentation.

La distinction entre *classe primitive* et *classe définie* n'est pas toujours disponible dans les formalismes de RC, bien que nécessaire : elle existe par exemple dans les GC et les LD. En particulier, cette distinction n'est pas disponible dans le modèle RDF(S) et donc une extension du modèle s'impose de ce point de vue.

Dans le projet ESCRIRE, l'hypothèse du monde clos est en vigueur : l'information disponible est considérée comme complète. Ce qui ne vas pas sans poser des problèmes, en particulier pour rechercher un ensemble d'individus *ne vérifiant pas* une contrainte donnée (qui revient en fait à rechercher le complémentaire de l'extension d'un concept donné). Il est alors nécessaire d'implémenter ou d'adapter une construction *ad hoc* dans le formalisme de RC.

Dans les résumés des articles de Medline utilisés pour l'expérimentation, une classe de gènes est interprétée comme une classe d'individus. Mais la question se pose : « doit-on représenter une classe de gènes comme un individu pour pouvoir la manipuler dans les annotations et/ou bien comme une classe dans l'ontologie ? » De plus, quel doit être le statut d'une relation entre gènes : une relation instance/classe, sous-classe/classe, ou encore classe/méta-classe ? Pour le projet ESCRIRE, la notion de gène et les relations entre les gènes ont toutes été réifiées, et ce choix de réification n'a pas posé de problème spécifique dans aucun des trois formalismes de RC. Les relations binaires et n-aires ont été représentées par des classes, et les propriétés de transitivité et de symétrie ont été traitées par du code associé : des mécanismes de raisonnement spécifiques pour contrôler la propagation des valeurs des attributs par transitivité doivent être pris en compte. Le modèle RDF(S) quant à lui doit être étendu de ce point de vue encore pour prendre en compte la réflexivité, la symétrie, la transitivité et les relations inverses, et le partage de propriétés associé.

Le langage pivot ESCRIRE permet de construire des requêtes où sont référencées des classes, mais il ne fournit aucune construction spécifique pour mener à bien un raisonnement sur les classes invoquées dans les requêtes, comme par exemple : rechercher les classes dont un individu donné `x` est une instance, ou bien les classes subsumant un autre classe `C` donnée. La gestion de telles requêtes et des raisonnements associés doit être laissée à la charge du du formalisme de RC.

4 Conclusion

Dans cet article, nous avons présenté une méthode originale pour annoter et consulter des documents sur la base de leur contenu. Un langage pivot a été conçu en XML qui constitue une passerelle entre les documents et un formalisme de RC. Une ontologie du domaine, représentée dans le langage pivot et dans le formalisme de RC, est exploitée aussi bien dans l'annotation que dans l'interrogation des documents. À moyen terme, cette expérimentation va nous servir à comparer de façon plus approfondie trois formalismes de RC— les logiques de descriptions, les graphes conceptuels, et les systèmes de représentation des connaissances par objets — du point de vue de la gestion des documents — annotation, consultation, interrogation — et plus généralement pour la mise en œuvre de la technologie du Web sémantique.

Références

[1] R. Al-Hulou, O. Corby, R. Dieng-Kuntz, J. Euzenat, C. Ramirez, A. Napoli, and R. Troncy. Three knowledge representation formalisms for content-based manipulation of documents. In M. Cristani, editor, *Workshop on Semantic Web SemWeb@KR-02, Toulouse*, 2002. CD Rom publication.

[2] R. Al Hulou, A. Napoli, and E. Nauer. XML : un formalisme de représentation intermédiaire entre données semi-structurées et représentations par objets. In C. Dony and H.A. Sahraoui, editors, *Langages et Modèles à Objets (LMO'00), Montréal*, pages 75–90. Hermès, Paris, 2000.

[3] D. Calvanese, G. De Giacomo, and M. Lenzerini. Queries and Constraints on Semi-structured Data. In *Advanced Information Systems Engineering, (CAiSE-99), Heildelberg, Germany*, Lecture Notes in Computer Sciences 1626, pages 434–438. Springer, Berlin, 1999.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. Representing and reasoning on XML documents: A description logic approach. *Journal of Logic and Computation*, 9(3):295–318, 1999.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logics Framework for Information Integration. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy*, pages 2–13, 1998.

[6] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

[7] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford, CA, 1996.

[8] J. Euzenat. XML est-il le langage de représentation de connaissance de l'an 2000 ? In C. Dony and H.A. Sahraoui, editors, *Langages et Modèles à Objets (LMO'00), Montréal*, pages 59–74. Hermès, Paris, 2000.

[9] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In R. Dieng, editor, *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, number 1937 in Lecture Notes in Artificial Intelligence, pages 1–16. Springer-Verlag, 2000.

[10] D. Fensel, F. van Harmelen, , I. Horrocks, D.L. McGuinness, and P.F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, March/April 2001.

[11] S.A. McIlraith, T.C. Son, and H. Zeng. Semantic Web issues. *IEEE Inteligent Systems*, 16(2):46–53, March/April 2001.

[12] P.F. Patel-Schneider and D. Fensel. Layering the Semantic Web: Problems and Directions. In I. Horrocks and J. Hendler, editors, *The Semantic Web — ISWC 2002, LNCS 2342*, pages 16–29, Berlin, 2002. Springer.

[13] P.F. Patel-Schneider and J. Siméon. Building the Semantic Web on XML. In I. Horrocks and J. Hendler, editors, *The Semantic Web — ISWC 2002, LNCS 2342*, pages 147–161, Berlin, 2002. Springer.