Du texte à la connaissance : annotation sémantique et peuplement d'ontologie appliqués à des artefacts logiciels

Florence Amardeilh[1], Danica Damljanovic[2]

[1] Mondeca. 3, cité Nollez, 75018 Paris, France
florence.amardeilh@mondeca.com
[2] Department of Computer Science. University of Sheffield,
Regent Court, 211 Portobello St, Sheffield S1 4DP, UK
d.damljanovic@dcs.shef.ac.uk

Résumé : Les applications logicielles possèdent généralement une courbe d'apprentissage considérable pour les nouveaux développeurs et pour ceux qui souhaitent en intégrer des parties dans leurs propres applications. L'attrait d'utiliser ici une technologie à base de sémantique repose sur son potentiel à associer un réseau de connaissance aux artefacts logiciels existants, structurés ou non. Ceci se traduit notamment par deux étapes clefs, l'annotation sémantique et le peuplement d'ontologie, qui restent d'importants challenges à résoudre. Nous présentons ici le Content Augmentation Manager, une plateforme servant de médiateur entre les outils d'annotation et les référentiels sémantiques. L'annotation des artefacts logiciels est réalisée par l'outil d'extraction d'information KCIT, capable de produire automatiquement des annotations sur la base d'une ontologie donnée.

Mots-clés : annotation sémantique, acquisition de connaissances, peuplement d'ontologie, artefacts logiciels, référentiels sémantiques, ontologies.

1 Introduction

Réussir la réutilisation de code et éviter les anomalies dans l'ingénierie logicielle requiert de nombreuses qualités, à la fois à la source du code et dans l'équipe de développement. Parmi ces qualités se trouvent la facilité d'identification des composants pertinents et la facilité de compréhension de leurs paramètres et de leurs profils d'usage. L'attrait à utiliser une technologie sémantique pour adresser ce problème repose sur son potentiel à associer un réseau sémantique de connaissance à la documentation logicielle. Cette documentation logicielle provient de multiples artefacts logiciels, comprenant aussi bien des données non structurées comme les articles des forums, les spécifications et les divers manuels que des données structurées issues du code source et des fichiers de configuration. La documentation enrichie peut alors être exploitée afin d'ajouter de nouvelles fonctionnalités, fournir au développeur de nouvelles façons de localiser et d'intégrer les composants, etc.

Deux étapes primordiales de cet enrichissement sont l'annotation sémantique et le peuplement d'ontologie. Dans ce contexte, nous définissons l'**annotation sémantique**

comme une représentation formelle d'un contenu, exprimée à l'aide de concepts, relations et instances décrits dans une ontologie, et liée à la ressource originelle. Les instances sont généralement stockées dans une base de connaissance, indépendamment de la ressource annotée. Elles peuvent donc être réutilisées pour annoter d'autres ressources, offrant par là un point de vue centralisé et consensuel de la connaissance du domaine. L'action d'ajouter des instances à une base de connaissance est appelée **peuplement d'ontologie**. Notons que la notion de peuplement d'ontologie doit être clairement distinguée de celle d'enrichissement d'ontologie. Dans ce cas, il s'agit plutôt d'ajouter de nouveaux concepts et relations au modèle formel de l'ontologie.

Bien que faisant déjà l'objet de nombreuses recherches, ces tâches d'annotation sémantique et de peuplement d'ontologie restent un véritable challenge, pas seulement dans celui de l'ingénierie logicielle mais quelque soit le domaine étudié. En effet, le rôle des humains demeure bien souvent irremplaçable et l'automatisation reste un des plus grands besoins pour de tels outils, particulièrement lorsqu'il s'agit d'annoter de grandes collections de documents (Uren et al, 2006).

Dans ce papier nous présentons notre solution, en mettant l'accent sur deux objectifs : premièrement extraire automatiquement des informations relatives aux artefacts logiciels par rapport à une ontologie de domaine, et deuxièmement résoudre le problème du peuplement automatique de l'ontologie et de l'annotation sémantique des artefacts à partir de ces extractions. Cette solution, appelée Content Augmentation Manager (CA Manager), a pour objectif d'aider à combler le fossé existant entre les outils d'extraction d'information et les référentiels sémantiques qui sont utilisés pour stocker la connaissance qui a été collectée. L'outil d'extraction d'information utilisé ici est le Key Concept Identification Tool (KCIT), capable de produire automatiquement des annotations sémantiques à partir d'artefacts logiciels sans configuration majeure. Son seul paramètre d'entrée est l'ontologie du domaine.

Dans la Section 2 nous discutons des travaux relatifs à notre problématique. Puis nous présentons KCIT dans la Section 3 et le CA Manager dans la Section 4. Les résultats de nos évaluations sont détaillés dans la Section 5. Finalement, nous concluons et envisageons nos futurs travaux dans la Section 6.

2 L'annotation sémantique en pratique

Un outil d'annotation sémantique permet de créer et de gérer un ensemble d'annotations sémantiques à partir d'un document donné. Leur objectif consiste à alléger le fardeau de l'annotation manuelle quelque soit la ressource concernée, et surtout lorsqu'il s'agit de grands volumes de données. La plupart de ces outils ont évolué vers des d'environnements de plus en plus automatisés grâce aux méthodes issues des champs de l'Extraction d'Information et de l'Apprentissage Automatique (Corcho, 2006). Ils peuvent aussi être utilisés pour peupler une ontologie, comme Melita (Ciravegna, 2002), les deux tâches convergeant vers les mêmes types d'outils considérés comme un moyen de capturer la connaissance d'un domaine.

KIM (Kiryakov et al., 2005) s'appuie sur son ontologie générale PROTON[1] pour annoter des pages Web en identifiant automatiquement, à l'aide de dictionnaires et de patrons d'extraction définis dans GATE, les phrases clefs et les entités nommées (personnes, organisations, lieux, etc.). Puis KIM est capable de les relier à l'URI d'une instance particulière dans l'ontologie. Les annotations sont ensuite exploitées pour l'indexation et la recherche sémantique, la co-occurrence et l'analyse des tendances de popularité.

A l'Université de Technologie d'Helsinki en Finlande, des chercheurs ont développé un cadre pour l'annotation automatique (Vehvilinen et al., 2006) et implémentent un outil spécifique, appelé Poka, appliqué au domaine de la Finnish General Upper Ontology YSO[2]. Comme KIM, Poka extrait des entités nommées des textes soumis en entrée mais en considérant leur lemmatisation comme KCIT.

D'autres travaux similaires comprennent les outils S-Cream (Handschuch et al., 2002), MnM (Vargas-Vera et al., 2002), Artequakt (Alani et al., 2003) et OntoSophie (Valarakos et al., 2004). Nous pouvons noter d'importantes différences entre le cadre proposé par le CA Manager et ces approches : certains utilisent les techniques d'apprentissage automatique et d'autres celles du traitement automatique du langage, certains se basent sur des ontologies génériques comme PROTON ou YSO et d'autres sur des ontologies de domaine, ou bien encore soit ils peuplent l'ontologie soit ils annotent les ressources documentaires. La principale différence entre ces plateformes et le CA Manager est que ce dernier préserve l'indépendance entre les outils d'extraction d'information et le référentiel sémantique utilisé, proposant ainsi une capacité d'adaptation importante pour différents besoins applicatifs, comme cela est souvent demandé dans un cadre industriel.

Pour plus de détails sur les plateformes existantes d'annotation sémantique, nous référons le lecteur aux études (Uren et al., 2006) et (Reeve & Han, 2005).

3 L'extraction d'information à partir d'artefacts logiciels

L'extraction d'information est habituellement la première étape pour opérer des tâches de peuplement d'ontologie et d'annotation sémantique. Le processus automatique de production d'extractions sur la base d'une ontologie n'est pas trivial. En effet, le langage utilisé pour décrire les concepts et les relations dans les ontologies peuvent grandement différer du langage des contenus textuels et le langage naturel humain est lui-même bien connu pour son ambiguïté et sa complexité (Church & Patil, 1982). La plupart des approches utilisent des listes statiques de dictionnaires et repèrent seulement le terme exact issu d'un contenu parmi ceux de la liste. Nous avons développé l'outil KCIT (Key Concept Identification Tool) pour retrouver automatiquement des concepts clefs depuis des contenus textuels par rapport à une ontologie de référence. Les extractions sont créées en se basant sur l'hypothèse qu'une partie spécifique d'un contenu se réfère à une instance particulière si les deux lemmes

[1] Site web de PROTON : http://proton.semanticweb.org/

[2] Site web de YSO : http://www.seco.tkk.fi/ontologies/yso/

correspondent. En faisant se correspondre ces lemmes, nous nous assurons que toutes les flexions morphologiques des termes pertinents sont repérées.

Le processus d'extraction de notre outil KCIT comprend plusieurs étapes :

- *Construire une liste des termes pertinents*. Etant donné une ontologie, la lexicalisation de toutes les ressources ontologiques (classes, instances, propriétés, valeurs de propriétés) sont lemmatisées et ajoutées à la liste du vocabulaire contrôlé, dite "gazetteer list" dans GATE.
- *Annoter les contenus*. Le contenu des artefacts logiciels (plus ou moins structuré selon la nature de ces artefacts) est d'abord lemmatisé puis comparé avec le vocabulaire contrôlé construit à l'étape précédente.
- *Résoudre les conflits*. Les extractions sont filtrées afin de résoudre les problèmes d'ambiguïté tels que supprimer les extractions redondantes.

Les prochaines sections de cet article décrivent ces étapes en détail.

3.1 Construire une liste des termes pertinents

Pour initialiser l'outil KCIT, nous prétraitons les ressources ontologiques (classes, instances, propriétés et valeurs de propriétés) afin d'extraire toute lexicalisation compréhensible par un humain : la partie identificatrice de leur URI, leurs libellés (i.e. label) et les valeurs de leurs propriétés instanciées. Un ensemble de règles heuristiques est appliqué à chaque item de cette liste. Bien qu'il ne soit pas nécessaire de configurer spécifiquement KCIT pour l'utiliser avec différentes ontologies, ce paramétrage permet néanmoins d'obtenir de meilleurs résultats. Voici ci-dessous des exemples de règles appliquées par défaut par KCIT :

- Les caractères tiret ("-") ou soulignement ("_") sont remplacés par des espaces blancs. Exemple : *Project_Name* devient *Project Name*.
- Les mots comme *camelCased* sont découpés dans leurs mots constituants. Exemple : *projectName* devient *Project Name*.
- Si le texte contient des "stop words" comme *of, in, the*, etc., KCIT ignore aussi le texte qui suit. Exemple : *pos tagger for french* donne deux syntagmes, le texte original et *pos tagger*.

Chaque item de cette liste est analysé séparément par l'application Onto Root (en haut à droite de la Fig. 1), composée de plusieurs modules génériques de traitement du langage fournit par GATE (Cunningham et al., 2006). Chaque item est tout d'abord découpé en unité textuelle (Tokeniser) à laquelle est assignée une information syntaxique (POS Tagger) et un lemme (Morpho). C'est ce lemme (ou ensemble de lemmes) qui sera ajouté au vocabulaire dynamique (Ontology Resource Root Gazetteer). Par exemple, si une ressource est identifiée *ProjectName* par son URI et libellée *project names* dans l'ontologie, la liste créée avant d'exécuter Onto Root contient les termes suivants : *ProjectName* comme identifiant, *Project Name* comme découpage de l'identifiant et *project names* comme libellé. Chacun de ces items est alors analysé séparément des autres par Onto Root, produisant la même sortie pour *ProjectName* et *Project Name* mais ajoutant le lemme *project name* à partir du libellé *project names*.

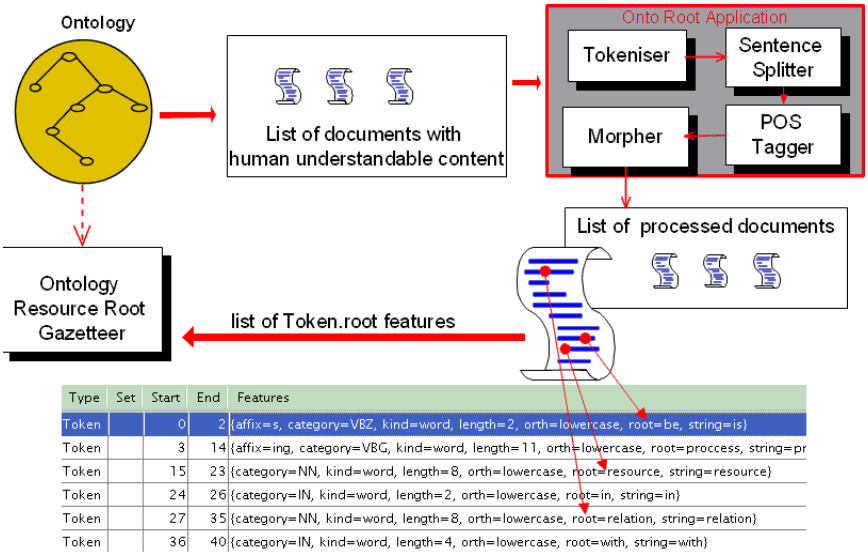

Fig. 1 – Construction automatique d'un vocabulaire à partir de l'ontologie

3.2 Annoter l'artefact logiciel avec les termes extraits

Une fois la liste des termes pertinents créée, la recherche de termes peut être appliquée sur notre corpus d'artefacts logiciels. A cause des variations morphologiques existant en anglais, comme d'en beaucoup d'autres langues, le comportement par défaut n'est souvent pas suffisant pour fournir la flexibilité requise et faire correspondre toutes les flexions morphologiques des termes pertinents. Afin de pouvoir comparer les lemmes créés lors de l'extraction vis-à-vis du vocabulaire dynamique généré à partir des termes ontologiques, la création des lemmes du contenu de l'artefact se fait aussi au moyen de l'outil Onto Root. Ainsi les lemmes extraits des ressources ontologiques et ceux de l'artefact analysé peuvent être comparés en faisant correspondre toutes les flexions morphologiques existantes des termes identifiés comme pertinents.

Les artefacts logiciels présentent toutefois un nouveau challenge pour les outils génériques de traitement du langage servant à délimiter les unités textuelles et les phases. En effet, certains de ces artefacts sont semi-structurés et contiennent des noms de variables, composés de un ou plusieurs mots (e.g., *getDocumentName*). Par conséquent, les ressources ANNIE English Tokeniser et Sentence Splitter (Cunningham et al., 2006) de GATE ont été paramétrées pour pouvoir analyser un code source Java et la JavaDoc associée. Ainsi *getDocumentName* est séparé en *get*, *Document*, et *Name*, avant d'être soumis aux algorithmes d'extraction.

3.3 Résoudre les conflits – Répondre au problème de l'ambiguïté

Nous l'avons dit, le langage humain est ambigu et il est possible d'utiliser la même expression dans différents contextes avec différents sens. Le processus d'analyse de

KCIT peut aussi conduire à la production d'annotations ambiguës à partir de la même unité textuelle ou d'un ensemble d'unités textuelles. Nous appliquons donc plusieurs règles heuristiques basées sur la structure de l'ontologie afin de résoudre ce problème d'ambiguïté. Par exemple, dans l'ontologie de domaine de GATE[3], il existe une instance, "*ANNIE POS Tagger*", partageant la même valeur pour son libellé et sa propriété *resourceHasName*. Cette expression comprend aussi le libellé de la classe *POS Tagger*. Lorsque le texte *ANNIE POS Tagger* apparaît dans un document, trois annotations sont créées qui nécessitent ensuite d'être désambiguïsées. Dans l'interface graphique, elles apparaissent comme des étiquettes se chevauchant (cf. *Start* et *End* de la Fig. 2). Dans l'ontologie GATE, la classe *POS Tagger* possède quatre instances dont *ANNIE POS Tagger*. Il devient donc possible de désambiguïser facilement les annotations à partir de l'instance correcte.

En sortie du processus, nous obtenons un ensemble d'extractions et leurs propriétés : l'URI de la ressource ontologique auquel le terme se réfère, son type (instance, classe, ou propriété) et d'autres caractéristiques pouvant être utilisées ultérieurement.

Type	Set	Start	End	Features
OntoRes		0	16	{instanceURI=http://gate.ac.uk/ns/gate-ontology#ANNIEANNIEPOSTagger, propertyName=resourceHasName
OntoRes		0	16	{URI=http://gate.ac.uk/ns/gate-ontology#ANNIEANNIEPOSTagger, type=instance}
OntoRes		6	16	{URI=http://gate.ac.uk/ns/gate-ontology#POSTagger, type=class}

Fig. 2 – Annotations ambiguës pour la chaîne d'entrée *ANNIE POS Tagger*

4 La plateforme Content Augmentation Manager

L'outil KCIT a été encapsulé dans une plateforme plus générale de production d'annotations sémantiques et de peuplement d'ontologie, nommée CA Manager. En fait, la philosophie au cœur du CA Manager est de pouvoir combler le fossé existant entre des outils d'extraction tel que KCIT et des référentiels sémantiques tel qu'ITM[4] ou Sesame[5]. Il a donc été conçu comme un médiateur, proposant une infrastructure modulaire et flexible qui lui permet de s'adapter à divers flux d'applications clientes et ce, quelque soit l'ontologie du domaine. Il est aussi capable de contrôler la qualité et la validité des résultats de l'extraction d'information par rapport à une ontologie donnée, de les confronter avec d'autres ressources existantes (internes ou externes) et de les enrichir.

Pour ce faire, le CA Manager repose à la fois sur les recommandations formulées par la communauté Web Sémantique (formats RDF/OWL, services web) ainsi que sur une infrastructure UIMA[6] (Unstructured Information Management Architecture) qui a été enrichie et configurée pour les tâches d'annotation et de peuplement.

[3]Ontologie décrivant l'application GATE : http://gate.ac.uk/ns/gate-kb

[4] ITM : http://mondeca.com/index.php/en/intelligent_topic_manager

[5] Sesame : http:// www.openrdf.org/

[6] UIMA : UIMA: http://www.alphaworks.ibm.com/tech/uima

4.1 Une plateforme web sémantique basée sur UIMA

L'infrastructure UIMA a pour objectif de fournir une plateforme de développement pour les outils de traitement automatique du langage naturel. Grâce à sa facilité de composition et d'intégration de modules internes ou externes, il a été rapidement accepté et recommandé par la communauté de l'Extraction d'Information (EI) et c'est pourquoi nous l'avons choisi comme fondation du CA Manager.

Néanmoins, bien qu'UIMA fournisse les bases pour développer un processus d'acquisition de connaissance, il ne donne pas pour autant des conseils sur ses étapes et leur enchaînement, notamment pour contrôler la validité des nouvelles annotations et instances. D'autre part, le Common Analysis Structure" (CAS) définit un schéma d'annotation de haut niveau qui doit être redéfini plus finement pour chaque nouveau besoin. Enfin, UIMA utilise une manière propriétaire d'exposer les web services (the Vinci IBM protocol), qui n'est pas réutilisable avec les standards prônés par la communauté Web Sémantique.

Dans la mise en œuvre de la plateforme du CA Manager (Martin et al., 2008), nous avons donc à la fois repris les préceptes formulés par UIMA comme le développement d'une architecture basée sur plusieurs moteurs d'analyse (Analysis Engines) permettant une configuration flexible et aisée des différents processus des applications clientes. Mais nous avons aussi tenté de faire évoluer l'infrastructure UIMA vers une exploitation des standards du Web Sémantique. Nous avons ainsi formalisé un schéma d'annotation en RDF, inspiré du CAS afin qu'il soit échangeable et enrichi au fur et à mesure du flux d'acquisition de connaissance, mais surtout générique aux tâches d'annotation et de peuplement quelque soit le domaine traité par l'ontologie de l'application. Enfin, nous avons développé la possibilité de déployer un processus de manière distribuée pour en améliorer la performance comme UIMA mais accessible via des web services reposant sur les langages et protocoles du Web Sémantique.

4.2 Extraire, Consolider et Stocker la connaissance

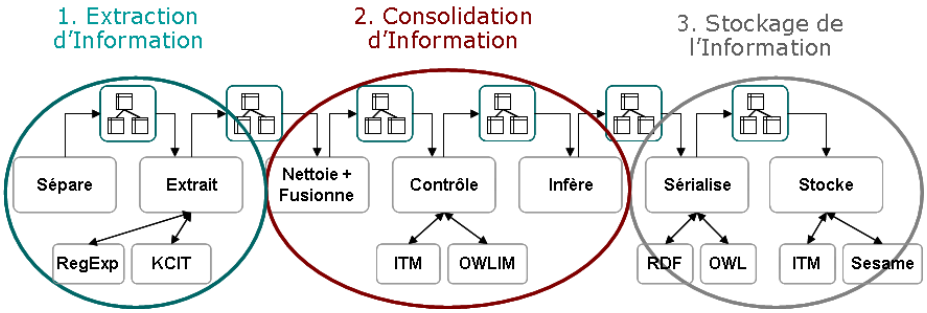

Fig. 3 – CA Manager : un processus modulaire et flexible

Le CA Manager propose donc un flux d'étapes logiques, cf. Fig. 3, dont certaines sont optionnelles, enrichissant progressivement le schéma d'annotation prédéfini. Ces

étapes peuvent être groupées en 3 modules principaux : 1) *Extraire* la connaissance pertinente et annoter le contenu; 2) *Consolider* les résultats vis à vis du modèle de l'ontologie et du référentiel sémantique; 3) Sérialiser la connaissance et le schéma d'annotation dans divers formats et les *stocker* dans le référentiel sémantique.

4.2.1 Extraire la connaissance depuis les textes

Le module d'Extraction d'Information est composé de deux étapes, *sépare* et *extrait*. La première étape, *sépare*, est optionnelle. Le contenu soumis en entrée au CA Manager est divisé en de multiples parties, e.g. un corpus en un ensemble de documents, ou un document en sous-sections bien identifiées.

La seconde étape, *extrait*, appelle l'outil d'extraction paramétré dans le flux, ici KCIT pour le traitement des artefacts logiciels, et récupère les occurrences des entités trouvées dans les artefacts. Un arbre conceptuel est généré en sortie, souvent dans un format propriétaire XML, qui a besoin d'être traduit dans le schéma d'annotation du CA Manager. Un mapping est préalablement créé manuellement entre toutes les entités pouvant être extraites et l'ontologie du domaine afin de générer ensuite automatiquement le schéma d'annotation de l'artefact analysé. Ce mapping est implémenté sous la forme d'un ensemble de règles d'acquisition de connaissance (RAC) tel que décrit dans (Amardeilh, 2007).

4.2.2 Consolider la connaissance grâce au référentiel

Comme indiqué dans (Alani et al., 2003), rares sont les outils pour le peuplement d'ontologie et l'annotation sémantique qui décrivent, ou même mentionnent, cette phase de consolidation dans leurs processus. Pourtant, cette phase est extrêmement importante afin d'assurer l'intégrité et la qualité du référentiel de l'application au moment du peuplement de l'ontologie. En fait, la plupart d'entre eux reposent seulement sur une validation manuelle pour contrôler les instances nouvellement ajoutées à la base de connaissance. A contrario, le CA Manager se distingue de ces outils en mettant l'accent sur l'automatisation de la consolidation des informations extraites en fonction de l'ontologie et de la base de connaissance. Afin de préserver l'intégrité de cette base de connaissance, la phase de consolidation est effectuée avant la création des instances dans le référentiel. Chaque algorithme de consolidation du CA Manager prend en compte deux axes : 1) la ressource ontologique concernée (instance de classe ou valeur de propriété); 2) les contraintes à contrôler (non redondance de l'information, restrictions de domaine et de couverture, cardinalités). Pour plus de détails sur ces algorithmes de consolidation, voir (Amardeilh, 2007). Dans le CA Manager, le module de consolidation se découpe en trois étapes, principalement *fusionne* et *contrôle*, et optionnellement *infère*.

La première étape, *fusionne*, vérifie que l'entité extraite n'existe pas en double dans le schéma d'annotation mais aussi dans la base de connaissance. Le référentiel sémantique est donc requêté afin de retrouver l'URI d'une instance présente dans la base de connaissance et correspondant à l'entité extraite. Les requêtes peuvent être simples (ex : nom de la classe et la chaîne de caractère identifiée dans le texte) ou multicritères (ex : nom de la classe et un ensemble de valeurs de propriétés identifiant

de manière non ambiguë une instance dans le référentiel). Par exemple, une personne peut être recherchée par son nom, comme l'instance *Niraj Aswani* de la classe *GATEDeveloper*. Mais dans des cas d'homonymie par exemple, rechercher le nom d'une personne n'est pas suffisant et il est plus pertinent d'interroger une combinaison de propriétés particulières et discriminantes (comme la date de naissance) pour filtrer parmi plusieurs instances ayant le même libellé.

L'étape *contrôle* vérifie que l'entité extraite est valide par rapport au modèle de l'ontologie. Ceci signifie contrôler le respect de l'appartenance à la classe la plus spécifique dans la hiérarchie, des domaines et couvertures pour les propriétés, des cardinalités, des formats des dates et des numériques, etc. Les triplets invalides sont étiquetés comme tels par une métadonnée de statut et stockés à part dans le référentiel sémantique sur le serveur. Ils peuvent être retrouvés ultérieurement afin notamment d'être présentés à un utilisateur final pour être manuellement validés.

La dernière étape de ce composant, *infère*, est optionnelle. Le composant peut se connecter à un moteur de raisonnement et d'inférence afin de découvrir de nouvelles entités pour relations entre elles mais aussi pour contrôler la cohérence globale et la qualité du référentiel sémantique après ces nouveaux ajouts.

4.2.3 Enregistrer les annotations et la connaissance dans les référentiels

Le composant Information Storage possède deux étapes, *sérialise* et *stocke*. L'étape *sérialise* parcourt le schéma d'annotation enrichi et consolidé par les étapes précédentes afin de générer une sortie dans le format requis par l'application cible (XML, RDF, OWL, etc.). La seconde étape, *stocke*, est optionnelle suivant si l'application utilise directement le format sérialisé ou si elle stocke les résultats dans un référentiel comme ITM et/ou dans un serveur d'annotation comme Sesame.

Le processus décrit ci-dessus est exposé comme un web service. Le flux de traitement de chaque application (ontologie du domaine, enchaînement des étapes, appels vers outils externes et format de sérialisation) doit être préalablement décrit dans un fichier de configuration situé sur le serveur. Une interface de test est fournie à l'adresse http://62.210.155.132/ca-test/. Nous avons aussi développé une interface pour la validation simultanée des annotations et des instances créées à partir du même artefact. Celles ayant été jugées *invalides* par les algorithmes de consolidation sont également récupérées et présentées à l'utilisateur comme "à valider". Il peut alors les corriger, sans risque de rendre le référentiel incohérent puisque l'interface repose entièrement sur l'ontologie qui contraint la saisie de l'utilisateur.

5 Evaluation

Dans le cadre du projet TAO[7] nous avons implémenté plusieurs processus pour tester la flexibilité du CA Manager : 1) ontologie PROTON + corpus d'articles de presse + expressions régulières simples + référentiel Sesame; 2) ontologie GATE + corpus d'artefacts logiciels + KCIT + référentiel Sesame; 3) ontologie GATE +

[7] TAO website : http://www.tao-project.eu

corpus d'artefacts logiciels + KCIT + référentiel ITM. Nous avons réalisé une évaluation de la chaîne complète de traitement sur la base du dernier processus à partir des mesures classiques de précision et de rappel sur la base d'un corpus constitué de 20 documents de la plateforme d'ingénierie textuelle GATE. Nous avons choisi différents types d'artefacts pour constituer un corpus représentatif, décomposé comme suit: 4 articles de forum de la mailing liste de GATE ; 3 classes java du code source de GATE ; 7 chapitres du manuel utilisateur de GATE ; 3 publications au sujet de GATE ; 2 pages Web accessibles à partir du site web gate.ac.uk ; 1 guide du développeur de GATE.. Mais comme les résultats dépendent grandement de la complexité et de la nature de l'ontologie utilisée, nous réétudions actuellement cette évaluation avec les mesures Learning Accuracy (Hahn & Schnattinger, 1998) et Balanced Distance Measure (BDM) (Maynard et al., 2008), qui permettent de prendre en compte ces contraintes.

Table 1. Precision and recall measures for the selection of the GATE software artefacts

	Precision	**Recall**
4 Forum posts	98.6111111	100.0
7 chapters of the GATE User Manual	96.0007672	96.9611548
2 Web pages	94.379845	95.0787402
3 publications	89.796798	95.8392268
3 java classes	97.2592593	98.8636364
1 GATE application developers guide	96.484375	98.4063745
Total	94.2830463	96.8763326

Nous avons aussi évalué plus spécifiquement la performance de l'outil d'extraction KCIT (cf. section 3) sur le même corpus à l'aide des mesures de précision et de rappel. Nous avons tout d'abord annoté manuellement ces documents afin de créer un corpus étalon de référence. Puis, nous l'avons traité automatiquement avec KCIT et comparé les résultats avec le corpus étalon. Les résultats sont présentés dans la Table 1.

Pour les 20 documents sélectionnés, 4523 annotations créées sont correctes, 41 annotations sont partiellement correctes, 126 annotations manquent, et 255 sont fausses. En regardant de plus près les documents annotés, la majorité des annotations manquantes ou fausses sont dues à la sortie du Morphological Analyser. Par exemple, l'analyseur n'a pas réussi à extraire correctement la racine des mots lorsque le pluriel des acronymes (ou des termes *camelCased*) était utilisé. Par exemple, la racine extraite pour *LanguageResources* restait *LanguageResources*. D'autre part, beaucoup d'annotations ont été créées dues au terme *learn*, bien que seulement une minorité soit correcte. La raison est que KCIT ne repose pas sur le contexte pour décider si un terme est pertinent ou non. Ainsi, chaque occurrence du mot *learn*, même dans la phrase « *learn GATE using movie tutorials* » était annoté comme référant au plugin appelé *learning*. De plus, certaines annotations se chevauchant n'étaient pas filtrées par KCIT, ceci affectant la performance globale. Par exemple, le terme *Stemmer PR* est annoté en référence à *Stemmer PR*, instance de la classe *Processing Resource*. Mais la partie libellée *PR* est aussi annotée, en référence à la classe *Processing*

Resource. Pourtant, bien que correcte, cette seconde annotation doit être supprimée durant la phase de filtrage de KCIT, car elle est redondante.

En général, nous pouvons conclure que la performance de KCIT est satisfaisante dans les cas où les documents sont courts et spécifiques au domaine. Par exemple, les annotations des articles de forums ou des classes java étaient produites avec une précision et surtout un rappel très bons. Plus les documents sont importants, moins le vocabulaire utilisé est spécifique au domaine (comme dans le cas des publications), et ainsi la performance se dégrade, mais toujours en restant à un niveau raisonnable. La raison de ceci, comme déjà mentionné, est que KCIT ne fait pas usage du contexte et échoue à filtrer les annotations qui sont étiquetées avec des concepts qui sont clairement en dehors du périmètre du domaine, mais partagent le même nom. Par exemple, *features* dans le contexte des artefacts logiciels de GATE est souvent utilisé pour décrire les caractéristiques des annotations générées par GATE. Cependant, lorsque le contexte traite des caractéristiques globales d'un composant logiciel, cette annotation du terme *features* a besoin d'être supprimée.

6 Conclusion et travaux futurs

Nous avons présenté la plateforme du CA Manager et son outil d'extraction d'information KCIT. KCIT est capable d'enrichir automatiquement ses dictionnaires à partir des libellés extraits et lemmatisés de l'ontologie de domaine pour repérer des termes liés à des artefacts logiciels. Le CA Manager sert de médiateur et consolide ces extractions par rapport au référentiel sémantique, peuple l'ontologie, crée les annotations sémantiques décrivant ces artefacts et stocke la nouvelle connaissance dans le référentiel. Cette plateforme innove par la combinaison d'une infrastructure UIMA et des technologies du Web Sémantique. Nous avons présenté les résultats d'évaluation de l'outil KCIT qui ont montré que sa performance dans des domaines restreints, comme celui de l'ingénierie logicielle, est raisonnable. Comme la performance de l'outil dépend de la couverture de l'ontologie du domaine, nous pensons qu'il est assez fiable pour couvrir n'importe quel domaine du moment que l'ontologie contient assez de données. Cependant, comme montré par les résultats d'évaluation, la phase de filtrage nécessite d'être améliorée, ce qui devrait réduire le nombre d'annotations ambiguës basées sur le contexte. Enfin, nos travaux futurs impliquent de finaliser l'évaluation du CA Manager en mettant à l'épreuve de nouvelles métriques comme la Learning Accuracy (LA) et la BDM.

Remerciements. Cette recherche est partiellement financée par le projet Transitioning Applications to Ontologies (TAO) du EU Sixth Framework Program (FP6-026460).

Références

ALANI H., KIM S., MILLARD D. E. (2003). Web based Knowledge Extraction and Consolidation for Automatic Ontology Instantiation. In Knowledge Capture Conference (K-CAP'03), Workshop on Knowledge Markup and Semantic Annotation, Florida.

AMARDEILH F. (2007). Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle. Thèse de doctorat, Univ. Paris X.

CHURCH K. & PATIL R. (1982). Coping with syntactic ambiguity or how to put the block in the box. In American Journal of Computational Linguistics, 8(3-4).

CIRAVEGNA F., DINGLI A., PETRELLI D., WILKS Y. (2002). User-system cooperation in document annotation based on information extraction. In 13th Int. Conf. on Knowledge Engineering and Management (EKAW'02), LNCS 2473, Springer-Verlag. p. 122-138. Madrid.

CORCHO O. (2006). Ontology based document annotation: trends and open research problems. In Int. J. Metadata, Semantics and Ontologies, 1(1), Inderscience. p. 47-57.

CUNNINGHAM H., MAYNARD D., BONTCHEVA K., TABLAN V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia.

CUNNINGHAM H., MAYNARD D., BONTCHEVA K., TABLAN V., URSU C., DIMITROV M., DOWMAN M., ASWANI N., ROBERTS I. (2006). Developing Language Processing Components with GATE Version 3.1 (a User Guide).

FELLBAUM C. (1998). WordNet - An Electronic Lexical Database. MIT Press.

HAHN U. & SCHNATTINGER K. (1998) Towards text knowledge engineering. In 15th National Conference on Artificial Intelligence (AAAI). p. 524-531, Menlo Park, CA, MIT Press.

HANDSCHUCH S., STAAB S., CIRAVEGNA F. (2002) S-CREAM - Semi-automatic creation of metadata. In 13th Int. Conf. on Knowledge Engineering and Management (EKAW'02), LNCS 2473, Springer-Verlag. p. 379-391. Madrid.

KIRYAKOV A., POPOV B., TERZIEV I., MANOV D., KIRILOV A., GORANOV M. (2005). Semantic annotation, indexing, and retrieval. In J. Web Semantics, Science, Services and Agents on the WWW, 2(1), Elsevier. p. 49-79.

MARTIN J., HERRERO G., CAPELLINI A., FRANCART T., AMARDEILH F., MARINOVA Z. (2008). TAO Suite: Architecture and integration requirements and specifications. Deliverable D5.2, TAO project IST-2004-026460.

MAYNARD D., PETERS W., YAOYONG LI. (2008). Evaluating evaluation metrics for ontology-based applications: infinite reflections. In Int. Conf. on Language Resources and Evaluation, Marrakech, Morocco.

REEVE L. & HAN H. (2005). Survey of semantic annotation platforms. In Symposium on Applied Computing (SAC'2005), Santa Fe, New Mexico, USA, p. 1634-1638.

UREN V., CIMIANO P., HANDSCHUH S., VARGAS-VERA M., MOTTA E., CIRAVEGNA F. (2006). Semantic annotation for knowledge management: requirements and a survey of the state of the art. In J. Web Semantics, Science, Services and Agents on the WWW, 4(1). p. 14-26.

VALARAKOS A., PALIOURAS G., KARKALETSIS V., VOUROS G. (2004). Enhancing the Ontological Knowledge through Ontology Population and Enrichment. In 14th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW'04), Lecture Notes in Artificial Intelligence, Vol. 3257, Springer-Verlag. p. 144-156.

VARGAS-VERA M., MOTTA E., DOMINGUE J. (2002). MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In 13th Int. Conf. on Knowledge Engineering and Management (EKAW'02), LNCS 2473, Springer-Verlag. p. 379-391. Madrid.

VEHVILINEN A., HYVNEN E., ALM O. (2006). A semi-automatic semantic annotation and authoring tool for a library help desk service. In First Semantic Authoring and Annotation Workshop.

WITTE R., ZHANG Y., RILLING J. (2007). Empowering software maintainers with semantic web technologies. In 4th European Semantic Web Conference.