UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Présentée et soutenue par

Noureddine MOKHTARI

Extraction et exploitation d'annotations sémantiques contextuelles à partir de texte

Thèse dirigée par Rose Dieng-Kuntz et Olivier Corby

Et préparée à l'INRIA Sophia Antipolis, Edelweiss

Jury

Président	Peter Sander
Rapporteurs	Nathalie AUSSENAC-GILLES Chantal REYNAUD
Examinateurs	Olivier Corby Yannick Toussaint

A mes parents, ma femme et toute ma famille

Remerciements

C’est un grand plaisir pour moi de remercier toutes les personnes qui ont permis à ce travail d’être ce qu’il est.

Je remercie tout d’abord la regrettée Mme. Rose Dieng-Kuntz, ancienne Directrice de l’équipe EDELWEISS et directrice de cette thèse, pour avoir bien voulu m’accueillir dans son équipe de recherche et m’avoir encadré pendant le début de cette thèse. Je lui exprime ma profonde gratitude. Sa gentillesse et ses encouragements me furent et demeurent d’un constant appui et sans lequel la persévérance m’aurait manqué.

Je remercie Mr. Peter SANDER, Professeur à l’université de Nice Sophia Antipolis, qui m’a fait l’honneur de présider le jury de cette thèse.

Je remercie Mme. Nathalie AUSSENAC-GILLES, Directrice de recherches CNRS à l’IRIT ainsi que Mme. Chantal REYNAUD, Professeur à l’université Paris-Sud pour avoir accepté de rapporter ce manuscrit, ainsi que pour l’intérêt qu’elles ont manifesté à l’égard de ce travail de thèse.

Je remercie Mr. Yannick TOUSSAINT, Chercheur à l’INRIA Loria d’avoir accepté d’examiner mes travaux.

Je remercie Mr Olivier CORBY, Directeur de recherches INRIA, pour avoir encadré mon travail, et pour son aide précieuse, sa patience, et son support inestimable durant ce travail.

Je remercie nos partenaires dans le projet européen SevenPro et plus particulièrement la fonderie Estanda qui a mis à disposition les supports nécessaires à nos expérimentations.

J’adresse mes sincères remerciements à mes collègues de l’équipe EDELWEISS: Alain Giboin, Fabien Gandon, Isabelle Mirbel, Khaled Khelif et Adil El_Ghali pour leur soutien constant et leurs remarques pertinentes. A Freddy Limpens, Guillaume Erétéo, Adrien Bass, Nicolas Delaforge, Sébastien Comos; et aux anciens membres : Emmanuel Jamin, Stéphanie Péron, Corentin Follenfant, Priscille Durville, Leila Khelif, Amel Yessad, Amira Tifous, Mohamed Bennis, Hacene Cherfi, Birahim Sall, Cheikh Anta Diop, Ibrahima Diop, Gaoussou Camara, Abdoulaye Guisse, Reda Boucid, Bassem Makni, Aroua Hedhili, Phuc-Hiep Luong, Sylvain Dehors, Virginie Bottolier, Patricia Maleyran; et bien sûr les nouveaux membres de l’équipe Edelweiss, Nicolas Marie, Oussama Cherif, Jerome Maraninchi, Sada Kalidou Sow, Cheikh Mbacké, Thiam, Guillaume Husson, et Pavel Arapov.

Je remercie tous les membres de l’INRIA Sophia Antipolis pour leur accueil.

Je remercie surtout très chaleureusement mes parents, ma femme et toute ma famille pour leur soutien constant à travers ces longues années.

Je remercie mes amis : Mohamed, Karim, Hassan, Said, Mohamed-said, Mohamed, Anis,...

Je remercie tous les membres de l’association « le Rappel » d’Antibes.

Je remercie enfin toutes les personnes que j’ai oublié de remercier ici.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Contexte industriel et scientifique

L'intérêt pour les sources d'information et de connaissance dans les entreprises (ou les organisations) ne cesse de croître. En effet, à l'heure actuelle, ces informations et ces connaissances jouent un rôle crucial pour leur développement. Pour une meilleure productivité, un travail collectif conduit intuitivement à la nécessité de partager des connaissances. Ceci ne peut que faire croître l'importante des sources de connaissances et rend leur gestion nécessaire par les entreprises. Un système de gestion des connaissances s'impose pour offrir un accès global aux sources d'informations et faciliter le partage des connaissances.

Par ailleurs, suite au développement considérable de l'accès à Internet, la colossale masse d'information disponible sur le Web est devenue une mine d'or pour les entreprises. Cependant, l'exploitation de ce volume d'information, qui ne cesse de croitre, devient difficile et cela est dû, principalement, aux obstacles liés à la structuration et la représentation. Ces limites du Web actuel ont vu la naissance de la nouvelle génération du Web, le Web sémantique, qui offre des solutions pour compléter le contenu des ressources du Web en ajoutant de la sémantique sous forme de méta-données (annotations) en vue de rendre les ressources *compréhensibles* par des machines [Berners-Lee et al., 2001]. On entend par là, décrire des ressources selon une représentation formelle avec une sémantique clairement définie et qui soit conçue pour une interprétation par des programmes.

L'approche de l'équipe Edelweiss[1] repose, d'une part, sur des technologies du Web sémantique, et d'autre part, sur les moyens de communication de l'entreprise (i.e. l'intranet, l'intraweb) pour offrir un *Web Sémantique d'entreprise* (ou Web sémantique d'organisation) [Dieng-Kuntz, 2005] constitué *(i)* de ressources (documents, personnes, services), *(ii)* d'ontologies (décrivant le vocabulaire conceptuel) et *(iii)* d'annotations sémantiques sur les ressources (e.g. compétences des personnes, contenu sémantique des documents, etc.).

Dans ce cadre, l'équipe Edelweiss de l'*INRIA Sophia Antipolis* avait participé au projet européen SevenPro[2] visant à améliorer le processus d'ingénierie de production dans les entreprises de fabrication, au moyen de l'acquisition (basée sur une ontologie), la formalisation et l'exploitation des connaissances.

L'extraction d'information (ou l'annotation sémantique) fait partie des axes de recherche traités dans le projet SevenPro. Nos travaux s'inscrivent dans ce thème de recherche avec des propositions allant dans le sens de solutions génériques et évolutives pour l'extraction d'information à partir de texte.

---

[1] http://www-sop.inria.fr/edelweiss/

[2] Sevenpro: Semantic Virtual Engineering Environment For Product Design.

Problématique et objectifs poursuivis

L'hétérogénéité des sources d'informations textuelles, la difficulté d'extraire et d'exploiter ces informations ainsi que leur pertinence relative et évolutive font partie des problèmes des systèmes de gestion des connaissances. Plusieurs solutions sont proposées dans le sens de cet axe de recherche, ayant pour objectif de faciliter, d'une part, l'extraction de ces informations pertinentes (annotations) à partir de textes, et d'autre part, leur exploitation.

Les types d'annotations générées sont divers autant que les motivations et les besoins initiaux des approches proposées. Certaines approches extraient des annotations sous forme d'un ensemble de termes (ou concepts) afin de caractériser le thème d'un document, le résumer, l'indexer, etc. Certaines autres s'intéressent plus au sens du texte en offrant des concepts reliés avec des relations pour permettre, entre autre, d'avoir des réponses plus riche sémantiquement.

Dans un cadre industriel et particulièrement dans le cas de nos partenaires, le besoin de qualité et de précision de l'information est primordial. Les annotations sommaires ne sont ni pertinentes ni adaptées à leurs besoins.

Avec la masse colossale d'informations textuelles, la demande ne cesse de croître pour obtenir des réponses rapides, précises, et plus riche sémantiquement. Il est devenu primordial d'avoir un nouveau regard sur les textes pour en exploiter plus encore la richesse sémantique, avec comme support les formalismes de représentation des connaissances et de raisonnement, dans un cadre standardisé qui favorise le partage et la réutilisation.

Une des pistes que nous avons explorées est inspirée du comportement humain. En effet, quand les humains parlent entre eux, ils sont capables d'utiliser les informations implicites de la situation, ou son contexte, et d'augmenter la portée de la conversation. Malheureusement, cette capacité à transmettre des idées n'est pas la même pour les humains interagissant avec les ordinateurs. Par conséquent, les ordinateurs ne sont pas actuellement en mesure de tirer pleinement parti du contexte du dialogue homme-machine. En améliorant l'accès de l'ordinateur au contexte, on augmente la richesse sémantique de la communication dans l'interaction homme-machine et ceci est l'un des objectifs du Web sémantique.

Dans le domaine de l'extraction d'information à partir de texte, les types d'informations extraites peuvent être des mots clés, des termes (associés le plus souvent à des concepts d'une ontologie du domaine) [Maynard et al. 1999], ou des « annotations sémantiques » sous forme de triplets « sujet-prédicat-objet » [Khelif et al. 2007]. Actuellement, la forme la plus 'sémantique' des informations extraites est l' « annotation sémantique » représentée, le plus souvent, sous forme de triplets RDF. Dans ce type de bases d'annotations, les triplets générés sont tous au même niveau : il y a une absence de relations entre les annotations. Cette structure « plate » des bases d'annotations ne tient pas compte des dépendances sémantiques des annotations les unes par rapport aux autres. Il est par conséquent possible de trouver dans la base d'annotations des triplets avec des sens contradictoires. L'annotation sémantique qui devrait être vraie dans un contexte particulier est vraie dans toute la base d'annotation au même niveau que toute autre annotation.

Ceci nous a poussé à revoir les processus de génération d'annotations actuels et nous avons fait la constatation suivante : dans les travaux sur la génération d' « annotations sémantiques »

les informations permettant de lier les annotations entre elles sont négligées complètement ou partiellement. Ces informations concernent le 'contexte' d'apparition des annotations.

Les travaux qui se sont intéressés au 'contexte' des annotations utilisent certains aspects du 'contexte' et en négligent d'autres. Nous citons à titre d'exemple le système d'identification d'instances de classes à partir de texte « C-Pankow » [Cimiano et al. 2005] qui exploite le contexte d'apparition des instances pour résoudre les ambiguïtés. Dans d'autres travaux tels que [Goujon, 1999a] où des marqueurs linguistiques sont perçu comme un « contexte ». Dans d'autres travaux tels que le système OntoSeek [Guarino et al. 1999], l'exploitation de la structure des documents peut être perçue comme une exploitation du 'contexte'.

Dans le cadre de cette quête, à savoir, exploiter le contexte d'apparition des annotations dans le processus de génération d'annotations, nous cherchons à répondre aux questions suivantes :

Qu'est ce qu'un « contexte » et comment le modéliser ?

Il s'agit de définir la notion de contexte, et comment elle est perçue par différentes communautés.

Quel apport pourra avoir le « contexte » pour les annotations ?

Il s'agit, en plus de définir les annotations contextuelles, d'expliciter l'importance de la sémantique ajoutée aux annotations via la notion de « contexte ».

Comment extraire des annotations en prenant en compte leur contexte ?

Il s'agit d'offrir des outils de génération des annotations contextuelle à partir de texte et ce afin d'alimenter la base de connaissances.

Comment modéliser et formaliser les connaissances ?

Il s'agit de choisir un modèle (les ontologies à titre d'exemple) pour représenter formellement les connaissances afin de faciliter leur partage et l'interopérabilité entre les différents acteurs qu'ils soient humains ou informatiques.

Peut-on offrir un système automatique de génération d'annotations ?

Il s'agit de pousser l'automatisation de la génération des annotations à son plus haut niveau.

En réponse aux besoins exprimés par nos partenaires industriels dans le cadre du projet SevenPro et en exploitant la notion de « contexte » dans le processus d'annotation, nous nous sommes fixés les objectifs suivants :

- Proposer une approche de génération d'annotations contextuelles à partir de texte ;
- Intégrer les techniques du Web sémantique dans chaque étape de la méthodologie.

Contributions et champs de recherche concernés

En tenant compte des objectifs indiqués dans la section précédente, nous proposons l'approche CEEMA[3] (méthode d'extraction et d'exploitation des annotations contextuelles).

L'approche CEEMA possède les caractéristiques génériques suivantes :

- Elle est basée sur **les ontologies** pour **la description et la formalisation des connaissances** ;
- Elle intègre des techniques de génération d'annotations **permettant d'alimenter la base de connaissances au fur et à mesure** de l'arrivée de nouvelles sources textuelles ;
- Elle admet en entrée du **texte brut** avec une **ontologie** du domaine.

L'approche CEEMA possède les caractéristiques originales suivantes :

- Elle exploite **la notion de contexte** pour le processus de génération des annotations : la notion de contexte est perçue de différentes manières selon les auteurs dans le domaine de l'extraction d'information. Ceci nous a poussé à revoir les définitions de cette notion pour proposer une projection des aspects important du 'contexte' sur notre domaine de génération d'annotation sémantique. D'une manière générale, nous définissons le contexte d'une annotation par toutes les annotations qui sont liées à celle-ci par des relations contextuelles. Les relations contextuelles peuvent être des relations rhétoriques, spatiales, temporelles ou issues de la structure physique du texte comme la relation d'imbrication ;
- Elle exploite **la structure physique du texte** : CEEMA intègre une méthode permettant de reconstituer la structure physique du document avec ses imbrications (titres, paragraphes, phrases, segments). La structure physique du texte est exploitée lors de la génération des triplets RDF. L'idée principale est d'identifier une propriété dans un segment de texte et de sélectionner pour cette propriété son *sujet* et son *objet* associés. Le sujet et l'objet de la propriété sont sélectionnés, d'abord, dans le segment où elle a été identifiée, puis en cas d'échec dans la phrase qui l'englobe, puis de manière itérative dans des segments de plus grande portée ;
- Elle exploite **la structure logique du texte** : les annotations sémantiques sont liées entre elles par des relations contextuelles qui définissent le sens logique du texte, entre autres, les relations rhétoriques. Une méthode intégrée dans CEEMA permet de déterminer les arguments de ce type de relation. Cette méthode est inspirée de la théorie d'analyse RST (Rhetorical Structure Theory) [Mann et al. 1987] ;
- Elle génère **des annotations contextuelles** : l'aspect contextuel des annotations est vu à deux niveau : i) le premier niveau est matérialisé par le processus de

[3] CEEMA: Contextual Extraction and Exploitation Method of Annotation

génération de l'annotation sémantique lui-même et l'utilisation de la structure physique pour augmenter la portée à utiliser pour sélectionner les éléments constituant un triplet ; ii) le deuxième niveau est matérialisé par les relations contextuelles qui lient les annotations sémantiques. En plus des relations rhétoriques, d'autres relations contextuelles entre les annotations peuvent être déduites. En effet, la structure physique du texte est utilisée pour reconstituer les emboitements avec des graphes nommés entre les triplets RDF générés selon l'emplacement de leur propriété dans le texte. Par exemple, un graphe nommé regroupant les triplets d'une phrase est emboité dans un autre graphe qui représente les annotations générées dans le paragraphe englobant. Ces emboitements représentent les dépendances entre annotations sémantiques liées à la structure physique du texte ;

- Elle est **automatisée** : notre approche est basée sur des règles de grammaire pour identifier les instances de concepts et de propriétés. Dans la littérature, l'écriture de ces règles est usuellement une étape manuelle et est souvent décrite comme fastidieuse. Dans l'approche CEEMA, nous avons proposé une méthode basée sur l'ontologie du domaine pour automatiser cette étape. Nous avons exploité les labels (rdfs:label) des classes et des propriétés dans l'ontologie du domaine pour engendrer les règles de grammaires permettant d'identifier les instances des classes et des propriétés dans le texte.

Comme nous pouvons le constater, l'approche CEEMA est générique et ne dépend pas d'un domaine particulier. En effet, elle est réutilisable pour tout domaine et admet comme entrée une ontologie du domaine et des textes bruts du même domaine que l'ontologie.

Nos contributions se situent à la croisée de quatre axes de recherche :

- L'**ingénierie des connaissance**s : l'objectif de ce travail est d'extraire des annotations 'sémantiques' avec l'introduction de la notion de contexte;
- Les technologies du **Web Sémantique** : ce choix a été fait en vue d'offrir des connaissances (annotations + ontologies) réutilisables en optant pour les standards du Web sémantique comme moyen de représentation des connaissances ;
- Le **traitement automatique de la langue naturelle** (TALN) : en cherchant à puiser des connaissances de la masse volumineuse des textes qui ne cesse de croitre, nous tentons d'automatiser ce processus d'extraction ;
- La **notion de contexte** : vu la richesse sémantique que peut offrir la notion de contexte, plusieurs approches ont émergé autour de la définition de cette notion, son exploitation ou sa représentation. C'est devenu un axe de recherche à part entière qui concerne plusieurs domaines.

Organisation du document

Le premier chapitre dresse un état de l'art concernant les différentes approches de génération d'annotations à partie du texte. Nous commençons d'abord par les plateformes (ou systèmes) d'extraction d'information. Ensuite nous abordons les travaux qui ont utilisé, d'une manière ou d'une autre, la notion de contexte en mettant l'accent sur les différents points de vue sur la notion de « contexte » d'une approche à une autre. Nous abordons par la suite les approches d'extraction d'information basées sur les ontologies qui nous intéressent plus particulièrement. En effet, beaucoup de travaux ont émergé dans le domaine de l'extraction d'information intégrant l'exploitation des ontologies dans le processus d'extraction. Enfin, nous exposons un comparatif entre ces approches pour situer celle que nous proposons.

Dans le deuxième chapitre, nous présentons un état de l'art sur les modèles existant et permettant de modéliser la notion de « contexte ». Ceci nous permettra d'avoir une vue globale sur l'utilisation du contexte dans des domaines où la notion de « contexte » est exploitée d'une manière massive. Les travaux les plus connus sur les modèles du contexte font partie de la communauté informatique pervasive. En effet, il est judicieux de s'inspirer de ces travaux existant pour la modélisation et l'utilisation de la notion de contexte dans le domaine de génération d'annotation sémantique à partir du texte. L'idée principale est de faire une projection de la notion de « contexte » dans le domaine de l'informatique pervasive sur le domaine de génération d'annotation sémantique. Ceci nous a permis de mettre en relief certains aspects dans la modélisation des informations sur le contexte de type textuel tels que :

- La prise en compte de l'hétérogénéité des sources d'information ;
- L'importance de l'aspect spatial qui peut être assimilé à la structure physique du texte ;
- L'importance de l'aspect temporel ;
- L'importance des historiques bien qu'ils soient difficiles à gérer ;
- Les informations peuvent être incorrectes, ce qui mène à la nécessité de gérer les incohérences ;
- Un mécanisme de raisonnement efficace est requis ;
- Un modèle de représentation simple tel que RDF est adapté ;
- Une modélisation permettant de distinguer certaines informations pertinentes par rapport à d'autres ;
- L'importance de la gestion de la granularité d'information sur le contexte.

Dans le troisième chapitre, nous abordons des définitions de certains termes tels que « annotation », « méta-donnée », « annotation sémantique » et plus spécialement celles de la notion de « contexte ». Nous exposons, par la suite, plusieurs définitions du contexte existant dans la littérature.

La définition du contexte qui a retenu notre attention concerne le domaine de la logique et est celle de McCarthy qui est considéré comme le maitre en la matière. Cette définition [McCarthy, 1993] postule qu'une proposition « p » est vraie dans un contexte « c », où « c » est supposé capturer tout ce qui n'est pas explicite dans « p » mais qui est requis pour faire de « p » un énoncé significatif pour représenter ce qu'il est supposé établir. D'une manière similaire, dans la définition que nous proposerons du contexte d'une annotation sémantique, nous nous intéresserons aux relations dites contextuelles pouvant exister entre annotations sémantiques. Nous exposons aussi la définition de la notion d'« annotations contextuelles ». Nous montrons par la suite, comment la prise en compte de la granularité d'information est matérialisée dans notre modélisation du contexte. Enfin nous étudions les différents types de relations contextuelles pouvant exister entre les annotations sémantiques : relations contextuelles structurelles, spatiales, temporelles ou rhétoriques. Nous mettons l'accent sur les relations rhétoriques avec ce qu'elles peuvent apporter de sémantique et nous abordons un des moyens d'analyse du discours, à savoir, la théorie RST.

Dans le quatrième chapitre nous exposons notre approche de génération d'annotations sémantiques contextuelles à partir de texte. Nous abordons, tout d'abord, l'architecture générale de l'approche qui est constituée de deux principales phases : manipulation textuelle et manipulation sémantique.

La première phase consiste à reconstruire la structure physique d'un texte brut donné en entrée. En premier lieu une chaine de traitement est lancée sur le texte pour identifier les différents éléments textuels : tokens (mots, chiffre, etc.), phrases, paragraphes, etc. Des indicateurs numériques sont utilisés pour identifier les titres. Les imbrications sont ensuite construites entre : titres, paragraphes, phrases, etc. Les algorithmes permettant ces manipulations de la structure physiques du texte sont détaillés. A partir des éléments textuels identifiés et leurs imbrications des uns par rapport aux autres, nous déduisons les relations textuelles issues de la structure physique du texte.

La deuxième phase consiste à identifier les relations contextuelles de type rhétorique en utilisant des règles JAPE[4] [Cunningham et al., 2002] générées automatiquement. De même, en utilisant des règles JAPE générées automatiquement, nous identifions les instances de classes et de propriétés. Ensuite, nous générons des annotations sémantiques sous forme de « triplets RDF » en associant un sujet et un objet pour les propriétés identifiées tout en prenant en compte la position des sujets et objets dans les imbrications de la structure physique du texte. Par la suite les annotations générées sont reliées par les relations contextuelles formant ainsi des annotations sémantiques contextuelles.

Tous les algorithmes décrivant les étapes de ces deux phases sont exposés dans ce chapitre.

Dans le cinquième chapitre nous abordons l'implémentation de notre approche. Les étapes de l'approche CEEMA sont illustrées en déroulant un exemple de texte. Nous commençons par aborder l'architecture générale de notre système CEEMA-T[5]. Nous détaillons par la suite les outils de TALN utilisés (GATE, JAPE, TreeTagger), et ainsi que les technologies utilisés (moteur sémantique Corese, moteur de recherche XQuery Qizx/open et les transformations

[4] Java Annotation Patterns Engine

[5] CEEMA-T: Contextual Extraction and Exploitation Method of Annotation - Tool

XSLT). Ensuite, nous exposons les différents choix techniques adoptés ainsi que les problèmes rencontrés et les solutions proposées.

Nous détaillerons aussi, les différentes étapes de la chaîne de traitement implémentée avec la plate-forme GATE. Cette chaîne est constituée de trois étapes principales :

- *Prétraitement* : cette étape initiale permet de convertir les différents formats texte (pdf, ps, etc.) au format « txt » en conservant la structure physique du texte ;
- *Analyse morpho-syntaxique des textes* : cette étapes consiste à découper du texte (en phrase et en paragraphes), à faire une « tokenisation » (tokens de type nombre, ponctuation, etc.), à faire une lemmatisation ; à affecter à chaque *token* une catégorie d'ordre grammaticale (Verbe, Nom, Adjectif, etc.) ;
- *Détection des propriétés, des instances et des relations contextuelles* : les règles JAPE sont implémentées sous forme de *transducer* dans GATE.

Nous abordons par la suite le choix technique de XQuery pour le reconstitution de la structure du texte. En outre, nous argumentons le choix d'XSLT pour implémenter l'algorithme de générations des annotations sémantiques contextuelles.

Dans le sixième chapitre, nous abordons les expérimentations effectuées pour l'évaluation et la validation de notre méthodologie. Nous commençons par décrire le cadre du projet européen SevenPro ainsi que les données sur lesquelles l'expérimentation est faite. Nous donnons par la suite un bref (pour des raisons de confidentialité) descriptif de l'ontologie du domaine utilisée dans l'expérimentation. Ensuite, nous présentons une étude quantitative et qualitative de la méthodologie de génération des annotations contextuelles, d'une part, pour valider son aspect automatique, et d'autre part, pour juger la qualité des annotations générées.

Une première expérimentation est faite sur un corpus de taille réduite pour valider :

- La capacité des grammaires de détection à identifier les instances possibles d'une propriété dans le texte ;
- La capacité des grammaires de détection à identifier les instances de concepts pouvant être liées par les propriétés;
- La capacité des grammaires de détection à identifier les relations dites contextuelles (rhétoriques ou autre) ;
- La capacité de l'algorithme de reconstitution de la structure physique du texte à construire correctement les imbrications hiérarchiques pour un texte donnée;

Une deuxième expérimentation est faite sur un corpus de taille plus large pour valider la capacité à relier les bonnes instances de concepts par les bonnes propriétés (génération des triplets RDF), tout en garantissant la cohérence avec les contraintes décrites dans l'ontologie du domaine (domain, range) ;

Une troisième expérimentation étudie l'impact de l'utilisation du contexte dans la génération des annotations sémantiques. Nous terminons par une discussion sur les résultats obtenus lors des expérimentations.

Chapitre I : Etat de l’art sur l’extraction d’informations à partir de texte et la génération d’annotations

1 Introduction

Plusieurs classifications existent pour les travaux sur l'extraction d'information à partir de texte:

Selon le type de texte dont nous voulons extraire des annotations [Kshitija et al. 2008] :

- donnée de type *structurée*: ce type de donnée associé aux pages Web suivant une structure prédéfinie (par exemple un schéma XML ou une DTD). Nous pouvons associer à cette catégorie tous les travaux essayant d'extraire des annotations à partir de pages web et plus spécialement les informations disposées sous forme de listes, d'arbres et de tableaux.
- donnée de type *semi-structurée* : ce type de donnée suit une structure hiérarchique mais non prédéfinie. Nous pouvons mettre dans cette catégorie les pages Web qui ne suivent pas une structure particulière.
- donnée de type *non-structurée* : ces données sont souvent assimilées au texte brut. Le problème d'extraction d'information pour ce type de donnée est traité par les domaines de la fouille de texte, du traitement automatique de la langue, de l'apprentissage automatique et les systèmes question-réponse. Nous nous sommes intéressés plus particulièrement à ce type de texte.

Selon la technique utilisée pour extraire des annotations [Reeve et al. 2005]:

- *les approches basées sur l'apprentissage automatique*
    - Les approches probabilistes : ces types de travaux utilisent des modèles statistiques pour prédire la localisation des entités dans le texte. Par exemple, l'algorithme DATAMOLD [Borkar et al. 2001] utilise un modèle de Markov caché pour identifier des instances dans le texte.
    - D'autres travaux se basent sur des inductions telles que la boite à outils Amilcare [Ciravegna et al. 2002] dont le noyau algorithmique d'extraction d'information est basé sur des règles d'inductions.
- *les approches basées sur les patrons*
    - Beaucoup de travaux utilisant cette approche suivent la simple méthode décrite par Brin [Brin, 1998] centrée sur une base de patrons réduite et qui est initialement construite, souvent manuellement, pour identifier des entités nommées. Cette base de patrons est enrichie avec de nouveaux patrons construits à partir des nouvelles entités nommées identifiées. Cette augmentation est répétée jusqu'à ce qu'il n'y ait plus d'entités à identifier, ou que l'utilisateur arrête le processus.
    - D'autres travaux proposent des approches basées sur des règles construites manuellement pour générer des annotations. Un des moyens de construction de règles est la grammaire JAPE [Cunningham, et al. 2000]. Nous classons aussi dans cette partie les travaux qui identifient des termes candidats dans le texte à l'aide d'une taxonomie. Pour ces travaux une désambiguïsation des termes candidats est requise et est souvent faite à l'aide d'un corpus de termes déjà existant.

Selon la source d'information utilisée pour extraire des annotations [Mokhtari et al. 2008] :

- *Annotation par le contenu du document :* On distingue deux types de techniques d'annotation de documents par le contenu (ou indexation) : la technique classique, qui consiste en général à attribuer un ensemble de mots clés (ou termes) à chaque document, et la technique sémantique qui attribue une annotation basée sur des concepts (et non de simples mots-clés) et éventuellement sur les relations entre eux. Les travaux visant à extraire des annotations par le contenu se focalisent généralement sur l'extraction des termes. [Guarino et al., 1999] [Khelif et al., 2007] prennent également en compte les relations sémantiques entre termes. Dans [Guarino et al., 1999], les auteurs décrivent OntoSeek un système de recherche documentaire en ligne pour les « pages jaunes ». Afin de construire automatiquement des résumés, [Berri J., 1996] utilise la méthode d'exploration contextuelle [Desclés J. P. et al., 1994] qui repose sur des critères linguistiques et qui consiste à affecter des étiquettes sémantiques aux phrases contenant des indicateurs pertinents. Dans [Desmontils et al., 2002], est proposée une approche supervisée pour indexer des ressources web en reposant sur le contenu des pages web à l'aide d'une ontologie ; les termes sont pondérés par rapport à leur importance dans la page (titre, paragraphe,…). D'autres travaux utilisent les techniques d'extraction d'information pour l'annotation de textes dans un domaine particulier : par exemple, la génomique [Nédellec, 2004].
- *Annotation à partir de sources externes au document* : Les travaux cités dans cette partie font appel à des ressources externes au document. [Njmogue, et al. 2004] propose une approche basée sur un référentiel métier. L'idée principale est que l'indexation d'un document dépend des activités de l'entreprise et non pas des mots clés du document. Cette approche utilise à la fois une analyse linguistique et statistique du document et un traitement sémantique. Nous pouvons souligner que les activités de l'entreprise peuvent être considérées comme un contexte d'utilisation des documents. Dans [Abrouk, 2006], l'auteur propose une approche pour l'annotation semi-automatique de ressources selon les liens de référencement et ce, sans connaissance préalable du contenu du document. D'autres travaux se sont intéressés à modéliser le processus de recherche d'information et la modélisation de l'utilisateur. Nous citons à titre d'exemple [Hernandez, 2005] qui a comme but principal d'associer le thème relatif au document et la tâche de recherche d'information (l'intention), pour offrir un système de recherche d'information.

Par ailleurs, nous nous intéressons plus particulièrement aux approches dites « basées sur les ontologies ». Les Approches d'Extraction d'Informations (ou annotations) Basées sur les Ontologies (AEIBO) sont souvent automatiques ou semi-automatiques et offrent des résultats intéressants. Ces approches peuvent utiliser des techniques variées (probabiliste, règle linguistique, basée sur les données du Web, etc.) ou encore spécifiques à un type particulier de texte (Wikipédia, page jaune,etc.).

Nous ne pouvons parler d'annotations à partir de texte sans aborder les plateformes (ou systèmes) d'annotations. En effet, des travaux sur l'extraction d'information à partir de texte proposent des plateformes dites d'annotation. Ces plateformes (ou systèmes) d'annotation sont souvent extensibles et adaptables afin qu'elles soient utilisées comme support pour proposer des solutions plus performantes. Nous passerons en revue quelques unes de ces plateformes d'annotation. Ensuite nous abordons les travaux qui ont utilisé, d'une manière ou d'une autre, la notion de contexte en mettant l'accent sur les différents points de vue sur la notion de « contexte » d'une approche à une autre. Nous abordons par la suite les approches d'extraction d'information basées sur les ontologies qui nous intéressent plus particulièrement. En effet, beaucoup de travaux ont émergé dans le domaine de l'extraction

d'information intégrant l'exploitation des ontologies dans le processus d'extraction. Enfin, nous exposons un comparatif entre ces approches pour situer celle que nous proposons.

2 Plateformes et systèmes d'annotation

Plusieurs travaux sur l'extraction d'information à partir de texte ont abouti au développement de plateformes/systèmes d'annotation. Ces plateformes varient dans leur architecture, le type d'annotation générée, leur méthode (manuelle, semi automatique ou automatique) et même leur gestion des bases d'annotation.

Nous nous focaliserons plus particulièrement sur les plateformes (et systèmes) d'annotations automatiques ou semi-automatiques.

2.1 GATE

L'architecture générale d'ingénierie du texte (GATE) est un framework pour le développement et le déploiement des technologies du traitement automatique de la langue à grande échelle [Cunningham et al. 2002]. Il prévoit trois types de ressources : les ressources linguistiques (RL) qui, collectivement, se réfèrent aux données; les ressources de traitement (RT) qui sont des algorithmes, et des ressources de visualisation (RV) qui représentent la visualisation/édition des composants. GATE peut être utilisé pour traiter les documents dans différents formats, texte brut, HTML, XML, SGML et RTF.

Quand un document est ouvert dans GATE, un analyseur de structure du document est appelé et est responsable de la création de: un *document Gate*, une RL qui contiendra le texte du document original et un ou plusieurs ensembles d'annotations. Dans GATE, une annotation est définie par un ensemble d'attributs qui catégorisent une partie du texte. Par exemple, une annotation caractérisant un paragraphe contient : l'attribut « Type » avec comme valeur « paragraph », l'attribut « Start » qui définit le début du paragraphe dans le texte ; l'attribut « End » qui définit la fin du paragraphe dans le texte. En plus de l'annotation de type « paragraph », d'autres informations à extraire sont proposées par défaut dans GATE. En effet, GATE propose le système d'extraction d'information générique par défaut ANNIE [Maynard et al. 2001] qui regroupe un ensemble de RT et qui permet d'identifier des concepts génériques tels que des noms de personnes, des lieux, des organisations, des dates, etc.

Les RT les plus courantes sont les segmenteurs (Tokenizers), les analyseurs morpho-syntaxiques (Part Of Speech ou POS Taggers), les lexiques (Gazetteers), les transducteurs (JAPE transducers), et les patrons d'extraction (Templates).

Les annotations sont généralement mises à jour par le biais des RT lors de l'analyse du texte. Mais les annotations peuvent aussi être créées lors de l'édition directe du texte dans l'interface graphique GATE.

2.2 KIM

Bien que Kim [Popov et al. 2003, 2004] [Kiryakov et al. 2005] soit une plateforme d'annotation, il peut aussi être classé parmi les systèmes d'extraction d'information basés sur

les ontologies. En effet, pour annoter des pages Web, Kim se base, d'une part, sur son ontologie générique PROTON[6], et d'autre part, sur l'architecture GATE [Cunningham et al. 2002] pour utiliser ses dictionnaires et patrons d'extraction. Les annotations générées par Kim sont des entités nommées (personnes, lieux, etc.). Ces entités nommées sont identifiées principalement en se référant à une très grande liste d'entités nommées existante appelée « gazetteer » (ou dictionnaire). De plus, Kim est capable de relier les entités nommées identifiées à l'URI d'une instance particulière dans l'ontologie. Les annotations générées peuvent ensuite être exploitées pour l'indexation et la recherche sémantique, la co-occurrence et l'analyse des tendances de popularité.

Kim est un système basé sur une approche, comme beaucoup d'autres systèmes, qui nécessite une intervention humaine en vue d'apporter des adaptations à de nouvelles ontologies.

2.3 Ontea

Ontea [Laclavik et al. 2007, 2009] identifie des objets dans le texte avec leurs propriétés ou leur localisation en appliquant des patrons sur le texte. Ontea est une plateforme qui admet en entrée du texte non structuré (texte brut) et des patrons, et génère en sortie des paires « attribut-valeur » (appelé aussi : classe-individu, individu-propriété, clé-valeur) qui peuvent être transformées en triplets RDF. Ontea est une plateforme qui permet, d'une part, d'identifier dans le texte semi-automatiquement des instances de concepts d'une ontologie, et d'autre part, de peupler automatiquement une ontologie avec des instances identifiées dans le texte.

L'exemple suivant illustre ce que nous pouvons générer avec Ontea :

- Etant donnés le texte:

  Bratislava is the capital of Slovakia.

  Slovakia is in Europe.

- Avec un patron permettant d'identifier une « Location » : *"(in\by) + (the)? *([A-Z][a-z]+)"*

- Le résultat « attribut-valeur » est : *Location – Europe*

Sachant que dans l'ontologie nous avons « *Europe* est un *Continent* qui est une *Location*», nous obtenons le résultat « attribut-valeur » suivant: *Continent – Europe*

La figure suivante, montre d'autres exemples de ce que Ontea peut identifier dans le texte:

#	**Texte**	**Clé-valeur**	**patrons -expressions régulières**
1	Apple, Inc.	Company: Apple	*Company:* ([A-Za-z0-9]+)[,]+(Inc|Ltd)
2	Mountain View, CA 94043	Settlement: Mountain View	*Settlement:* ([A-Z][a-z]+[ ]*[A-Za-z]*)[ ]+[A-Z]{2}[ ]*[0-9]{5}
3	laclavik.ui@savba.sk	Email: laclavik.ui@savba.sk	*Email:* [-_.a-z0-9]+@[-_.a-zA-Z0-9]+\.[a-z]{2,8}
4	Mr. Michal Laclavik	Person: Michal Laclavik	*Person:* (Mr.|Mrs.|Dr.) ([A-Z][a-z]+ [A-Z][a-z]+)

Figure 1 Exemples d'extractions d'entités nommées par Ontea

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

Nous pouvons résumer les principales caractéristiques d'Ontea en ce qui suit :

- Ontea extrait semi-automatiquement des paires clé-valeur, que nous pouvons considérer comme des entités nommées avec certaines propriétés ;
- Ontea utilise une approche basée sur les patrons construits manuellement pour une langue donnée ;
- Le peuplement d'ontologie par les instances de classes identifiées est automatique ;

Nous avons identifié une caractéristique d'Ontea que nous jugeons, de notre point de vue, limitative : Ontea propose des extractions orientées classe/instance et ne va pas jusqu'à identifier les relations entre instances dans le texte.

Par ailleurs, nous soulignons qu'Ontea se veut une plateforme permettant des extensions possibles, comme l'intégration d'autres types de patrons (par exemple ceux utilisés dans la plateforme GATE) ou l'intégration de XPath comme moyen de construire des patrons pour les documents balisés (XML).

2.4 CAMELEON

CAMELEON [Séguéla et al. 1999] est à la fois un système et une méthode de gestion et de réutilisation de bases de marqueurs pour la génération de relations sémantiques. Selon les auteurs, « *un marqueur est une formule linguistique qui atteste de façon plus ou moins stable l'expression d'une relation sémantique entre deux termes* ». D'une autre manière, dans CAMELEON un marqueur est défini comme un patron lexico-syntaxique désignant dans le discours une relation entre deux termes. Ce marqueur utilise trois éléments : une relation, un identifiant et le schéma permettant l'extraction. L'exemple suivant, illustre un marqueur défini dans CAMELEON :

Relation : HYPONOMIE

Identifiant : Y_EST_LE_X_LE_PLUS

Schéma : Y ETRE ()*mots ARTICLE_DEFINI X ARTICLE_DEFINI (plus|moins)

CAMELEON propose par défaut des marqueurs génériques utilisant des connaissances linguistiques et offre la possibilité de générer des marqueurs spécifiques pour un domaine particulier. Les relations traitées sont la méronymie et l'hyponymie.

3 Approches d'extraction d'information utilisant le contexte

Dans cette section, nous aborderons des travaux d'extraction d'informations à partir de texte qui ont utilisé d'une manière ou d'une autre une notion de contexte.

3.1 Extraction de termes

Dans [Maynard et al. 1999], les auteurs proposent d'extraire des termes en prenant en compte leur contexte d'apparition. En premier lieu, des termes candidats (mots) sont extraits et leurs contextes sont identifiés. Le contexte d'un terme candidat est défini par un ensemble de mots. Les mots du contexte associé à un terme candidat sont identifiés en utilisant :

- Des informations linguistiques comme un filtre syntaxique qui limite les mots d'un contexte à des noms, adjectifs et verbes ;
- Des poids attribués aux mots du contexte suivant leur fréquence d'apparition avec le terme candidat et suivant le type des mots du contexte, si un mot du contexte est lui-même un terme alors il est plus important qu'un simple mot du contexte.

En plus des informations linguistiques et des poids attribués aux mots du contexte, une mesure de similarité est calculée entre un terme candidat et les termes de son contexte. Cette mesure de similarité est calculée en utilisant les distances ontologiques entre termes d'UMLS [Humphreys et al. 1993] (un thésaurus d'un domaine spécifique « biomédical »).

L'identification du contexte des termes candidats permet ainsi de valider les termes candidats comme termes en les classant suivant un ordre d'importance et de choisir les plus pertinents.

Nous soulignons que le « contexte d'un terme » considéré dans cette approche est un ensemble de mots ou de termes qui entourent le terme en question.

3.2 Méthode d'exploration contextuelle

La méthode d'exploration contextuelle [Desclés et al. 1994] s'appuie dans son analyse sur une hiérarchisation des connaissances linguistiques. A cet effet, quatre niveaux se distinguent d'après les auteurs:

1) les connaissances linguistiques : grammaticales et lexicales ;
2) les connaissances du domaine : les savoir-faire liés au domaine de compétence et les règles qui organisent ce domaine ;
3) les connaissances socioculturelles : elles dépendent de l'environnement social, des usages, des coutumes, etc ;
4) les connaissances encyclopédiques : elles sont générales et communes à une communauté donnée.

L'exploration contextuelle fait appel aux connaissances linguistiques en premier lieu et avant toute autre connaissance. L'utilisation des autres connaissances dépend de la complexité du domaine.

L'idée principale de cette méthode est de construire des règles heuristiques qui identifient en premier lieu un indicateur pertinent. Ensuite, des informations grammaticales et lexicales (appelées contexte linguistique) de la phrase où est identifié l'indicateur pertinent, sont utilisées pour associer une étiquette sémantique adéquate à la phrase.

Prenons un exemple simple : dans la phrase « **il est important de souligner** que les…» l'indicateur pertinent est écrit en « gras » et l'étiquette sémantique associé à la phrase est « *soulignement* ».

Les règles heuristiques sont construites (instanciées) à partir d'un schéma général modélisant les connaissances linguistiques et facilitant la génération des règles.

Dans cette méthode, la notion de contexte correspond au contexte linguistique décrit par des règles et qui n'est autre que la disposition grammaticale et lexicale dans une phrase.

Nous aborderons dans ce qui suit deux travaux utilisant cette méthode pour extraire des annotations.

3.2.1 VIGITEXT

Dans le processus d'intelligence économique, les sources d'informations les plus importantes sont les références de brevets, qui contiennent la description des résultats de recherches. Plusieurs outils sont utilisés pour analyser ce genre de ressources. Le principe de la méthode proposée dans [Goujon, 1999a] est le suivant: pour décrire des innovations, certains concepts (appellation selon l'auteur) sont identifiés, le rédacteur insiste sur ce qui est *amélioré*, sur les nouvelles *applications* ou les *utilisations* possibles et sur des *modifications* ajoutées ou obtenues. Ces concepts ainsi que d'autres, sont exploités à partir des résumés de brevets pour localiser une information intéressante.

Cette méthode est basée sur l'identification des concepts indépendamment du domaine du texte, par exemple /improvement/ (avec les indicateurs linguistiques comme : improve, enhance, amelioration,…), /modification/ (transform, alteration, modify,…), etc. Ces concepts, employés souvent dans les résumés de brevets pour décrire des innovations, semblent fournir une information intéressante pour un expert en matière d'intelligence économique.

Les concepts utilisés sont au nombre de 12 et sont organisés dans deux ensembles : *i)* un ensemble cohérent qui explique un changement (avec le concept général */change/* et quatre sous-concepts */improvement/*, */deterioration/*, */increase/* et */decrease/* ; *ii)* et un ensemble de concepts divers représentant un résultat (*/production/*, */resistance/*, */application/*, */control/*, */identification/*, */effect caused/* et */destruction/*). Les auteurs proposent l'outil VIGITEXT qui permet d'identifier les concepts et de les associer à leurs contextes, en utilisant la méthode d'exploration contextuelle basée sur des règles. Ce qui est appelée par les auteurs « contexte » n'est autre que l'étiquette sémantique associée à une phrase. Prenons l'exemple suivant :

Exemple de Règle [Goujon, 1999a]:

- *Conditions*

  L1 := { protect, protects, protecting }, L2 := { against, from }

  Il y a des *indicateurs* d'éléments de L1, et des *indices* d'éléments de L2 tels que : DistanceInWords : (*indicateur,indice*) < 6 et

  Position (*indicateur*) < Position (*indice*)

- *Actions*

  IdentifyBeginOfCIP(RelevantWordAfter (indice))

  CreateExtractAfter (indicateur).

En appliquant cette règle sur la phrase suivante: (1) « on the rind **protects** citrus fruit **against** freezing and chilling. », nous obtenons:

(1) => /resistance/ - « ***protects*** citrus fruit ***against*** freezing and chilling »

Pour obtenir ces résultats, une étude est faite au préalable sur un corpus de 30 résumés de brevets en anglais concernant les plantes transgéniques [Goujon, 1999b].

3.2.2 SERAPHIN

Ce système propose de construire des résumés de manière automatique [Berri, 96], et ce en utilisant la méthode d'exploitation contextuelle. Ce système se veut non lié à un domaine particulier puisqu'il est basé sur une méthode purement linguistique.

Ce système s'intéresse à extraire les phrases (ou parties de phrases) les plus « importantes » : les phrases qui sont les plus informatives, les plus indicatives et qui expriment le message central du document source.

Avec la méthode d'exploration contextuelle, SERAPHIN permet d'avoir un ensemble de phrases étiquetées sémantiquement et offre la possibilité de spécifier une stratégie pour pouvoir sélectionner les phrase pertinentes afin de construire le résumé. Les étiquettes sémantiques étant structurées hiérarchiquement (selon la structure du texte) il est possible ainsi de mettre des priorités pour sélectionner certaines phrases. Par exemple, si le résumé demandé doit être court, les phrases prioritaires seront celles de l'introduction et de la conclusion.

3.3 Le contexte de citation

Dans [Abrouk, 2006], l'auteur propose d'annoter un document en utilisant les documents où il a été cité. Cette approche permet d'annoter des documents sans connaissance préalable de leur contenu en utilisant un regroupement thématique de leurs citations dans d'autres documents. L'auteur utilise une méthode dite de classifieur flou non supervisée. Cette approche est motivée par l'absence d'accès au contenu du document pour des causes de confidentialité.

L'idée principale est d'utiliser les concepts qui annotent des documents non confidentiels pour annoter les documents confidentiels, à condition que ces derniers soient cités par les premiers et que tous les documents partagent une ontologie commune. Un outil est développé pour mettre en œuvre l'approche et est validé sur une grande base de documents techniques.

4 Approches d'extraction d'information basées sur les ontologies

Dans cette section, nous aborderons un état de l'art des approches EIBO « Extraction d'Information Basée sur les Ontologies ». Ces approches concernent notre centre d'intérêt puisque nos travaux s'inscrivent dans ce cadre. Mais avant d'aborder ce point, nous mettrons en évidence certains termes employés dans ce cadre, qui tournent autour des « textes » et des «ontologies ».

4.1 Généralité

L'articulation entre textes et ontologies peut être multiple. En effet, en vue de traitements automatiques plus efficaces, les textes peuvent être considérés comme sources de connaissances pour enrichir les ontologies et réciproquement.

4.1.1 Des textes vers les ontologies

Nous pouvons résumer l'enrichissement de l'ontologie à partie de texte en deux points :

- Les ontologies peuvent être construites en se basant sur le texte comme source de connaissance. Dans ce cas, le terme mentionné dans la littérature est souvent « ontology learning » (apprentissage d'ontologie) où les approches proposées cherchent à automatiser le plus possible ce procédé en se basant souvent sur le traitement automatique de la langue naturelle et sur des connaissances linguistiques.

- Les textes peuvent aussi contenir, par exemple, des instances de concepts permettent d'enrichir une ontologie existante. Dans ce cas, le terme mentionné dans la littérature est souvent « ontology population » (peuplement d'ontologie). Dans ce cadre, il s'agit d'un typage d'instances de concepts. Les travaux utilisant le terme « peuplement d'ontologie » pour le « typage d'instances » considèrent que ces informations extraites font partie de l'ontologie elle-même et non d'une base de connaissance.

4.1.2 Des ontologies vers les textes

L'apport des ontologies pour les textes correspond aux annotations sémantiques (semantic annotation ou knowledge mark-up). Il s'agit de caractériser le contenu informationnel à l'aide d'une ontologie ou d'une base de connaissances. D'une manière plus simple, ceci correspond à faire un étiquetage sémantique du texte ou de portions du texte à l'aide d'instances de concepts ou avec des relations les reliant. Cet étiquetage suit un ou plusieurs schémas d'annotations définis par une (ou plusieurs) ontologie(s) correspondant aux tâches pour lesquelles cette annotation sémantique est construite.

Nous nous intéressons plus particulièrement aux travaux qui traitent le deuxième point, à savoir, l'utilisation des ontologies pour annoter des textes, toutefois, nous aborderons aussi les travaux sur le peuplement d'ontologie puisque ces travaux extraient, entre autres, des instances de concepts.

Nous avons constaté une confusion des termes employés dans certain travaux, entre le *peuplement d'ontologie* et l'*annotation du texte basée sur les ontologies*, due souvent au fait que les deux approches peuvent amener à extraire des instances de concepts et/ou des instances de relation. Pour éviter cette confusion, nous avons choisi d'employer le terme « peuplement d'ontologie » pour les travaux qui extraient des instances de concepts sans instances de relations. Ces extractions sont perçues, d'un point de vue formel, comme une identification d'instances typées. Nous appellerons *approche d'annotation basée sur les ontologies*, toute approche allant jusqu'à extraire des instances typées en les reliant les unes aux autres suivant les contraintes de l'ontologie. Pour faire partie de ces approches, par exemple, les travaux qui extraient des annotations représentées par des triplets RDF doivent identifier le sujet et l'objet d'une propriété donnée dans le texte. Ainsi les triplets construits ne se limitent pas à des triplets de typage.

L'appellation « approche ou système EIBO » est souvent utilisée pour désigner les approches qui englobent les travaux sur « *peuplement d'ontologie* » et sur « *annotation du texte basée sur les ontologies* ».

4.2 Architecture commune des EIBO

La majorité des approches d'extraction d'information à partir de texte basées sur les ontologies partagent une architecture commune composée des modules suivants :

- Générateur d'ontologies : ce module permet d'automatiser le plus possible la génération des ontologies en se basant sur des textes et/ou d'autres ressources ;

- Editeur d'ontologies : ce module permet de donner la possibilité à l'expert du domaine de valider et/ou éditer les ontologies ;
- Lexique sémantique d'une langue : ce module est souvent utilisé pour une langue donnée et soit est considéré comme une ontologie par certains travaux, soit est utilisé comme support de génération pour des ontologies. Un des exemples les plus connus de ce lexique est WordNet [Miller, 1990].
- Module de prétraitement : ce module permet d'adapter les textes au type d'entrée du module d'extraction ;
- Extracteurs d'information : ce module a pour principale fonctionnalité d'alimenter la base de connaissances (ou base de données) par les informations extraites du texte en se basant sur des ontologies et/ou les lexiques sémantiques. L'intervention de l'expert du domaine est souvent permise soit pour orienter l'extraction ou valider les informations extraites.

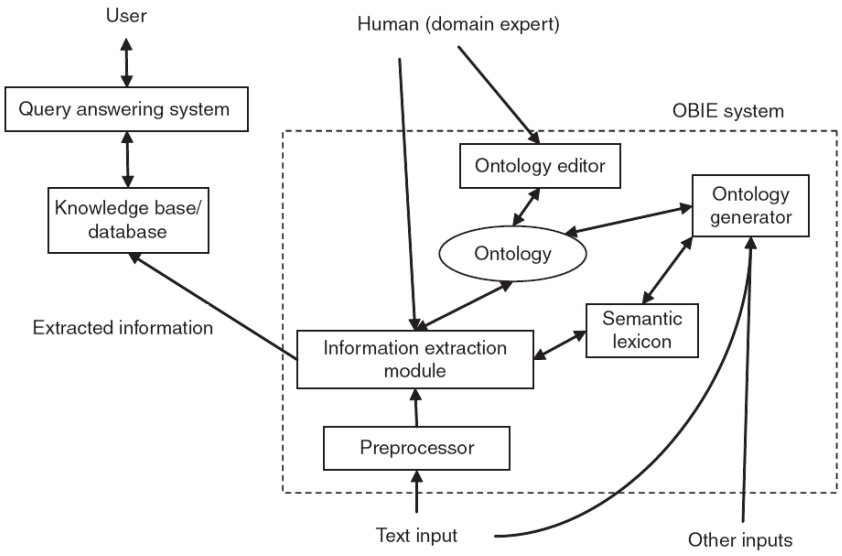


Figure 2 Architecture générale des systèmes EIBO [Wimalasuriya et al. 2010]

La Figure 2 montre une vue globale des différents composants et modules d'un système EIBO. Nous soulignons que l'interaction avec l'utilisateur pour offrir un support d'interrogation de la base d'annotation n'est pas considérée comme faisant partie d'un système EIBO;

Nous allons dans ce qui suit, voir quelques travaux qui ont proposé un système EIBO. Nous nous sommes focalisés sur les approches ayant une ou plusieurs des particularités suivantes :

- Les systèmes EIBO allant jusqu'à extraire des instances (associées à leur concept) et des propriétés avec leur valeur, ou encore ceux qui extraient des triplets RDF;
- Les systèmes EIBO basés sur des patrons/règles ou des expressions régulières et nous nous intéresserons moins aux approches probabilistes ;
- Les systèmes EIBO ayant utilisé le contexte des entités extraites, ou ayant exploité la structure du texte (titre, paragraphe,…) ;
- Les systèmes EIBO ayant une automatisation du processus d'extraction.

4.3 OntoSeek

OntoSeek [Guarino et al. 1999] est un système de recherche documentaire en ligne, conçu spécialement pour les «catalogues de produit », ou les pages ‘jaunes’. La motivation principale de ce travail est la masse importante des résultats proposés pour les usagers, par les méthodes de recherches classiques. Les auteurs proposent de décrire le contenu des documents par une ontologie linguistique généraliste. Les auteurs argumentent qu’une telle approche peut rencontrer quelques problèmes :

- le changement continu des documents exige une évolution de l’ontologie. Or il est difficile de maintenir à jour une ontologie avec la même fréquence de mise à jour que celle des documents ;
- le processus de formulation des requêtes dépend d’un ensemble de termes (ou concepts) rigide ;
- la taille du vocabulaire et les descriptions hétérogènes requièrent une ontologie avec une large couverture.

Comme solution, l’auteur utilise l’ontologie Sensus : 50000 concepts issus de la fusion de l’ontologie linguistique (psycholinguistique) WordNet [Miller, 1990] et de l’ontologie Penman [Bateman, 1990]. Les annotations générées à partir de texte sont représentées sous forme de graphes, en utilisant le formalisme des graphes conceptuels [Sowa, 84]. Le système OntoSeek utilise l’opération de subsomption, pour inférer sur les annotations.

Différentes techniques de recherche d’informations ont été testées dans OntoSeek pour montrer l’intérêt d’utiliser une ontologie linguistique couplée avec une description du contenu structurée afin d’améliorer la recherche d’information pour les pages jaunes. Cependant, d’autres problèmes ont été soulevés au cours de l’élaboration du système OntoSeek, tels que la généricité de l’ontologie qui entraîne des ambiguïtés dans les relations et les concepts.

Nous soulignons qu’OntoSeek est un système fortement lié à un type particulier de texte, à savoir, « les pages jaunes », ce qui limite son exploitation sur d’autres types de texte.

4.4 Exemple sur l’indexation de contenu textuel

Les objectifs liés à l’indexation automatique du contenu textuel ne diffèrent pas beaucoup de ceux de l’annotation à partir de texte. En effet, la différence entre ces deux problématiques réside principalement, d’une part, dans le fait que la génération des annotations est guidée par un modèle déjà prédéfini du domaine (i.e. l’ontologie), et d’autre part, dans les motivations initiales vis-à-vis des documents (ou texte) : par exemple, une représentation formelle du contenue sémantique du texte dans le cas de l’annotation. Parmi les motivations des deux problématiques définies par [Euzenat, 2005] pour les documents et les représentations formelles de leur contenu, nous relevons la différence principale entre indexation et annotation par les deux définitions suivantes:

- L’annotation qui, à partir d’un ensemble de documents et de représentations formelles, engendre une fonction des premiers vers les seconds permettant de retrouver un contenu formel à partir des documents ;
- L’indexation qui, à partir d’un ensemble de documents et de représentations formelles, engendre une fonction des seconds vers les premiers permettant de retrouver les documents à partir des représentations ;

Par ailleurs, des travaux proposent d'exploiter l'apport des ontologies pour une meilleure indexation, nous citons à titre d'exemple les travaux de Desmontils [Desmontils et al. 2002a, 2002b] qui propose une approche pour indexer des ressources Web basée sur le contenu des pages Web. Le but est de construire un index associé à une ontologie. Les étapes de cette approche sont les suivantes (figure 2):

- La première étape consiste à extraire des termes candidats pour des pages Web en utilisant des analyses linguistiques (patrons morpho-syntaxiques). Ces termes sont pondérés par rapport à leur fréquence dans la page et leur importance dans le document (i.e. un titre est plus important qu'un paragraphe). A partir de ces termes pondérés, un index associé à chaque page est construit.
- Ensuite, en utilisant WordNet [Miller, 1990], certain termes précédemment identifiés sont proposés comme « concepts candidats ».
- Des concepts représentatifs de chaque page sont sélectionnés parmi les « concepts candidats ». La sélection dépend d'une part de leur fréquence d'apparition, et d'autre part, de leur relation avec d'autres concepts. Ceci permet une représentativité qui favorise, dans un document, les concepts qui sont en étroite relation avec d'autres concepts plus que ceux qui sont isolés. Ceci est valide même si ces derniers ont une fréquence d'apparition plus importante. Les relations étroites entre concepts que nous citons sont des relations de proximité dans le texte.
- Les concepts sélectionnés pour être représentatifs de leur page Web sont associés aux concepts d'une ontologie du domaine. Un processus de désambiguïsation est appliqué et est basé sur les labels de l'ontologie du domaine.
- La dernière étape est de construire un index structuré en utilisant d'une part les concepts représentatifs des pages, et d'autre part, la hiérarchie entre concepts dans l'ontologie du domaine.

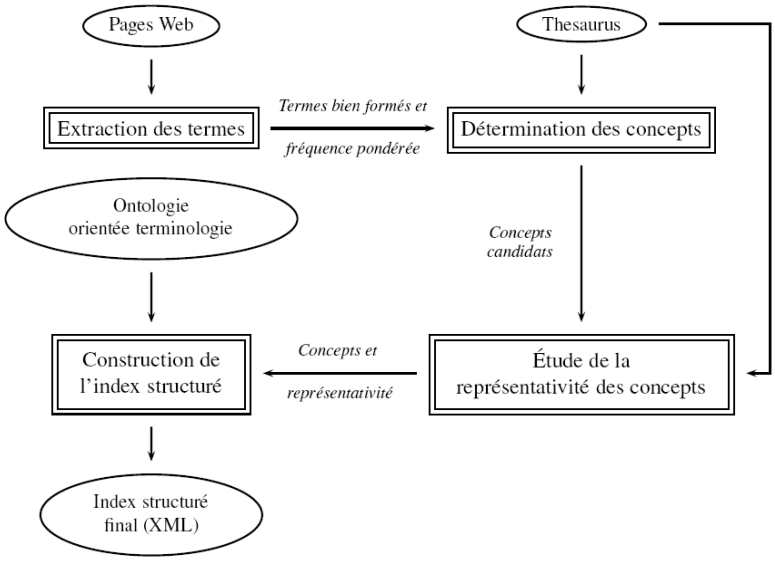


Figure 3 Processus d'indexation dans [Desmontils et al. 2002]

L'approche est semi-automatique et l'utilisateur pourra superviser toutes les étapes du processus. Ainsi, le processus permet d'avoir une vue globale du site web et facilite la recherche d'informations.

4.5 Kylin

Les textes d'entrée du système Kylin [Wu et al. 2007], sont des pages Web de l'encyclopédie Wikipédia. Kylin s'intéresse plus particulièrement aux « infoboxes » des pages de Wikipédia. Les « infoboxes » sont des tableaux d'attributs/valeurs résumant une page Wikipédia. Par exemple, pour une page Wikipédia sur une ville donnée, un tableau contient des attributs tels que: la population, la superficie, etc.

L'idée principale de Kylin est d'utiliser les « infoboxes » existant dans une méthode d'apprentissage pour extraire d'autres informations (autres attributs/valeurs).

L'architecture de Kylin se compose de trois modules : un module de prétraitement, un module de classification et un module d'extraction.

- Le module de prétraitement raffine les « infoboxes » en sélectionnant les attributs les plus importants. Ce module génère un ensemble de données (dataset) pour l'apprentissage automatique du système d'extraction.
- Le module de classification contient deux types de classifieur. Le premier permet de prédire si un article Wikipédia appartient à une catégorie donnée. Le second classifieur permet de prédire si une phrase donnée contient la valeur d'un attribut donné.
- Le module d'extraction permet d'identifier les valeurs d'attributs dans les phrases.

Par ailleurs, Kylin peut être considéré comme un système EIBO puisqu'il construit une ontologie à partir des « infoboxes ». Afin de construire cette ontologie, les titres (i.e le nom d'une ville) des « infoboxes » sont associés aux concepts de WordNet. A partir de cette association, des relations de type « est un » entre les « infoboxes » sont déduites. Par la suite, d'autres correspondances sont identifiées entre les attributs des « infoboxes » avec les attributs de l'infobox « fils » ou « parent » à l'aide de la relation « est un » déjà identifiée entre « infoboxes ».

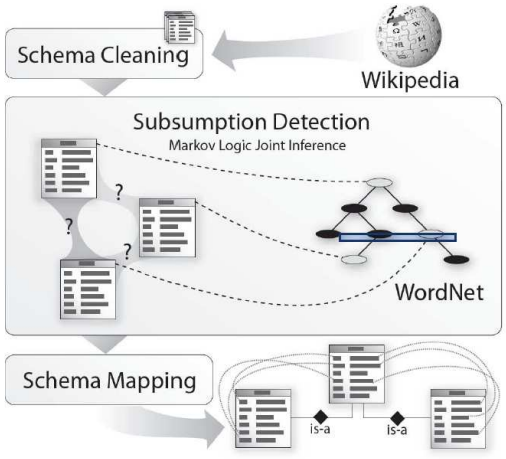


Figure 4 Générateur d'ontologie de Kylin [Weld et al. 2008]

Les informations extraites par Kylin ne sont pas faites pour être utilisées directement dans une base de connaissances, mais elles sont utilisées dans le cadre d'un système de correction pour Wikipédia, qui permet aux utilisateurs de vérifier et/ou corriger les pages [Wu et al. 2007].

L'utilisation par Kylin des modèles de structure (infoboxes) des pages Web de Wikipédia, restreint cette approche à des structures prédéfinies et spécifiques à un type particulier de texte « pages de Wikipédia ». En outre, les informations extraites ne sont pas utilisables

directement et requièrent une intervention humaine pour la validation et/ou la correction. Ceci nous permet de conclure que Kylin est un système d'extraction d'information semi-automatique où l'intervention humaine est fortement requise.

4.6 SOBA

Soba [Buitelaar et al. 2006, 2008] utilise la liste des instances des concepts de l'ontologie comme une liste d'entités nommées « gazetteer » pour pouvoir les identifier dans le texte. Soba permet d'extraire des instances et de les associer à leurs concepts dans l'ontologie et d'extraire aussi des valeurs de propriétés. L'originalité de ce système est qu'il permet d'extraire ces informations à partir de sources diverses dans une page web: tableau, texte et titre des images.

Soba est basé sur des règles linguistiques pour extraire des entités nommées telles que : les noms des personnes, les lieux, les valeurs numériques et les expressions des dates. De plus, Soba se base sur d'autres règles spécifiques aux types de données traitées (les rapports et images des matchs de la coupe du monde de football de 2006).

Le texte d'entrée de Soba se limite à des textes balisés (HTML). Suivant la classification de [Kshitija et al. 2008], nous pouvons placer Soba dans les systèmes liés aux données textuelles semi-structurées (les pages web sans structure prédéfinie).

Soba extrait des instances de concepts (i.e. les noms des Pays qui s'affrontent), et des valeurs de propriété (i.e la date du mach). Toutefois, Soba dépend de règles spécifiques au domaine d'application, ce qui limite son utilisation à un domaine particulier.

4.7 PANKOW

Cimiano et ses collègues ont mis en place dans un système d'extraction d'information basé sur les ontologies, appelé Pankow[7] (annotation basée sur des patrons et des connaissances sur le Web), qui annote sémantiquement une page Web donnée en utilisant les résultats de recherches sur le Web [Cimiano et al. 2004]. Pankow est un système non supervisé et est basé sur la masse d'informations du Web.

Le noyau de Pankow est un mécanisme de génération de patrons permettant d'identifier des chaînes de caractères dans le texte. Ce noyau comporte aussi des schémas de patrons facilitant la construction d'autres patrons qui identifient les instances ou les concepts d'une ontologie donnée.

Pankow prend des pages web (ou un site Web donné) en entrée. Pour chaque page, Pankow identifie les noms propres candidats tels que « *Nelson Mandela*, *South Africa* ». Ces derniers sont utilisés ainsi que les concepts d'une ontologie pour identifier « des phrases hypothèses ». Par exemple, le nom propre « *South Africa* » est combiné avec les concepts « *Country* » et « *Hotel* » dans un patron linguistique pour avoir les deux phrases hypothèses suivantes : « *South Africa is a country* » et « *South Africa is a hotel* ».

Ensuite, le moteur de recherche Google est interrogé sur les phrases hypothèses via l'API de service Web de Google. Un ensemble de résultats est retourné pour chaque phrase hypothèse. Pankow calcule le total des résultats pour chaque phrase hypothèse et celle qui a le total le plus élevé sera retenue. Ainsi, des paires instance-concept sont construites pour chaque page.

[7] Pattern-based Annotation through Knowledge on the Web

Nous voulons mettre l'accent sur une amélioration de Pankow avec le système C-Pankow [Cimiano et al. 2005] qui fonctionne sur les mêmes principes, mais améliore les performances en tenant compte du contexte. Par exemple, le terme « Niger », est souvent affecté au concept « Pays » et non aux concepts « Rivière », « Etat » ou « Région ». C-Pankow propose de résoudre ce problème d'ambigüité en prenant en compte le contexte d'apparition de l'instance (ou de l'entité nommée).

Par ailleurs, il faut souligner que c'est un des rares travaux qui prennent en compte la localisation des entités par rapport à d'autres dans le texte pour apporter des solutions sémantiques. Cependant, les annotations générées par ce système se limitent, comme celui de Pankow, à l'identification des entités nommées (instances), et il ne va pas jusqu'à proposer d'identifier les propriétés avec leur valeur.

4.8 iDocument

Dans le système *iDocument* [Adrian et al. 2009c], l'utilisateur exprime une question sur une collection de textes en utilisant une ontologie du domaine. La requête est formulée avec SPARQL.

L'architecture d'*iDocument* comprend cinq éléments (Figure 5) : l'ontologie du domaine, une collection de textes (texte brut), des requêtes SPARQL, le pipeline d'extraction basé sur l'ontologie et les résultats de requêtes.

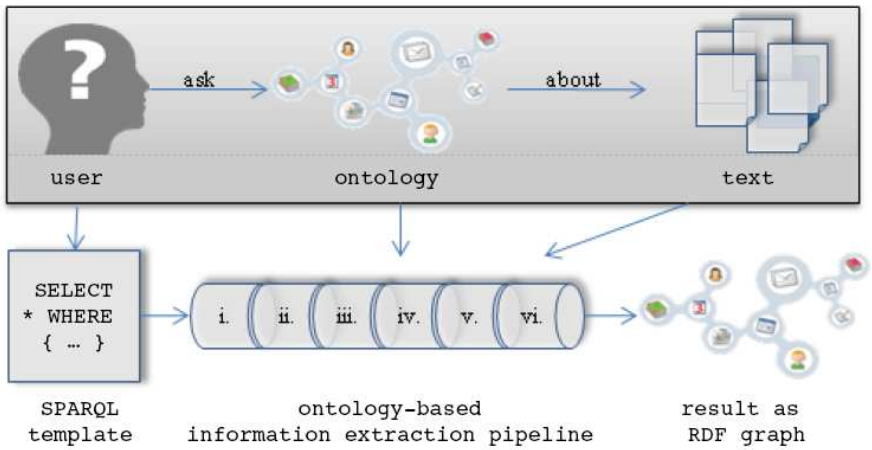


Figure 5 L'architecture d'iDocument [Adrian et al. 2009a]

L'ontologie[8] du domaine est écrite suivant les standards du web sémantique (RDFS ou OWL).

La collection de textes est supposée avoir un contenu pertinent pour répondre aux requêtes. *iDocument* étant basé sur le framework Aperture[9], il supporte les formats de documents les plus courants (PDF, DOC, HTML,…).

L'utilisateur définit une requête qui déclenche le processus d'extraction d'informations et la génération d'annotations en lien avec les arguments utilisés dans la requête. La requête est

[8] http://ontologies.opendfki.de/

[9] Aperture est un framework open source sous Java qui extrait les données et les métadonnées des documents dans un vocabulaire standard, http://aperture.sourceforge.net/

considérée comme un template d'extraction d'informations. Ce template peut être défini avec SPARQL. Par exemple :

```
select * where { ?p  rdf:type  foaf:Person ;    foaf:member ?o .
?o  rdf:type  foaf:Organization }
```

Le pipeline d'extraction se résume en six tâches : i) normalisation ; ii) segmentation ; iii) symbolisation ; iv) instanciation ; v) contextualisation et vi) peuplement. Ce pipeline est implanté en utilisant une approche probabiliste *Believing Finite-state Cascades* [Adrian, al. 2008]. Dans les processus traditionnels de parcours des tokens (les éléments textuels élémentaires dans un processus de TALN : mots, ponctuations, etc.), le token est associé ou non (notion binaire) à une entité nommée. L'approche probabiliste qu'utilise iDocument introduit les probabilités (notion non binaire) en prenant en compte les tokens précédant le token en cours de parcours. D'une manière plus simple, le token est associé à une entité avec une probabilité qui dépend aussi de la probabilité des tokens précédant celui-ci. De même, chaque tâche du pipeline d'extraction est basée soit sur la collection de textes soit sur les résultats de la tâche précédente et produit des hypothèses pondérées (approche probabiliste).

La *normalisation* transforme le document texte en une représentation RDF constituée de deux éléments. Le premier contient le texte brut et le second contient des annotations sur le document telles que l'auteur, la date et le titre. La tâche *segmentation* partitionne le texte en paragraphes, phrases et tokens. Durant la *symbolisation*, les tokens de type « entité nommée » sont identifiés parmi tous les tokens.

Dans la tâche d'*instanciation,* le reste des tokens est désambiguïsé. Un token est associé à des instances de concepts, à des propriétés candidates ou à des valeurs de propriétés. Cette désambiguïsation est faite à l'aide de la requête SPARQL qui a déclenché ce processus d'extraction. Par exemple, si un token est utilisé dans la requête en tant qu'instance de concept alors il sera considéré comme tel.

La tâche de *contextualisation* a pour but de désambiguïser les tokens encore ambigües en modifiant la requête SPARQL initiale. Par exemple, les concepts de la requête sont remplacés par leurs concepts parents respectifs.

La dernière tâche est le *peuplement* qui consiste à ajouter les instances extraites à l'ontologie du domaine.

Le résultat du pipeline d'extraction est transformé en un graphe RDF qui peut être visualisé et approuvé par l'utilisateur.

iDocument offre une interface utilisateurs pour visualiser et interagir avec les différents modules. Par exemple : navigation dans l'ontologie, navigation sur les différentes requêtes SPARQL, parcourir la liste des entités nommées, etc.

Le système *iDocument* propose aussi dans [Adrian et al. 2009b] un outil pour permettre aux utilisateurs d'accepter ou de rejeter les annotations générées par le système ou encore de construire de nouvelles annotations.

iDocument est une initiative basée sur un modèle probabiliste permettant d'alléger le processus d'extraction d'information et génération d'annotation en ciblant le traitement sur le texte ayant un rapport avec le graphe de la requête formulée par l'utilisateur du système. *iDocument* nécessite, d'une part, une base d'annotations existante, et d'autre part, une forte interaction de l'utilisateur pour valider les annotations.

4.9 MeatAnnot

MeatAnnot [Khelif et al. 2007] fait partie des travaux qui vont plus loin dans la génération des annotations. En effet, les travaux de Khelif qui ont beaucoup inspiré notre travail, proposent d'extraire, en plus des termes candidats (instances de concept), des relations reliant ces instances de concepts. MeatAnnot utilise le méta-thésaurus biomédical UMLS [Humphreys et al. 1993].

Le processus d'extraction d'informations débute par une phase de prétraitement qui consiste à préparer le texte d'entrée.

MeatAnnot identifie les instances de relations d'UMLS dans le texte. L'auteur considère que les relations sont caractérisées par un ensemble de verbes et de syntagmes verbaux. L'apparition d'un de ces syntagmes dans le texte est considérée comme un indice de l'existence d'une instance de relation. Par exemple, la relation « *prevents* » peut être caractérisée par « has a preventive effect » et par « prevents ». Les relations sont identifiées en utilisant des règles de grammaire JAPE construites manuellement.

Les phrases où ces relations sont identifiées sont considérées comme potentiellement porteuses d'arguments de ces relations. Ces phrases sont analysées pour extraire des termes candidats à l'aide de règles linguistiques. Ces termes candidats sont comparés avec les instances de concept d'UMLS en utilisant des requêtes. Si ces termes candidats existent dans UMLS alors ils seront identifiés comme instances de concepts et arguments pour la relation identifiée initialement dans la même phrase. Ainsi les triplets RDF peuvent être construits pour chaque phrase. Le choix du sujet/objet dans un triplet est fait suivant le rôle linguistique identifié pour les termes candidats.

4.10 Autres travaux

D'autres systèmes EIBO s'intéressent exclusivement au peuplement d'ontologie et les annotations générées sont de type « instances de concepts ou attribut-valeur» [Vargas-Vera, et al. 2001] [Declerck, et al. 2008].

Les auteurs de la majorité des systèmes EIBO basés sur des règles linguistiques proposent de construire manuellement ces règles. Cela signifie qu'une personne ou un groupe de personnes ont à lire des documents du corpus et à créer les règles d'identification appropriées. Nous constatons que cet exercice est long et fastidieux et n'est pas adapté à de corpus volumineux. Afin de résoudre ce problème, certains systèmes visent à construire automatiquement les règles d'extraction d'information à partir de texte. Les auteurs de [Vargas-Vera, et al. 2001] ont conçu et mis en œuvre un système EIBO nommé MnM qui fonctionne sur ce principe. Ils ont utilisé un outil d'induction de dictionnaire nommé Crystal [Soderland et al. 1995] pour identifier les règles d'extraction. Cet outil fonctionne sur les principes de l'algorithme d'apprentissage inductif [Mitchell et al. 1982] et recherche la plus spécifique généralisation qui couvre toutes les instances positives. Les instances positives (des mots dans le texte identifiant des instances ou des valeurs des propriétés de l'ontologie) sont identifiées manuellement en utilisant un outil de marquage et sont utilisées comme base d'apprentissage. Par ailleurs, MnM est dédié aux pages Web partageant une thématique commune (i.e sites Web).

D'autres travaux sont encore plus restrictifs puisqu'ils traitent un type particulier de texte. Par exemple le système Vulcain [Todirascu et al. 2002] est dédié au texte des e-mails.

Le système d'Embley [Embley, 2004] est un des premiers systèmes EIBO à combiner des règles linguistiques avec les éléments des ontologies (concepts et propriétés) en utilisant des expressions régulières. C'est aussi un des travaux qui proposent que les règles linguistiques utilisées dans l'extraction d'information soient considérées comme faisant partie de l'ontologie.

Un autre système appelé ontoX [Yildiz et al. 2007] utilise une technique similaire à celle d'Embley et propose en plus des instances, d'extraire des valeurs « *littérales* » de propriétés (i.e. *float* ou *decimal*). Un autre système d'extraction basé sur les règles linguistiques est proposé par [Saggion et al. 2007] et est basé sur l'architecture GATE [Cunningham et al. 2002]. Ce système d'extraction utilise à la fois des règles linguistiques et une liste d'entités nommées pour extraire des instances et des valeurs de propriétés.

D'autres travaux se sont limités à n'extraire que des instances, tels que [Li et al. 2007] qui est basé sur une technique de classification. Dans le système OntoSyphon [McDowell et al. 2006], les auteurs proposent d'extraire des instances en se basant sur les informations redondantes du Web. D'une manière similaire à iDocument [Adrian et al. 2009a], les auteurs d'OntoSyphon proposent d'exploiter les résultats de recherches des requêtes lancées sur des moteurs de recherches dans le Web. OntoSyphon s'appuie sur une ontologie de domaine pour formuler ces requêtes. Les résultats des recherches sont utilisés pour identifier des instances de concepts et les redondances de ces instances sont utilisées pour valider leur correspondance aux concepts adéquats.

5 Comparatif des approches d'extraction d'informations et d'annotation à partir de texte

Après avoir dressé un état de l'art sur les différentes approches d'extraction d'information et d'annotations partir de texte, nous résumons dans ce qui suit les principales caractéristiques de ces approches et nous mettons en évidence certains critères :

- *Automatisation de l'approche :* avec ce critère nous essayons de mettre en évidence les travaux les plus automatisés ;
- *Utilisation de la notion de contexte y compris la structure du texte* : cette caractéristique permet de constater si l'approche utilise la notion de contexte.
- *Type des annotations:* ce critère permet de mettre en évidence les approches qui extraient des annotations sous forme de triplets et qui ne se contentent pas des termes ou instances.
- *Type de source textuelle:* ce critère permet de savoir si l'approche est utilisable pour un type particulier de document (i.e. Wikipedia), pour un domaine particulier (i.e. médical), pour des textes du domaine de l'ontologie, ou sans restriction.

Nous soulignons que le résumé du Tableau 1 n'a pas pour but d'évaluer ces approches, puisque leurs techniques sont aussi différentes que leurs objectifs. Cependant, ce tableau nous permet de motiver notre proposition de nouvelle approche de génération d'annotations.

	Automatisation	Notion de contexte	Type des annotations générées	Type de source textuelle
Approches basées sur les ontologies				
C-PANKOW	Automatique	Partiellement	instances	Pas de restriction
ontoSyphon	Automatique	non	instances	Pas de restriction
SOBA	Automatique	non	Instance, valeurs de propriétés	Page HTML du domaine
Embley	Automatique	non	Instance, valeurs de propriétés	Documents du domaine
Saggion et al.	Automatique	non	Instance, valeurs de propriétés	Documents du domaine
ontoX	Automatique	non	Instance, valeurs *datatype* de propriétés	Documents du domaine
Vulcain	Automatique	non	Instance, valeurs de propriété	Mails du domaine
Vargas-Vera et al.	Automatique	non	Instance, valeurs de propriété	Page Web pour un site particulier
KIM	Semi-automatique	non	Instance, valeurs de propriété	Documents du domaine
iDocument,	Semi-automatique	non	Triplet RDF	Documents du domaine
Kylin	Semi-automatique	non	Classes, instances, valeurs de propriété, taxonomie	Pages de Wikipédia
OntoSeek	automatique	Utilise la structure des pages	Graphes conceptuels	Pages jaunes
MeatAnnot	automatique	non	Triplet RDF	Domaine biomédical
Autres approches d'extraction d'information et annotations				
Ontea	Semi-automatique	non	Triplet RDF	Texte brut
Desmontils et al. 2002	Semi-automatique	non	Concepts	Pages Web

Maynard et al. 1999	Automatique	Partiellement: avec des poids	Termes	Domaine biomédical
VIGITEXT	Automatique	Contexte linguistique	Phrase pertinente marquée	Documents de brevets
CAMELEON	Semi-automatique	Contexte linguistique	Termes, relations particulière	Texte brut

Tableau 1 Classification des travaux d'extraction d'information et annotations à partir de texte

Il est difficile de proposer un système automatique d'extraction d'informations tout en proposant des annotations plus riches sémantiquement. Par exemple, nous pouvons décider de se restreindre à extraire des instances de concepts sans les relations qui les lient pour proposer un système d'extraction entièrement automatique.

Nous avons constaté que les sources textuelles sont aussi impliquées dans cette relation entre l'automatisation du système d'extraction et le type d'annotations générées. En effet, des systèmes automatiques d'extraction sont proposés pour extraire des annotations sémantiques, mais dans un domaine particulier ou pour un type particulier de texte. Par exemple, MeatAnnot permet d'extraire automatiquement des annotations sémantiques sous forme de triplets RDF pour des textes du domaine particulier du biomédical. Un autre exemple est OntoSeek, qui propose d'extraire des annotations sous forme de graphes conceptuels, en se limitant à un type particulier de document : « les pages jaunes ».

Pour les autres approches qui ne sont pas basées sur des ontologies, nous avons constaté les mêmes caractéristiques. En effet, par exemple: Ontea permet d'extraire des triplets RDF mais l'approche est semi-automatique ; les auteurs de VIGITEXT proposent une approche automatisée qui dépend des textes particuliers sur les brevets.

Dans le domaine d'extraction d'information et annotations à partir de texte, il est difficile de cumuler ces trois caractéristiques: i) automatisation du système d'extraction ; ii) génération des annotations sémantiques et iii) ne pas dépendre d'un type particulier de textes.

Nous nous somme intéressés à cette problématique, à savoir, concilier ces trois caractéristiques en se basant sur une simple hypothèse :

Hypothèse

« *L'utilisation du contexte dans l'extraction d'information et l'annotation à partir de texte permet :*

- *d'améliorer la qualité des annotations générées ;*
- *d'alléger l'intervention d'experts humains dans la validation des annotations générées;*
- *de moins dépendre des types de textes traités.* »

Les approches abordées dans l'état de l'art qui mentionnent la notion de contexte font référence, soit au contexte linguistique (la disposition grammaticale et lexicale d'une phrase) tel CAMELEON [Séguéla et al. 1999] et [Desmontils et al. 2002], soit au contexte défini par un ensemble de mots (ou termes) [Maynard et al. 1999] et [Cimiano et al. 2005]. Dans nos travaux, nous essayons d'explorer un point de vue différent du contexte et de son exploitation dans le processus d'annotations.

L'approche que nous proposons est appelée CEEMA[10] : *méthode d'extraction et d'exploitation des annotations contextuelles*. Cette approche sera présentée dans le chapitre qui suit.

La Figure 6 montre la disposition des approches abordées dans l'état de l'art par rapport aux trois caractéristiques citées précédemment. Les approches citées avec un fond gris représentent les approches qui dépendent d'un type particulier de texte.

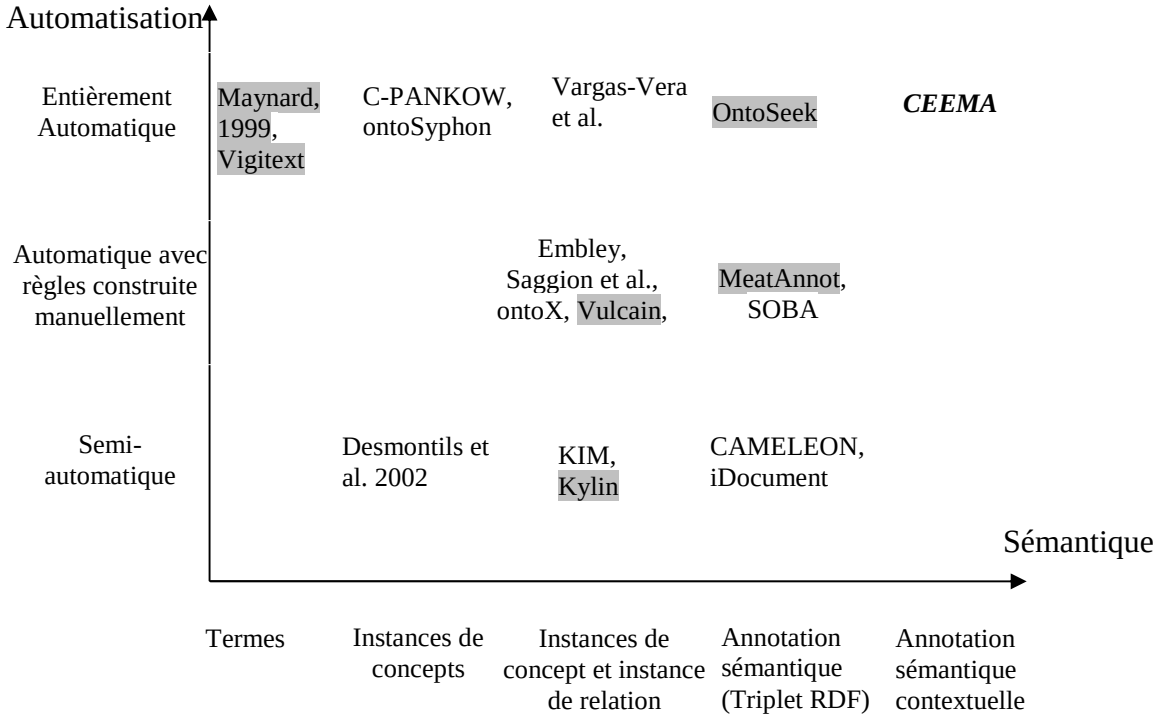


Figure 6 Relation entre *sémantique des annotations*, *automatisation de l'extraction* et *dépendance au type de texte traité*

[10] CEEMA: Contextual Extraction and Exploitation Method of Annotation

6 Conclusion

Dans ce chapitre, nous avons vu plusieurs types d'approches d'extraction d'informations à partir de texte et d'annotations :

- Les plateformes d'annotations, qui servent de support de base pour l'analyse du texte. Ces plateformes offrent un mécanisme de base d'analyse linguistique et/ou sémantique de texte avec la possibilité d'étendre ces mécanismes pour une analyse permettant d'engendrer des annotations plus riche sémantiquement. Nous citons à titre d'exemple la plateforme GATE ;
- Les travaux sur l'extraction d'informations à partir de texte, qui ont exploité la notion de contexte. Nous avons exposé, entre autres, la méthode proposée par Desclés est ses collègues dite « exploration contextuelle » [Desclés et al. 1994]. Cette méthode s'intéresse plus particulièrement à la partie linguistique du contexte. Les diverses approches ayant utilisés la notion de contexte font référence au « contexte » selon leur point de vue. Ceci nous a motivé pour explorer l'utilisation de la notion de « contexte » dans d'autre domaine tel que « l'informatique pervasive » (le chapitre II) ;
- Les travaux sur l'extraction d'informations basés sur les ontologies. Nous nous sommes intéressés plus particulièrement à ce type d'approches qui exploitent la richesse sémantique de l'ontologie pour offrir des annotations sémantiques.

Nous avons ensuite dressé un tableau récapitulatif sur les différentes approches d'extraction d'informations abordées dans ce chapitre. Ce tableau nous a permis de motiver notre proposition d'une nouvelle d'approche d'extraction d'informations à partir de texte « CEEMA ». Cette approche que nous proposons se veut : automatique, ne dépendant pas d'un domaine particulier et les annotations générées prennent en compte le contexte de leur apparition.

Chapitre II : Modélisation du contexte dans le domaine de l'informatique pervasive

« L'informatique omniprésente ou pervasive » fait référence à la tendance à l'informatisation, la connexion en réseau, la miniaturisation des dispositifs électroniques et leur intégration dans tout objet du quotidien, favorisant ainsi l'accès aux informations partout et à tout moment[11]. L'informatique pervasive est un domaine qui a fait une avancée importante dans la modélisation et l'utilisation des informations sur le contexte, ce qui nous a poussé à étudier les différentes propositions dans ce domaine et plus précisément celles qui sont conceptuellement génériques. La classification que nous exposons dans ce qui suit aborde les travaux sur l'utilisation des informations sur le contexte d'un point de vue « modèle ». Nous résumerons aussi les exigences et recommandation exposées dans [Bettini et al.2009] pour avoir une bonne approche modélisant les informations sur le contexte dans l'informatique omniprésente. Nous précisons que les termes cités dans cette partie comme « information sur le contexte », « application contextuelle » sont ceux utilisés dans le domaine de l'informatique omniprésente. Nous ne manquerons pas aussi dans ce chapitre de souligner les aspects importants à prendre en compte lors de notre modélisation du contexte pour des informations de type textuel.

[11] source : Technique de l'ingénieur

1 Introduction

Il existe une quantité croissante de recherches sur l'utilisation du contexte pour le développement d'applications informatique afin de les rendre plus flexibles, adaptables et capables d'agir de manière autonome pour le compte d'utilisateurs. Le développement d'applications qui prennent en compte la notion de contexte « dites applications contextuelles » est par nature complexe. Ces applications s'adaptent à l'évolution des informations venant du contexte : contexte physique, contexte informatique et contexte de la tâche de l'utilisateur.

La communauté informatique pervasive comprend de plus en plus les avantages de la modélisation formelle de l'information du contexte. Tout d'abord, en raison de la complexité inhérente aux applications contextuelles compatibles, le développement doit être soutenu par des méthodes de génie logiciel adéquates. L'objectif global est de développer des applications contextuelles capables d'évolution de manière autonome. Un bon formalisme de modélisation d'information sur le contexte permet de réduire la complexité des applications contextuelles et améliore leur maintenabilité et évolutivité. En outre, la collecte, l'évaluation et la maintenance des informations sur le contexte sont coûteuses. La réutilisation et le partage des informations sur le contexte entre les applications contextuelles devraient être considérés dès le début de la modélisation. L'existence de modèles bien conçus des informations sur le contexte facilite le développement et le déploiement des applications.

Les approches actuelles de modélisation des informations sur le contexte –ou modélisation contextuelle– varient dans la facilité avec laquelle les objets du monde réel peuvent être capturés par les ingénieurs logiciels :

- dans le pouvoir expressif des modèles des informations sur le contexte ;
- dans l'aide qu'elles peuvent fournir pour le raisonnement sur des informations sur le contexte ;
- dans la performance du raisonnement ;
- dans l'évolutivité de la gestion des informations sur le contexte.

Un grand nombre d'applications sensibles au contexte basées sur des modèles de contexte différents ont été développées au fil des années dans divers domaines d'application. Ces expériences influencent l'ensemble des exigences définies pour la modélisation de contexte et du raisonnement, et donc également influencent la recherche sur les modèles d'informations sur le contexte. Nous cherchons à définir des modèles qui ont un haut pouvoir expressif, pouvant supporter le raisonnement sur le contexte, et ont de bonnes performance dans le raisonnement.

2 Exigences sur la bonne approche de modélisation des informations sur le contexte

Nous exposons dans ce qui suit un résumé des exigences définies dans [Bettini et al. 2009] utiles, à notre point de vue, pour les modèles de contexte et leurs systèmes de gestion de contexte.

2.1 L'hétérogénéité et la mobilité

Les modèles d'information sur le contexte doivent être capables de restituer une grande variété de sources d'information sur le contexte qui diffèrent dans leur taux de mise à jour et leur niveau sémantique. Un modèle de contexte devrait être en mesure d'exprimer les différents types d'informations sur le contexte. De plus, le système de gestion de contexte doit assurer la gestion de l'information sur le contexte en fonction de son type. Beaucoup d'applications sensibles au contexte sont aussi mobiles (i.e. en cours d'exécution sur un appareil mobile) ou dépendent de sources d'information mobiles (par exemple, des capteurs mobiles). Cela ajoute au problème de l'hétérogénéité. En effet, l'emplacement et la disposition spatiale des informations sur le contexte jouent un rôle important en raison de l'exigence de mobilité.

Si nous adaptons cette exigence à la modélisation d'informations sur le contexte de type textuel, c'est la capacité de prendre en compte, d'une part, l'hétérogénéité des sources textuelles avec différents formats de texte et indépendamment d'un domaine particulier, et d'autre part, la localisation des informations de type textuel par rapport à d'autres (structure physique du texte).

2.2 Les relations et les dépendances

Il existe des relations entre les différents types d'informations sur le contexte qui doivent être acquises pour assurer un comportement correct des applications. Les relations correspondent à la dépendance par laquelle les entités tirées du contexte peuvent dépendre d'autres entités : par exemple, un changement de la valeur d'une propriété (e.g. la bande passante réseau) peut affecter les valeurs d'autres propriétés (e.g. reste de la puissance de la batterie).

Ce que nous retenons de l'exigence concernant les relations et les dépendances pour les informations de type textuel, c'est de prendre en compte les dépendances qui peuvent exister, d'une part, entre textes (e.g. les liens de citations : tel document est cité par tel document), et d'autre part, entre des textes et des entités non textuelles (e.g. tel document est rédigé par telle personne).

2.3 L'historique des informations sur le contexte

Les applications sensibles au contexte doivent avoir accès à l'état passé et à l'état futur (pronostic). Par conséquent, l'historique des informations sur le contexte est une autre caractéristique qui doit être saisie par les modèles de contexte et gérée par le système de gestion de contexte. La gestion de l'historique des informations sur le contexte est difficile si le nombre de mises à jour est très élevé. Cependant, des techniques de résumé peuvent être appliquées (par exemple, réduire des mises à jour de position à une fonction de mouvement en utilisant des techniques d'interpolation).

Suite à cette exigence, ce que nous retenons pour les informations de type textuel, c'est la gestion de version des textes. Cependant, il est à noter que la gestion de l'historique documentaire requiert un système qui prend en compte tout le cycle de vie du document (ou texte).

2.4 L'imperfection

En raison de leur nature dynamique et hétérogène, les informations sur le contexte peuvent être de qualité variable. En fait, elles peuvent même être incorrectes. La plupart des capteurs

disposent d'une imprécision inhérente (par exemple, à quelques mètres des positions GPS). En outre, les informations sur le contexte peuvent être incomplètes ou contradictoires. Ainsi, une bonne approche de modélisation de contexte doit inclure la modélisation de la qualité des informations pour étayer un raisonnement sur le contexte.

Il est donc important de pouvoir, d'une part, détecter et gérer les informations erronées ou contradictoires, et d'autre part, identifier les incohérences sémantiques et pouvoir les prendre en compte dans le raisonnement.

2.5 Le raisonnement

Les applications contextuelles utilisent les informations sur le contexte pour, d'une part, déterminer s'il y a un changement pour l'utilisateur et/ou pour l'environnement informatique ; et d'autre part, prendre une décision si une adaptation à un changement est nécessaire. Il est donc important que les approches de modélisation soient en mesure de supporter à la fois la vérification de cohérence du modèle et des techniques de raisonnement prenant en compte le contexte. De plus, les techniques informatiques de raisonnement doivent être efficaces.

En faisant une projection sur les informations sur le contexte de type textuel, cette exigence pousse à avoir un mécanisme de raisonnement efficace qui permet d'exploiter au mieux la sémantique dégagée par le texte.

2.6 Approvisionnement efficace des informations sur le contexte

L'accès efficace aux informations sur le contexte est nécessaire, surtout en présence de nombreux objets sources de données. Pour sélectionner les objets pertinents, des attributs doivent être définis au préalable et doivent être considérés dans le modèle. Ceci permet d'orienter les applications contextuelles vers un type d'information sur le contexte prioritaire. Communément les attributs utilisés sont : l'identité des objets de contexte, l'emplacement, le type d'objet, l'heure ou l'activité de l'utilisateur.

Ce que nous retenons pour les informations sur le contexte de type textuel, c'est la nécessité de connaître au préalable les informations à prendre en compte en priorité. Il s'agit de définir les entités textuelles, leur contexte et les attributs utiles à prendre en compte, tels que l'aspect temporel, la position d'un texte par rapport à un autre, ou encore, d'autres aspects jugés importants pour un domaine particulier.

3 Quelques modèles de représentation du contexte

3.1 Les modèles basés sur « clé-valeur » et les modèles de balisage

Les modèles clé-valeur utilisent de simples paires de clé-valeur pour définir la liste des attributs et leurs valeurs décrivant les informations sur le contexte utilisées par les applications contextuelles. Les modèles d'informations sur le contexte basés sur des balises utilisent divers langages de balisage incluant XML.

Le standard du W3C pour la description des appareils mobiles, « *Composite Capabilities/Preference Profile (CC / PP)* » [Klyne et al. 2004], est probablement la première approche de modélisation du contexte à utiliser RDF, « Resource Description Framework », et à inclure des contraintes élémentaires et des relations entre les types de contexte (voir Figure 7). CC/PP peut être considéré comme un représentant à la fois de la catégorie de modèles de

clé-valeur et des modèles de balisage, car il est basé sur la syntaxe RDF pour stocker des paires clé-valeur avec les balises appropriées.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:ex="http://www.example.com/schema#">
 <rdf:Description rdf:about="http://www.example.com/profile#MyProfile">
  <ccpp:component>
    <rdf:Description
         rdf:about="http://www.example.com/profile#TerminalHardware">
     <rdf:type
        rdf:resource="http://www.example.com/schema#HardwarePlatform" />
     <ex:displayWidth>320</ex:displayWidth>
     <ex:displayHeight>200</ex:displayHeight>
    </rdf:Description>
  </ccpp:component>
        <!--  Possibilité d'ajouter d'autres «component» descriptifs -->
 </rdf:Description>
</rdf:RDF>
```

Figure 7 Exemple d'une partie de profil CC/PP en XML

CC/PP ainsi que les autres modèles d'information sur le contexte de clé-valeur et de balisage ont déjà fait l'objet d'une description et d'une évaluation dans la littérature et leurs limites ont été présentées dans [Indulska et al, 2003] et [Strang et al, 2004]. Les principales critiques de ces approches concernent leurs capacités limitées de: (i) refléter la diversité de types de contexte, (ii) représenter les relations, les dépendances, la rapidité et la qualité des informations sur le contexte, (iii) permettre la vérification de cohérence.

En dépit des critiques, les modèles simples peuvent être suffisants pour certains domaines ou types d'information sur le contexte. Dans notre cas, nous cherchons à extraire des annotations à partir de texte, un problème souvent lié à une certaine difficulté, et les annotations générées ne requièrent pas toujours un modèle de représentation avec une expressivité poussée telle que RDF. Cependant, il est plus que nécessaire, de notre point de vue, que ce modèle de représentation admette des solutions d'extensions ou qu'il soit englobé dans un modèle plus expressif, comme l'est OWL pour RDF.

3.2 Les modèles centrés sur le domaine d'application

Il existe un nombre important de travaux dans différents domaines d'application sur les types d'informations sur le contexte qui peuvent améliorer considérablement les fonctionnalités des applications contextuelles dans un domaine particulier. Un exemple pertinent parmi ces

travaux est le modèle de contexte W4[12], et le développement de son infrastructure qui supporte la navigation sensible au contexte [Castelli et al. 2007]. Ce modèle prend en charge la représentation du contexte, comme des tuples « Qui, Quoi, Où, Quand » et fournit une interface pour stocker et interroger ces tuples. Cependant, bien que cette approche soit générique, un effort important est requis pour spécifier « quelles » informations sur le contexte doivent être modélisées étant donné qu'elles varient d'un domaine à un autre.

3.3 Les modèles basés sur objet-rôle

Les premières approches de la modélisation de contexte, représentées par CC/PP et des approches similaires, ne répondent pas à de nombreuses exigences énumérées à « la section 2.1. ». D'autres approches, caractérisées par des outils de modélisation du contexte plus expressives, fournissent de meilleures solutions à quelques unes des exigences identifiées. Les approches de modélisation de contexte basées sur les « Faits », dont l'approche basée sur objet-rôle (ou parfois appelée entité-relation), proviennent de tentatives pour créer des modèles de contexte suffisamment formels pour supporter le traitement des requêtes et de raisonnement, ainsi que de fournir des concepts de modélisation appropriés pour une utilisation dans des tâches de génie logiciel telles que l'analyse et la conception.

Les approches de modélisation de contexte basées sur objet-rôle sont inspirées de la modélisation des systèmes d'information (modèle entité-relation pour les SGBD). Ces approches fournissent une correspondance facile entre les informations du contexte du monde réel et leur conceptualisation. Le meilleur exemple pour ces approches est le langage de modélisation du contexte (CML[13]) [Henricksen et al. 2006]. Cette approche utilise également la logique des prédicats pour raisonner sur des abstractions de contexte de haut niveau, en particulier, pour satisfaire les exigences suivantes : l'hétérogénéité, l'historique du contexte et le raisonnement.

[12] (Who, What, Where, When)

[13] Context Modelling Language

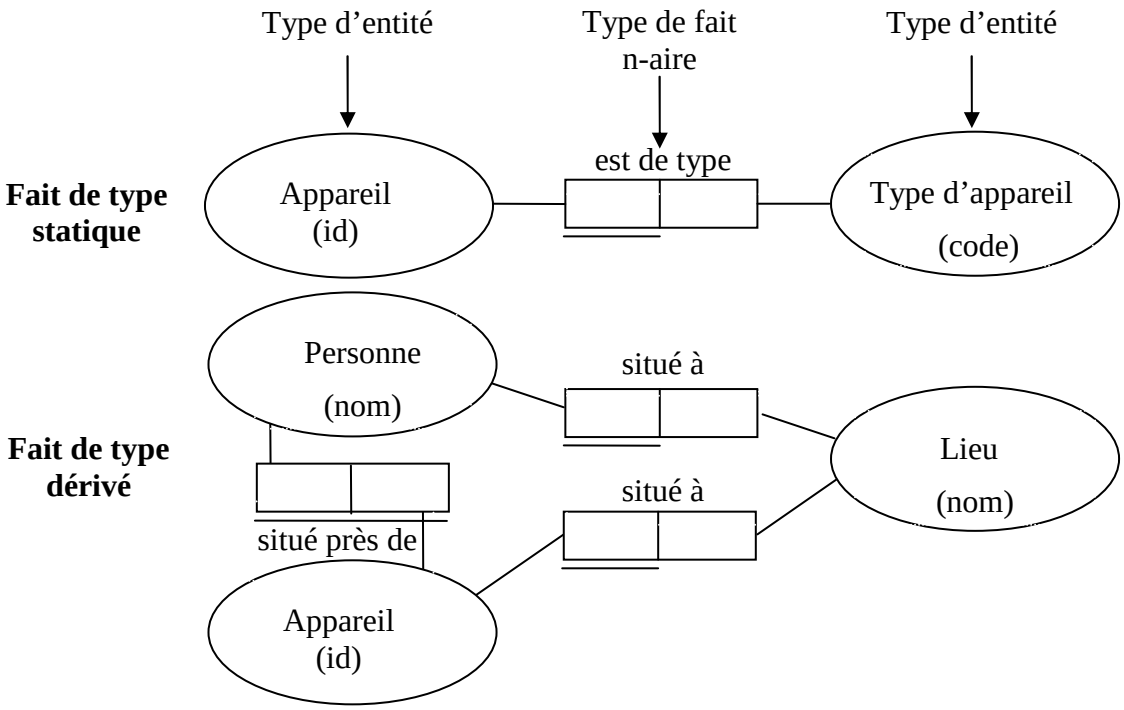


Figure 8 Exemples élémentaires de « faits » en CML

La Figure 8 montre un exemple d'un « fait » statique et un autre dérivé d'autres « faits ». Plusieurs types de « fait » de base sont proposés pour représenter des dépendances plus complexes entre entités, tel l'aspect temporel.

L'une des forces principales de CML est son applicabilité sur différentes étapes du processus de génie logiciel. Sa notation graphique prend en charge l'analyse et la conception des exigences sur les applications contextuelles (section 2.1). Sa représentation relationnelle et sa grammaire pour l'abstraction de haut niveau de contexte supporte les représentations exécutables et l'interrogation. CML fournit également un support plus complet pour la capture et l'évaluation de l'information imparfaite et l'information historique.

Toutefois, CML a plusieurs faiblesses. Il dispose d'un modèle d'information « plat », tous les types de contexte sont uniformément représentés comme des faits atomiques. Si une structure hiérarchique est nécessaire, ou une dimension particulière du contexte est dominante dans la perspective d'interrogation (comme dans les modèles spatiaux, qui accordent une plus grande importance à l'emplacement qu'à d'autres types d'informations), alors d'autres représentations peuvent être plus appropriées que CML.

3.4 Modèle spatial des informations sur le contexte

Les exigences (détaillées plus haut) concernant « la mobilité, l'historique des informations sur le contexte et l'efficacité d'approvisionnement des informations sur le contexte » sont destinées en particulier aux modèles de contexte spatial.

L'espace est un contexte important dans de nombreuses applications contextuelles. La plupart des définitions du contexte parlent de l'espace comme d'un facteur essentiel. Par exemple, Schilit, Adams et Want définissent trois aspects importants du contexte «Où vous êtes, qui vous êtes et quelles sont les ressources proches» [Schilit et al. 1994].

En outre, dans la définition la plus fréquemment utilisée du contexte [Dey et al. 2001], l'espace peut être considéré comme un aspect central pour le contexte d'une entité : «*Une entité est une personne, un lieu ou un objet qui est considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes* ». Suivant cette définition, les lieux sont des entités spatiales, et l'interaction nécessite généralement une certaine proximité. Ainsi, certaines approches de modélisation de contexte donnent à l'espace et à l'emplacement un traitement préférentiel.

La plupart des modèles spatiaux organisent leurs informations sur le contexte selon leur emplacement physique. Cela pourrait être l'emplacement des entités du monde réel qui est décrite dans les informations sur le contexte (par exemple, les limites d'une pièce), l'emplacement du capteur qui mesure les informations sur le contexte.

Etant donné que nous cherchons à manipuler des informations sur le contexte de type textuel, nous considérons la structure physique du texte (sa mise en forme) comme un aspect spatial des entités textuelles. En effet, la localisation d'une entité textuelle par rapport à une autre peut être vue comme une localisation spatiale dans un document.

3.5 Modèles des informations sur le contexte basés sur les ontologies

Les approches ontologiques de la modélisation de contexte sont connues pour leur qualité expressive très poussée et ont pour but de satisfaire principalement les exigences suivantes : l'hétérogénéité, les relations et les dépendances, et le raisonnement.

Le contexte peut être considéré comme un type particulier de connaissance. Par conséquent, il est tout à fait naturel de vérifier si un cadre connu pour la représentation des connaissances et de raisonnement peut être approprié pour manipuler des contextes.

A la recherche d'un haut potentiel expressif et d'un moyen de raisonnement performant, de nombreux modèles basés sur des ontologies optent pour OWL [Wanget al. 2004].

Les approches de modélisation des informations sur le contexte basées sur des ontologies exploitent la représentation et la puissance du raisonnement de OWL à des fins multiples: (a) l'expressivité du langage est utilisée pour décrire les données dans un contexte complexe qui ne peut être représenté, par exemple, en CC/PP [Klyne et al. 2004]; (b) en fournissant une sémantique formelle des informations sur le contexte, il devient possible de partager et/ou d'intégrer le contexte de différentes sources; (c) les outils de raisonnement peuvent également être utilisés à la fois pour vérifier la cohérence de l'ensemble des relations décrivant des informations sur le contexte, et surtout, pour reconnaître qu'un ensemble de cas particuliers d'information sur le contexte de base et leurs relations révèle effectivement la présence d'une caractérisation plus abstraite (par exemple, l'activité de l'utilisateur peut être automatiquement reconnue).

Nous pouvons considérer les approches basées sur les ontologies parmi les plus complètes. En effet, elles vérifient la majorité des exigences d'une bonne approche de modélisation des informations sur le contexte. Ces approches, certes, offrent un moyen expressif très poussé, mais cette expressivité pèse lourd dans la performance.

3.6 Modèle basé sur l'abstraction de contexte

L'idée principale pour ce type de modèle est de faire une abstraction du contexte de bas niveau qui reçoit les perceptions de capteurs par des couches qui génèrent ou déclenchent les actions du système. Plusieurs notions différentes ont été utilisées pour se référer à la couche de contexte de haut niveau. Nous citons à titre d'exemple la notation de « situation » que

définit [Dobson et al. 2006]. Dobson définit une « situation » comme un graphe résultant d'un homomorphisme d'un autre graphe représentant un contexte. Par conséquent, le contexte et la situation sont modélisés comme des graphes. La notion de situation est utilisée comme un concept de haut niveau pour une représentation de l'état d'une ou plusieurs entités. Souvent cette notion est liée à l'aspect temporel (voir Figure 9).

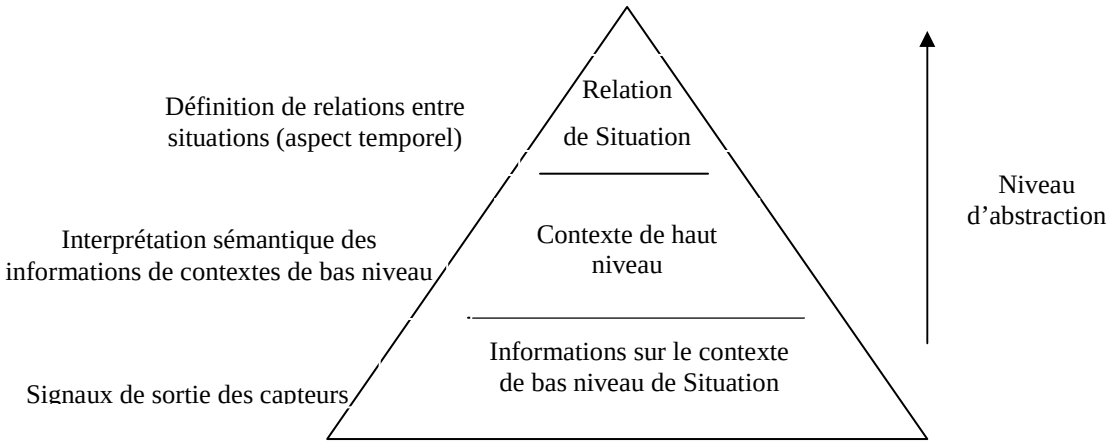


Figure 9 Vue d'ensemble des couches d'abstraction et interprétation de la sémantique du contexte

A titre d'exemple, la modélisation d'une conférence peut se faire à l'aide de plusieurs situations : « débuter l'enregistrement », « énoncer le discours du conférencier », « exposer de nouveau diaporama », « poser des questions », « finir l'enregistrement ». Ces situations sont liées par des liens d'ordonnancements temporels tels que : durant, avant, etc. Chaque situation décrit un état avec des informations sur le contexte récoltées.

Ce que nous pouvons retenir de ce type de modélisations pour les informations sur le contexte de type textuels, c'est qu'il faut prendre en compte l'abstraction des informations. Ce que nous appelons aussi la gestion de la granularité des informations sur le contexte.

4 Conclusion

Dans le domaine de l'informatique pervasive, des travaux ont fait une avancée importante dans la modélisation des informations sur le contexte. Il est judicieux de s'inspirer de ces travaux existant pour la modélisation et l'utilisation de la notion de contexte pour l'extraction d'information et d'annotations sémantiques à partir de texte.

L'idée principale est de faire une projection de la notion de « contexte » dans le domaine de l'informatique pervasive sur le domaine de la génération d'annotations sémantiques. Ceci nous a permis de mettre en relief certains aspects dans la modélisation des informations sur le contexte de type textuel tels que :

- La prise en compte de l'hétérogénéité des sources d'information ;
- L'importance de l'aspect spatial qui peut être assimilé à la structure physique du texte ;
- L'importance de l'aspect temporel ;
- L'importance des historiques bien qu'ils soient difficiles à gérer ;

- Les informations peuvent être incorrectes, ce qui mène à la nécessité de gérer les incohérences ;
- Un mécanisme de raisonnement efficace est requis ;
- Un modèle de représentation simple tel que RDF est adapté ;
- Une modélisation permettant de distinguer certaines informations pertinentes par rapport à d'autres;
- L'importance de la gestion de la granularité d'information sur le contexte

Il peut paraître prétentieux de satisfaire tous ces aspects dans le domaine de l'extraction d'informations et d'annotation à partir du texte en prenant en compte la notion de « contexte ». Néanmoins, nous avons élaboré une modélisation générique qui regroupe la majorité de ces aspects et nous avons réussi une tentative de génération d'annotations avec la prise en compte du contexte, tout en mettant en évidence des pistes de recherches que nous avons jugé utile de signaler.

Chapitre III : Modélisation des annotations sémantiques contextuelles

1 Introduction

Les êtres humains transmettent des idées les uns aux autres et réagissent à ces idées. Cela est dû à plusieurs facteurs : la richesse de la langue qu'ils partagent, la compréhension commune de beaucoup de règles décrivant la façon dont le monde fonctionne, et une compréhension implicite de la vie quotidienne. Quand les humains parlent entre eux, ils sont capables d'utiliser les informations implicites de la situation, ou son contexte, d'augmenter la portée de la conversation. Cette capacité à transmettre des idées n'est pas la même pour les humains interagissant avec les ordinateurs. Par conséquent, les ordinateurs ne sont pas actuellement en mesure de tirer pleinement parti du contexte du dialogue homme-machine. En améliorant l'accès de l'ordinateur au contexte, on augmente la richesse sémantique de la communication dans l'interaction homme-machine et ceci est l'un des objectifs du Web sémantique.

Afin d'utiliser efficacement le contexte, nous devons comprendre ce que c'est et comment il peut être utilisé. Nous exposerons dans ce chapitre plusieurs définitions du contexte existant dans la littérature. Nous proposerons par la suite une projection de ces définitions par rapport à notre problème qui est « la modélisation du contexte dans le cadre des annotations générées à partir de texte ».

Par ailleurs, nous avons besoin de définir certains termes abordés dans notre thèse, à savoir, méta-donnée, annotation, annotation sémantique.

2 Définition relatives aux annotations

2.1 Méta-donnée vs annotation

Nous mettons l'accent dans cette partie sur l'amalgame entre le terme « méta-donnée » et le terme « annotation » dans le domaine du Web sémantique et que Yannick Prié et Serge Garlatti exposent dans [Prié et al. 2005].

- Une méta[14]-donnée est littéralement définie par « *une donnée sur une donnée* ». Cette définition est un peu vague voire ambiguë, et elle est comprise de manière différente d'une communauté à une autre. Par exemple, dans certains cas la donnée est considérée comme ayant le même statut que celui de sa méta-donnée, traitable par un système informatique ; dans d'autres, la donnée n'est interprétable que par un être humain, et seule la méta-donnée en permet le traitement automatique.
- Selon [Prié et al. 2005], « *une annotation est à la base une note critique ou explicative accompagnant un texte, et par extension, une quelconque marque de lecture portée sur un document, que celui-ci soit textuel ou image* ».

Dans la littérature, la communauté anglophone utilise le terme « annotation » lorsqu'il s'agit d'un descriptif attaché à ce qu'il décrit ou en cours d'extraction ou d'identification. Dès que ce descriptif est stocké dans une base sur un serveur, la communauté anglophone utilise le terme « metadata » (méta-donnée).

[14] Préfixe "meta" (grec) : ce qui dépasse, englobe un objet, une science. (source : Le Robert)

Dans le cadre de notre thèse, nous ne faisons pas de distinction entre les deux termes et nous avons choisi de conserver le terme « annotation ».

2.2 Annotation & annotation sémantique

La Digital Library Federation (DLF), une association constituée des quinze bibliothèques américaines les plus importantes (aux Etats-Unis), a défini trois sortes d'annotations qui peuvent s'appliquer aux ressources documentaires d'une bibliothèque numérique [Handschuh, 2005] :

- *L'annotation administrative* indique les informations associées à la création et à la maintenance de la ressource documentaire telles « qui, quoi, où et comment ». Dans ce cadre, le schéma des annotations « DublinCore [15]» fait office de standard pour l'annotation avec des descripteurs tels que l'auteur, le titre, l'éditeur, la date de publication, la langue, etc.
- *L'annotation structurelle* relie des parties de ressources documentaires entre elles afin de constituer une représentation physique d'un document. Selon [Rinaldi et al., 2003], cette annotation permet de définir la structure physique du document, son organisation en entête et corps, en sections, paragraphes et phrases.
- *L'annotation descriptive* décrit une ressource documentaire relativement à son contenu informationnel.

Nous nous somme intéressés dans ce mémoire aux deux types d'annotations : structurelle et descriptive.

Par ailleurs, le contenu d'un document peut être analysé selon différents points de vue, et chacun d'entre eux peut être utile pour un but bien précis de l'application qui utilise les annotations descriptives issues de ces analyses. Selon Euzenat [Euzenat, 2005], il existe trois structures pouvant constituer des points de vue différents d'approches pour *l'annotation descriptive* : « la *structure grammaticale* pour analyser les relations entre syntagmes, la *structure rhétorique* pour dégager l'argumentation d'un texte ou encore la *structure logique* pour interroger le sens d'un document ».

Dans le cadre du Web Sémantique, une *annotation sémantique* réfère le plus souvent à l'annotation descriptive qui s'intéresse à la structure logique du contenu d'un document.

Le terme « sémantique » est lui-même ambigu mais il indique une volonté de faire émerger le sens d'un contenu d'une manière formelle selon les préceptes de la logique [Amardeilh, 2007]. L'objectif des annotations sémantiques, comme le souligne [Corcho, 2006], est d'exprimer la « sémantique » du contenu d'une ressource afin d'améliorer sa compréhension, sa recherche et donc sa réutilisation par les utilisateurs finaux.

Dans le cadre de ce mémoire, nous retenons la définition d'Amardeilh de **l'annotation sémantique** [Amardeilh, 2007]: « *une représentation formelle d'un contenu, exprimée à l'aide de concepts, relations et instances décrits dans une ontologie, et reliée à la ressource documentaire source.* »

[15] http://dublincore.org/

3 Contexte dans le cas d'annotations sémantiques

3.1 Définitions du contexte

Pour élaborer une modélisation du contexte qui peut être utilisée de manière concrète dans le domaine de l'extraction d'information à partir de texte, nous examinerons comment les chercheurs ont tenté de définir le contexte dans leurs travaux. Bien que la plupart des personnes, tacitement, comprennent ce qu'est le contexte, elles ont du mal à élucider cette notion.

Dans le travail de Schilit et Theimer [Schilit et al. 1994], où le terme « conscience contextuelle »[16] a été introduit pour la première fois, les auteurs se réfèrent au contexte comme « *l'emplacement, l'identité des personnes et des objets* à proximité, et les *changements de ces objets* ». Dans [Dey, 2001] l'auteur critique ce type de définition utilisant des exemples en argumentant que celles-ci, sont difficiles à mettre en œuvre dans des applications informatiques. L'auteur ajoute que lorsque nous voulons déterminer si un type d'information, ne figurant pas dans la définition, fait partie du contexte ou non, il est difficile de déterminer comment nous pouvons utiliser la définition pour résoudre le problème.

D'autres définitions ont simplement fourni des synonymes pour le contexte, par exemple, se référant au contexte par « l'environnement » ou par « la situation ». Pareillement aux définitions utilisant des exemples, les définitions qui utilisent simplement des synonymes de contexte sont difficiles à appliquer.

Dans [Schilit et al. 1994] les auteurs affirment que les aspects importants du contexte sont : *où vous êtes, avec qui vous êtes,* et *quelles sont les ressources à proximité.*

Pascoe [Pascoe, 1998] définit le contexte comme le sous-ensemble des états physiques et conceptuels de l'intérêt porté pour une entité particulière. Cette définition, très générale, ne traite pas des aspects pratiques pour les applications informatiques réelles.

Pour pallier les limites des définitions proposées de la notion de contexte, Dey [Dey, 2001] propose la définition générale suivante, qui est probablement la plus largement acceptée :

« *Le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet, considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes.* »

Dans le domaine de la logique, McCarthy, qui est considéré comme le maître en la matière, définit le contexte [McCarthy, 1993] comme la généralisation d'une collection d'hypothèses. Les contextes sont ainsi formulés comme des objets formels de première classe. McCarthy postule qu'une proposition « p » est vraie dans un contexte « c », où « c » est supposé capturer tout ce qui n'est pas explicite dans « p » mais qui est requis pour faire de « p » un énoncé significatif pour représenter ce qu'il est supposé établir.

[Brezillon et al. 2003] commente la définition de McCarthy en soulignant ses conséquences : (a) un contexte est toujours relatif à un autre contexte, (b) les contextes sont de dimension infinie (c) ils ne peuvent donc pas être décrits complètement.

[16] Le terme anglophone est « context-aware »

[Brezillon et al. 2003] propose alors la définition suivante : « *le contexte est l'information qui caractérise les interactions entre humains, applications et l'environnement* ».

Les auteurs dans [Zimmermann et al. 2007] proposent une extension aux définitions proposées afin d'obtenir une meilleure compréhension du contexte pour les utilisateurs et permettre d'incorporer plus facilement la notion de contexte par les développeurs dans des applications informatiques. Zimmermann propose un point de vue dit « opérationnel » de la notion du contexte. Il propose cinq catégories fondamentales pour l'information contextuelle décrivant une entité : *individualité*, *activité*, *lieu*, *temps*, et *relations.* Toute information contextuelle est supposée appartenir à l'une des ces catégories.

La catégorie *individualité* contient des propriétés et des attributs décrivant l'entité elle-même. La catégorie *activité* couvre les tâches pouvant impliquer cette entité. Les catégories *lieu* et *temps* de l'information contextuelle fournissent les coordonnées spatio-temporelles de l'entité. Enfin, la catégorie des *relations* représente des informations sur toute relation possible de l'entité avec d'autres entités.

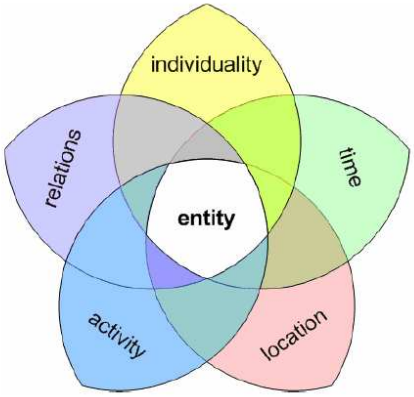


Figure 10 Les cinq catégories fondamentales pour l'information contextuelle [Zimmermann et al. 2007]

En ce qui concerne l'individualité, de notre point de vue, cette modélisation contredit, en partie, la définition de McCarthy. En effet, McCarthy postule que le contexte d'une entité ne fait pas partie de l'entité elle-même, mais concerne des informations qui complètent sa compréhension.

Dans ce qui suit nous ne proposons pas une définition alternative aux définitions déjà proposée, mais nous ferons une projection de ces définitions sur un type particulier d'entité, à savoir, « *des entités textuelles* ».

3.2 Modélisation du contexte pour le texte et sa sémantique

Nous avons constaté, après des observations empiriques, que l'utilisation d'une annotation sémantique quelconque est étroitement liée à son contexte d'apparition dans le texte. En effet, l'interprétation d'une annotation sémantique particulière peut produire des incohérences si nous ignorons les autres annotations sémantiques qui *la précédent*, *la suivent*, *l'imbriquent*, etc. Ceci nous a amené à modéliser la notion de contexte pour le texte et ce que cela implique pour les annotations sémantiques [Mokhtari et al. 2009a].

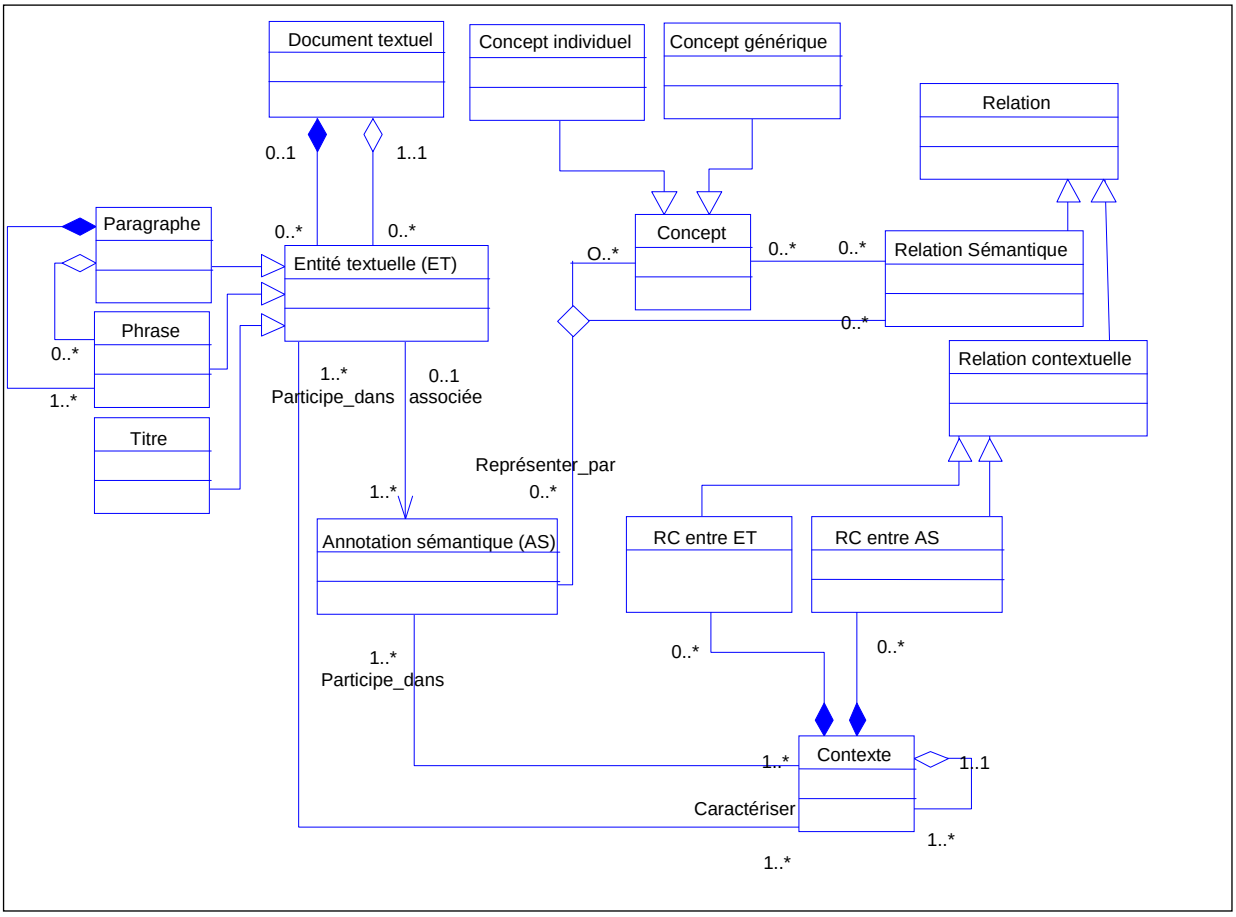


Figure 11 Modélisation de la notion de contexte pour des entités textuelles (diagramme de classe UML)

La Figure 11 montre une vue globale de la modélisation du contexte pour des entités textuelles et leurs sémantiques. Dans ce qui suit, nous allons définir et expliquer le rôle des différentes composantes du diagramme de classe UML pour une meilleure compréhension de notre modélisation de la notion de contexte.

3.3 Contexte d'une entité textuelle

3.3.1 Définition d'une entité textuelle

Nous appelons une **entité textuelle** « *tout élément textuel (un mot, une phrase, une partie d'une phrase, un titre, un texte entre parenthèses, un paragraphe, une section, le texte de tout un document, etc.) duquel une annotation sémantique (AS) peut être générée* ».

3.3.2 Définition du contexte d'une entité textuelle

*Le **contexte d'une ET** est défini par un ensemble de tuple: RCi(ETn, ET) ∪ RCj(ET, ETm) où:*

- *ETn et ETm sont les entités textuelles en relation avec l'ET en question ;*
- *RCi et RCj sont les relations contextuelles (structurelles, ou autres) qui lient ETn (resp. ETm) à l'ET.*

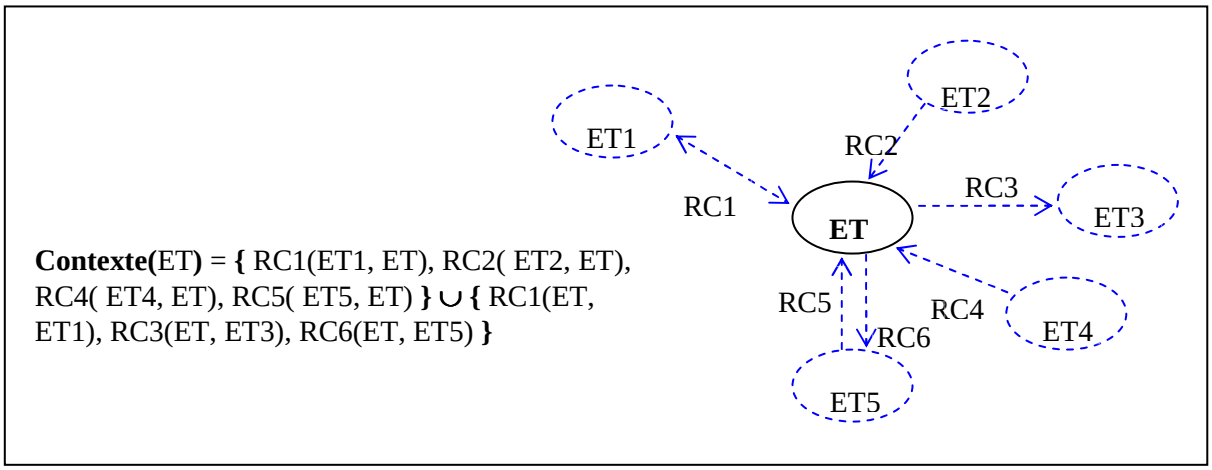


Figure 12 Contexte d'une entité textuelle « ET »

La Figure 12 montre un exemple du contexte d'une entité « ET » représenté par un ensemble de tuples constitué d'entités textuelles et de relations contextuelles.

Après avoir modélisé le contexte d'une entité textuelle, une question demeure : quels sont les types de relations contextuelles pouvant exister entre les entités textuelles ?

3.3.3 Relations contextuelles entre les entités textuelles : relations structurelles

Dans le cadre de notre travail, identifier les relations entre entités textuelles est une forme de description de textes. Une description qui se préoccupe de l'emplacement du texte et de sa forme et non pas de son sens. En effet, ceci correspond à une *annotation structurelle* qui permet de définir la structure physique du document et son organisation. Les relations contextuelles entre les entités textuelles sont des relations qui définissent la structure physique du texte (ou du document).

Les relations contextuelles entre entités textuelles sont des relations qui décrivent la mise en page du document (du texte), qui définissent aussi les différentes zones de texte, leur agencement les unes par rapport aux autres (relations de *succession, d'imbrication*, etc.), ainsi que l'ensemble de leurs caractéristiques typographiques : police, couleur, gras, italique, etc.

3.4 Contexte d'une annotation sémantique

En revenant sur la définition de *l'annotation sémantique* proposée par [Amardeilh, 2007] et que nous avons adoptée, l'annotation sémantique est exprimée par des concepts, relations et instances décrits dans une ontologie.

Une annotation sémantique est associée au texte qu'elle annote, c'est-à-dire, à l'entité textuelle qu'elle annote.

De la même manière que l'entité textuelle, son annotation sémantique associée a aussi un contexte qui a besoin d'être modélisé. Ce contexte concerne la « sémantique » (modélisée par des annotations sémantiques) que l'entité textuelle transmet.

3.4.1 Définition du contexte d'une annotation sémantique

*Le **contexte d'une AS** est défini par un ensemble de tuple RCi(ASn, AS) ∪ RCj(AS, ASm) où :*

- *ASn et ASm sont les annotations sémantiques en relation avec l'AS en question ;*
- *RCi et RCj sont les relations contextuelles (rhétorique ou autre) qui lient l'ASn (resp. ASm) à l'AS.*

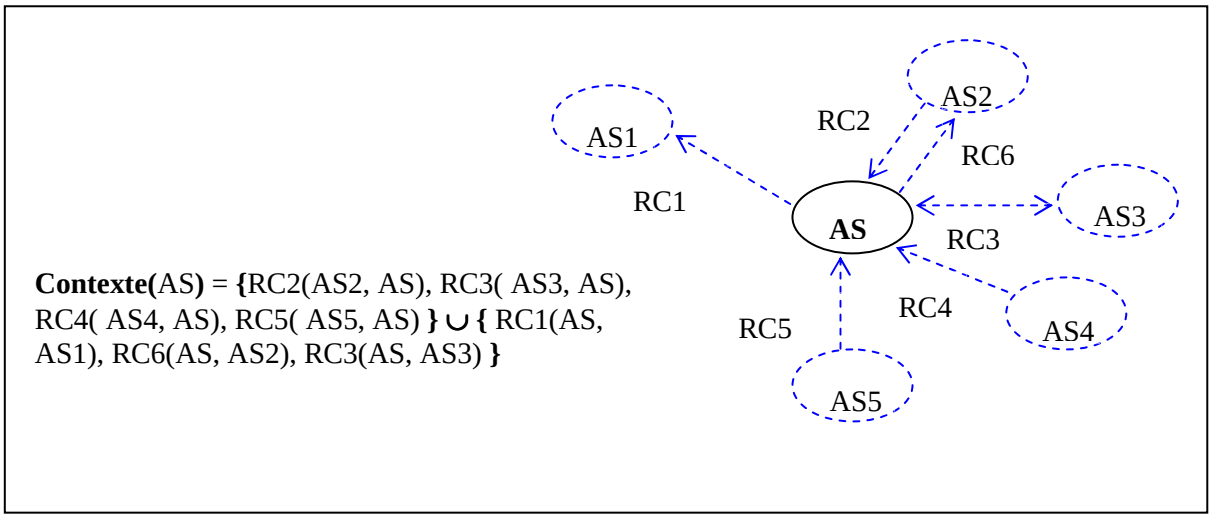


Figure 13 Contexte d'une annotation sémantique « AS »

La Figure 13 montre un exemple du contexte d'une annotation sémantique « AS » représenté par un ensemble de tuples constitué d'annotations sémantiques et de relations contextuelles.

3.4.2 Définition d'annotation sémantique contextuelle

*Nous définissons **une annotation sémantique contextuelle** comme toute annotation sémantique augmentée de son contexte.*

3.4.3 Relations contextuelles entre annotations sémantiques

Nous rappelons que *l'annotation descriptive* décrit une ressource (du texte dans notre cas) vis-à-vis de son contenu informationnel. Par conséquent, identifier des relations entre des annotations sémantiques fait partie aussi de l'annotation descriptive.

Nous mettons l'accent sur le fait que le contenu d'un document textuel peut être analysé selon différents points de vue. Pour les annotations descriptives, il existe trois structures du document à analyser [Euzenat, 2005]:

- Dans la *structure grammaticale* où l'analyse est portée sur les relations entre syntagmes, plusieurs informations peuvent être intéressantes à identifier. Nous citons à titre d'exemple le rôle linguistique d'un mot dans une phrase (sujet, verbe, etc.). Dans ce cadre, il est question de *contexte linguistique*, et les relations contextuelles sont des

relations de type linguistique (par exemple : *est le complément de, est le verbe de, le temps d'un verbe, etc.*).

- Dans la *structure logique*, où l'analyse est portée sur le sens du texte, il s'agit d'identifier des concepts, des instances et des relations. Ceci correspond à la construction des annotations sémantique (concepts, instances et relations). Dans ce cadre, il s'agit de relations du domaine décrites dans une ontologie.
- La *structure rhétorique*, où l'analyse est portée sur l'argumentation d'un texte, révèle l'aspect pertinent de la sémantique existante entre les annotations sémantiques. En effet, dans ce cadre, les relations contextuelles correspondent aux relations rhétoriques.

D'autres types de relations contextuelles sont aussi pertinents que les relations rhétoriques, à savoir, les relations spatiales et les relations temporelles. Mais avant de définir les différents types de relations contextuelles pouvant exister entre les annotations sémantiques, nous allons aborder une notion dont ces relations contextuelles dépendent, à savoir, *la notion de granularité.*

4 Notion de granularité

L'utilisation de la structure du texte, à travers les entités textuelles, nous permet de choisir le niveau de détail à étudier ou « granularité ». En effet, nous pouvons étudier, à titre d'exemple, d'une part les « paragraphes » et les relations contextuelles entre eux, et d'autre part les relations contextuelles entre « phrases ».

En outre, la notion de granularité vient conforter notre vision sur l'emboitement entre les contextes et ainsi assurer la présence du contexte à tous les niveaux d'abstraction. Par conséquent, nous pouvons étudier les relations contextuelles, non seulement entre les entités textuelles du même niveau d'abstraction, mais aussi entre les entités textuelles appartenant à des niveaux différents.

Par ailleurs, le niveau de granularité « mot » est considéré comme un premier niveau où l'étude de la notion de contexte n'est pas utile. En effet, dans ce niveau de granularité, chercher la sémantique correspondant aux entités textuelles de type « mots » se résume à identifier des concepts et des relations qui les lient. Le contexte d'un « mot » à ce niveau de granularité n'est autre que l'annotation sémantique à laquelle il participe. En effet, comme le montre la Figure 14, nous ne pouvons parler d'annotations sémantiques contextuelles qu'à partir d'un niveau de granularité où les entités textuelles (ETs) permettent de construire des annotations sémantiques (AS). C'est-à-dire, à partir des entités textuelles de type ETp et celles qui les englobent : « phrases, paragraphes, etc. »

La Figure 14 illustre la prise en compte de différents niveaux de granularité des entités textuelles dans la modélisation de la notion de contexte. Ceci affecte aussi bien les entités textuelles que leurs annotations sémantiques associées.

Notre modélisation est la même pour chaque niveau de granularité, à savoir, des entités textuelles, leur sémantique et des relations contextuelles. Ce qui diffère d'un niveau à un autre, c'est principalement le type de relation contextuelle.

Dans le niveau de granularité « phrase », nous soulignons que les relations contextuelles entre annotations sémantiques AS ne sont pas du même type que celles existant entre des concepts dans une annotation sémantique. En effet, les relations contextuelles (rhétoriques ou autres) du niveau « phrase » ne dépendent pas du domaine d'application.

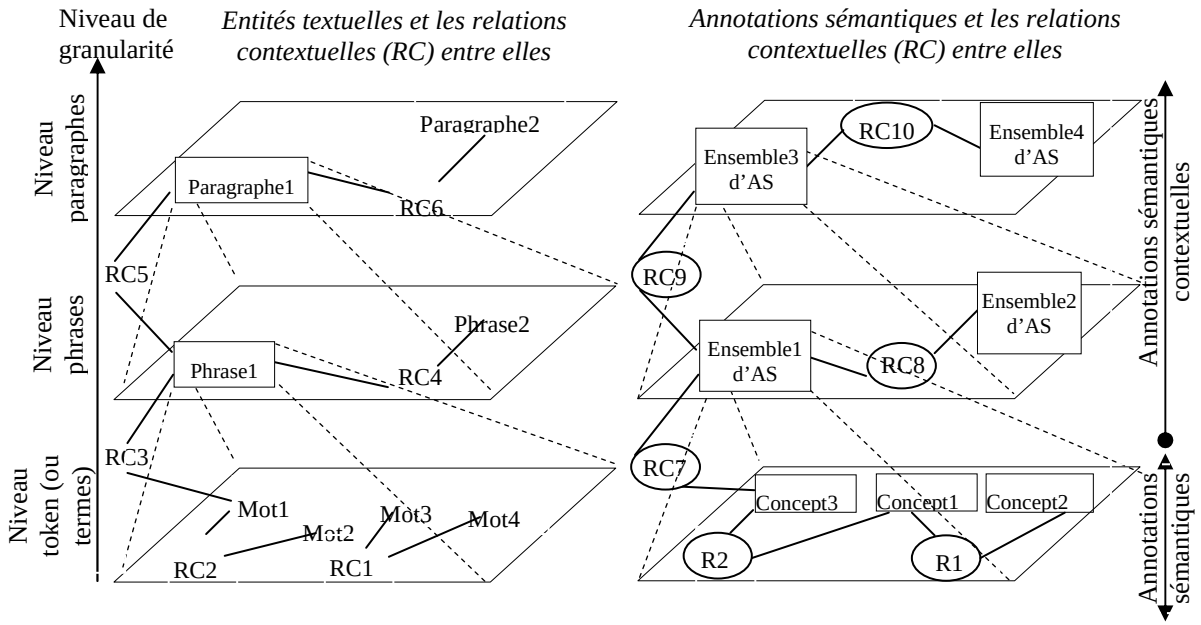


Figure 14 La prise en compte des niveaux de granularité dans la modélisation du contexte

Nous soulignons que nous n'avons pas mentionné dans la Figure 14 tous les niveaux de granularité existant. Nous citons à titre d'exemple, le niveau intermédiaire « partie de phrase ». En effet, dans la même phrase il peut y avoir des relations contextuelles entre ses sous parties.

5 Différents types de relations contextuelles

Différents types de relations contextuelles peuvent exister [Mokhtari et al. 2009b] et qui dépendent d'une part de l'objet que nous manipulons (entité textuelle ou annotation sémantique), et d'autre part, du niveau de granularité de l'entité textuelle dont nous cherchons à identifier le contexte.

5.1 Relations contextuelles entre entités textuelles

Dans cette partie, nous allons lister les différents types de relations contextuelles pouvant exister entre les entités textuelles.

5.1.1 Relations contextuelles structurelles

Nous définissons une relation contextuelle structurelle comme toute relation exprimant la position d'une entité textuelle par rapport à une autre du même niveau de granularité ou non.

Nous pouvons prendre à titre d'exemple les relations de *succession*, *d'imbrication*, etc.

5.1.2 Relations contextuelles temporelles

Nous définissons une relation contextuelle temporelle entre ET comme toute relation exprimant une notion relative au temps entre une entité textuelle et d'autres du même niveau de granularité ou non.

Le type de ces relations temporelles dépend du niveau de granularité de l'entité textuelle. En effet, dans le cas où les entités textuelles étudiées sont des documents, nous pouvons considérer comme relations contextuelles les relations étudiant des aspects comme la date de rédaction du document, la date de la dernière mise-à-jour, la date de publication, etc. Dans ce cadre il s'agit de modéliser l'aspect « versioning » des entités textuelles.

5.2 Relations contextuelles entre annotations sémantiques

Dans cette partie, nous allons lister les différents types de relations contextuelles pouvant exister entre les annotations sémantiques.

5.2.1 Relations contextuelles spatiales

Nous définissons une relation contextuelle spatiale comme toute relation exprimant la position d'un objet décrit par une AS par rapport à un autre objet décrit par une autre AS du même niveau de granularité ou non.

Nous pouvons prendre à titre d'exemple, des relations exprimant un rang (*devant, derrière, après*...) ou un lieu (*dans, chez, sous*...).

Les relations spatiales sont connues, de même que les relations temporelles, pour entraîner des difficultés de raisonnement. Cependant une des pistes à explorer est l'utilisation des relations d'Egenhofer [Egenhofer et al., 1991] pour représenter les relations contextuelle spatiales.

5.2.2 Relations contextuelles temporelles

Nous définissons une relation contextuelle temporelle entre AS comme toute relation exprimant la notion de temps entre une annotation sémantique et d'autres du même niveau de granularité ou non.

Nous distinguons deux types de relations temporelles entre AS qui dépendent du niveau de granularité de l'annotation sémantique à étudier.

Au niveau d'une étude portée sur des entités textuelles de type « mots », nous pouvons considérer le *temps des verbes* (*présent*, *futur*, etc.) comme une relation contextuelle temporelle pour l'AS qui contient « le verbe » et qui exprime le moment où une action (ou un évènement) se manifeste.

Au niveau d'une étude portée sur des entités textuelles de type « phrase ou partie de phrase », nous pouvons avoir à titre d'exemple « *avant*, *depuis*, *pendant*, etc.» comme relations contextuelles temporelles entre AS.

Les relations temporelles sont connues pour être difficiles à exploiter (inférence). En effet, dans cette thèse nous nous concentrons sur l'extraction des relations contextuelles et des annotations sémantiques contextuelles. Néanmoins, une des pistes à explorer est d'utiliser le raisonnement dit « de propagation de contraintes », et plus particulièrement en utilisant les relations (ou intervalles) d'Allen [Allen, 1984]. Les relations d'Allen (Figure 15) peuvent être utilisées pour représenter les relations contextuelles temporelles. Par exemple les relations *avant* et *durant* peuvent être représentées respectivement par les relations (A before B ou A meets B), et (A during B). Ainsi nous pouvons faciliter l'inférence sur l'aspect temporel.

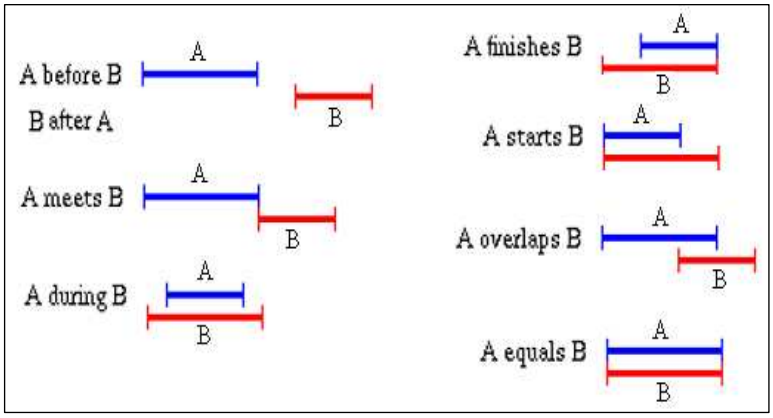


Figure 15 Les relations d'Allen

5.2.3 Relations contextuelles rhétoriques

Nous appelons relation contextuelle rhétorique toute relation exprimant une argumentation entre une annotation sémantique et d'autres du même niveau de granularité ou non.

Nous pouvons avoir comme relations contextuelles rhétoriques entre AS, les relations explicites dites *liens logiques* (dites aussi : *relations de discours* ou *connecteurs logiques*) qui expriment : une addition (*de plus, d'ailleurs*, etc.), une illustration (*ainsi, comme*, etc.), etc.

Ces relations contextuelles rhétoriques concernent aussi les relations de discours implicites, où il y a une argumentation, entre deux phrases par exemple, sans l'existence de connecteur logique.

Lorsque nous parlons de relations rhétoriques, il est naturel d'aborder la théorie d'analyse du discours permettant l'analyse de texte, telle que RST (Rhetorical Structure Theory) [Mann et al. 1987].

La RST est définie en tant que théorie descriptive et fonctionnelle du texte avec des aspects à la fois sémantiques et intentionnels. Les auteurs proposent une vingtaine de relations rhétoriques pouvant lier deux parties (ou segments) de texte adjacents. Un des segments, dit « primordial pour la cohérence », possède le statut de noyau. L'autre segment de texte possède le statut de satellite et est dit segment optionnel.

Les segments sont soit minimaux (correspondant généralement aux propositions) ou composés. Dans la RST l'identification des relations rhétoriques repose sur une interprétation sémantico-pragmatique du contenu du segment. Cette interprétation est liée aux opinions de l'analyste du texte. D'où une analyse subjective.

Par ailleurs, il existe des schémas rhétoriques décrivant l'organisation structurelle d'un texte, permettent de lier un noyau et un satellite, deux ou plusieurs noyaux entre eux, et un noyau avec plusieurs satellites.

La structure du texte est donc définie en termes de compositions d'applications de schémas, et ce de manière récursive. La structure rhétorique finale d'un texte est hiérarchique et se présente sous la forme d'un arbre RST.

La Figure 16 montre un exemple de texte (parlant de lactose) analysé suivant RST.

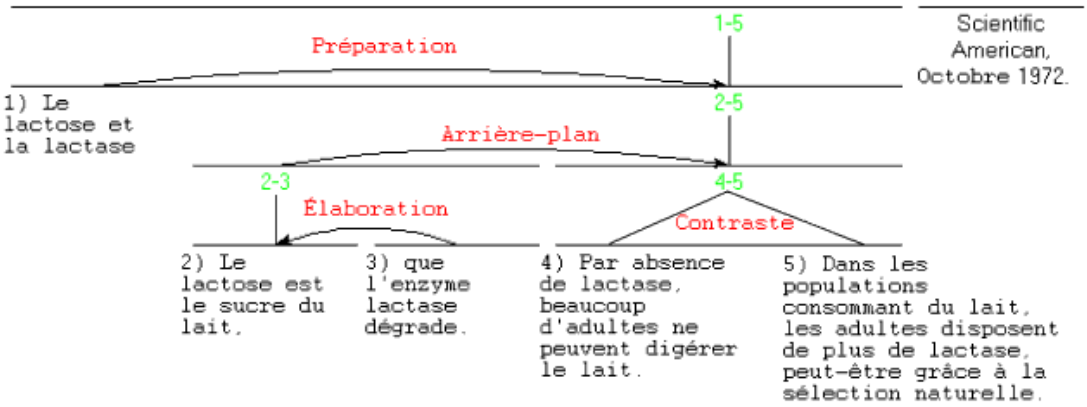


Figure 16 Exemple[17] d'arbre d'RST

Nous ne prétendons pas proposer une étude exhaustive sur les relations rhétoriques ou d'extraire automatiquement la structure RST. Néanmoins nous souhaitons offrir une plateforme de génération automatique d'annotations sémantiques où les relations contextuelles rhétoriques ajoutent une sémantique supplémentaire et où leur exploitation devient accessible via les technologies du Web sémantique. Nous nous sommes inspirés aussi de l'arbre RST pour offrir un arbre structurant les annotations sémantiques en utilisant les relations contextuelles.

Pour les relations contextuelles entre ET comme pour celles existant entre AS, nous ne prétendons pas avoir exposé une liste exhaustive des différents types de relations contextuelles pouvant exister dans un texte. Une mise en relief d'un type particulier de relations contextuelle par rapport à un but applicatif précis pourra être utile pour couvrir un point de vue différent dans l'analyse de texte. A titre d'exemple, des relations de type « le titre est *plus important que* le paragraphe » pourraient ajouter une sémantique additionnelle.

6 Conclusion

La notion de contexte est perçue de différentes manières selon les auteurs dans le domaine de l'extraction d'informations. Dans ce chapitre, nous avons parcouru diverses définitions génériques de la notion de contexte. Nous avons proposé une projection des aspects importants du 'contexte' sur notre domaine de génération d'annotation sémantique. D'une manière générale, nous définissons le contexte d'une annotation par toutes les annotations qui sont liées à celle-ci par des relations contextuelles. Les relations contextuelles peuvent être des relations rhétoriques, spatiales, temporelles ou issues de la structure physique du texte comme la relation d'imbrication.

[17] Exemple disponible sur www.sfu.ca.rst/07french/index/html

En outre, nous avons vérifié par le biais de la notion de granularité que la récursivité mentionnée dans la définition de McCarthy [McCarthy, 1993] est présente dans notre modélisation de la notion de contexte.

La modélisation de la notion de contexte proposée nous a permis de définir quelles informations à extraire de texte pour engendrer des annotations sémantiques contextuelles.

Chapitre IV : Génération d’annotations sémantiques contextuelles

1 Architecture générale du processus d'extraction

La génération d'annotations sémantiques contextuelles passe par plusieurs étapes et avec différents algorithmes [Mokhtari et al. 2009c]. Nous pouvons regrouper ces étapes en deux grandes phases : la première phase regroupe tous les traitements et manipulations aboutissant à reconstituer la structure du texte (*manipulation textuelle*), la seconde phase concerne les traitements permettant de générer des annotations contextuelles (*manipulation sémantique*).

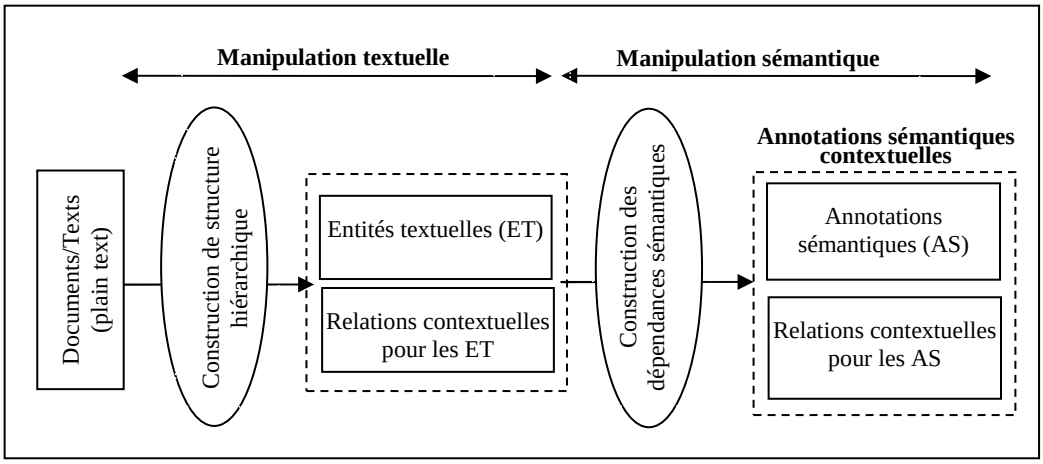


Figure 17 Principales phases du processus de génération des annotations contextuelles

L'objectif de la première phase est d'identifier les entités textuelles ainsi que les relations contextuelles issues de la structure du texte. Ces relations, nous le rappelons, sont du type : *est imbriqué dans, est le successeur de, est le prédécesseur de,…* Les relations contextuelles identifiées à ce niveau sont déduites de la structure hiérarchique.

L'objectif de la deuxième phase est de générer des annotations RDF correspondant à chaque entité textuelle et les relations contextuelles à ce niveau sont, entre autres, des relations de discours.

Nous rappelons que les entités textuelles dans un document peuvent être : une section, une sous-section, un paragraphe, une phrase ou une partie de phrase.

Nous soulignons que par souci d'optimisation, l'implantation de cette architecture entraîne le regroupement de certains traitements dans les deux phases.

2 Manipulation de la structure du texte

La phase de manipulation de la structure du texte est basée principalement sur la plateforme GATE qui offre un support d'implémentation pour les différentes heuristiques que nous proposons.

2.1 Identification des titres

Bien que la plateforme GATE offre des « *transducers* » par défaut permettant d'identifier les titres, cette identification n'est pas performante et donne beaucoup de résultats erronés. Ceci nous a poussé à proposer une méthode plus performante.

Nous nous somme intéressés à identifier les titres dans un texte en se basant sur les *indices numériques* qui précèdent les titres. Nous nous somme focalisés sur un type particulier de ces indices. En effet, le type d'indices pris en compte et par lequel nous pouvons identifier les titres est représenté par la grammaire[18] suivante:

Indice_numérique ::= *Nombre* ('.' *Nombre*)*

Nombre :: = [0-9]+

Voici un exemple simple que peut engendrer cette grammaire : *7.5.2*

Nous avons utilisé une règle JAPE pour pouvoir identifier ce type d'indice. Cependant, si notre intérêt ne se focalise que sur les deux principales caractéristiques de cet indice, à savoir, une alternance entre chiffres et points ('.'), nous aurons beaucoup de résultats erronés. En effet, tout chiffre à virgule (écrit sous forme d'un point) sera identifié à tort comme un indice numérique.

Pour résoudre ce problème, nous avons élaboré des heuristiques qui éliminent le bruit pouvant être engendré.

- *Heuristique* 1. l'indice numérique doit être suivi d'un (ou plusieurs) caractère de type espace : *espace simple* et/ou *tabulation*
- *Heuristique* 2. l'indice numérique doit être au début de la phrase à laquelle il appartient.
- *Heuristique* 3. la phrase contenant l'indice numérique doit être identifiée comme la seule phrase dans le paragraphe qui la contient.

Ces trois heuristiques sont traduites sous forme d'algorithmes permettant de filtrer les indices numériques correspondant à un titre parmi ceux correspondant à des valeurs numériques dans le texte.

La règle JAPE (pour plus de détails sur JAPE, voir Chapitre V section 3.2) qui permet d'identifier les indices numériques précédant les titres s'écrit comme suit (`TitleNumber` correspond à l'indice numérique):

[18] La notation utilisée est l'EBNF définie par le W3C dans *Extensible Markup Language* (XML) 1.1 section 6 (http://www.w3.org/TR/2004/REC-xml11-20040204/#sec-notation)

```
Rule:TitleNumber
({Token.kind == number}
 (({SpaceToken.kind == control}({SpaceToken.kind == space})*)|
  (({Token.lemma == "."}{Token.kind == number})+{SpaceToken.kind ==
control} {SpaceToken.kind == space})*
  )
 )
)
:TitleNumber -->
:TitleNumber.TitleNumber = {kind = "titleNumber", rule=TitleNumber}
```

En plus de la prise en compte de l'alternance entre les chiffres et les points, cette règle code aussi l'« *Heuristique 1* ». Les deux autres heuristiques sont prises en compte au moment de la sélection des phrases correspondant aux titres. Les phrases sélectionnées comme « titres » sont les phrases ayant un indice numérique dans leur début et formant l'unique phrase du paragraphe qui les contient respectivement, en concordance avec les heuristiques 2 et 3.

Une fois que les phrases correspondant aux titres sont identifiées, il reste à construire les imbrications entre les titres, les sous titres et les paragraphes.

2.2 Construction des imbrications hiérarchiques : la structure du texte

Afin de construire la structure hiérarchique du document, nous avons élaboré un algorithme permettant d'identifier la portée des titres. Nous rappelons que le résultat de l'application des « *transducers* » dans la plate-forme GATE est sous forme d'un fichier XML. Chaque unité linguistique (mot, phrase, paragraphe, parenthèse, …) est balisée. Parmi les informations ajoutées à travers ces balises, nous citons les indicateurs *début* et *fin* des unités linguistiques dans le texte. Pour chaque unité linguistique, le fichier XML de sortie de GATE contient les attributs 'début' et 'fin' indiquant la position dans le texte de cette unité.

L'algorithme de construction des imbrications hiérarchiques *« Algorithme 1 »* se base entre autres sur ces indicateurs de positions.

Algorithme 1 : Construction des imbrications hiérarchiques du document

```
Node BuildDocumentHierarchy() {
  L : La liste de tous les titres déjà identifiés. Les titres sont triés selon l'ordre de position dans le texte ;
  L.index : un indice qui aide à parcourir la liste en l'incrémentant ;
        //Appliquer les heuristiques 2 et 3 de détection des titres
  Node : Le type de nœud de l'arbre « Tree » constitué de deux champs, un pour le titre « title » et un pour contenir les paragraphes associés au titre directement « Lp »
  Lps : La liste de tous les paragraphes détectés par GATE dans le texte
  i : un entier positif qui nous permet de parcourir la liste des titres
  Node root, nd ;
  L.index ← 1 ;
  root = new Node ();
  Lps' = selectParagraphsBefore(L.get(1)) ;
            // Sélectionne les paragraphes se trouvant avant le premier titre.
  root.insert(buildParagraphs(Lps'));
        //la fonction "insert" est une fonction d'insert dans la fin de liste.
  root.add(buildTree(L));
  return root
}//end BuildDocumentHierarchy
```

Nous mettons dans cet algorithme, en premier, tous les titres déjà identifiés dans une liste *L* en les ordonnant suivant leur ordre d'apparition dans le texte en utilisant les indicateurs de positions. Par la suite, nous sélectionnons parmi tous les paragraphes, les paragraphes se trouvant avant le premier titre en utilisant la fonction « *selectParagraphsBefore* ».

La fonction « *buildParagraphs »* permet de construire les imbrications des phrases, parties de phrases (pop), mots, etc. au sein des paragraphes (nous détaillerons cette fonction plus loin). Ensuite, un document est construit et est composé d'une part des paragraphes se positionnant avant le premier titre et d'autre part des nœuds du document générés en lançant la fonction récursive « *buildTree »* .

```
Node buildTree(L: liste des titres) {
  Node root = new Node();
  If (L.index > L.length()) { return root ; }
  title t = L.get(L.index);
  title first = t;
  while (L.index <= L.length()){
    t = L.get(L.index);
    if (first.SameLevel(t)) {     // le premier cas, le titre courant est un frère du titre « first »
        if (L.index == L.length()) { Lps = selectParagraphsAfter(t));}
        else {Lps = selectParagraphsBetween(t, L.get(L.index+1));
    }
    Node nd = new Node() ;
    nd.insert(buildParagraphs(Lps));
    nd.setTitle(t);
    root.add(nd);
    L.index ++;
   }else {
      if (first.isUpperLevelOf(t)) {
           // le deuxième cas, le titre courant a un niveau inférieur à  celui du titre « first »
           root.add(buildTree(L));
      } else {
          // le troisième cas, le titre courant a un niveau supérieur à celui du titre « first »
          return root ;
      } //endIf
   }//endIf
  }//endWhile
}//endBuildTree
```

« *buildTree* » est une fonction récursive. Le principe de cette fonction est de parcourir la liste des titres ordonnées « L » et de construire en même temps une structure hiérarchique (Tree). Après avoir construit le premier nœud correspondant au premier titre de la liste « L », le titre suivant devient le titre courant et peut être un de ces trois cas :

- le titre courant est un frère du titre « first ». Dans ce cas, il est ajouté à l'arbre avec ses paragraphes ;
- le titre courant est un titre de niveau inférieur. Dans ce cas, un sous arbre est créé par l'appel récursif de la fonction « *buildTree* » ;

- le titre courant est un titre de niveau supérieur, et dans ce cas de figure nous sortons de l'appel récursif de la fonction par un « *return root* ».

Après avoir vu la fonction récursive « *buildTree* » nous allons décrire le fonctionnement de « *buildParagraphs* » qui permet de construire les imbrications au sein d'un paragraphe. En effet, au sein même d'un paragraphe nous construisons les imbrications qui correspondent aux phrases incluses dans le paragraphe. De même, nous construisons les plus petites entités textuelles « ETp » constituant une phrase. Ces « ETp » sont identifiés par l'intermédiaire de leurs délimiteurs qui peuvent être : des relations contextuelles (entre autre de discours) ou des signes de ponctuation (virgules, parenthèses,…).

La fonction « *buildParagraphs* » prend en paramètre la liste des paragraphes dont nous cherchons à construire les imbrications « Lps ». Pour chaque paragraphe « p » de « Lps », nous générons un nœud paragraphe nodeP. Ce dernier contient toutes les phrases qui lui sont associées. Avant d'ajouter une phrase à son paragraphe, la fonction « *buildPartsOfPhrase* » est lancée pour construire les différentes « ETp » (nous détaillerons cette fonction dans ce qui suit). Tous les nœuds de paragraphes générés sont insérés dans la liste « Lps' ». Cette liste est retournée par la fonction « *buildParagraphs* ».

```
Paragraph_Liste buildParagraphs(Lps : liste des paragraphes) {
  LSentences      : la liste de toutes les phrases
  nodeParagraph : Le type de nœud de la liste des paragraphes. Il est constitué de deux
                  champs, un pour le paragraphe « parag » et un pour contenir les phrases
                  associés à ce paragraphe « sentences »
  nodeSentences : Le type de nœud de la liste des phrases. Il contient les « ETp » associés
                  à cette phrase
  Lps'            : La liste des paragraphes à retourner
  Lps = OrderingParagraphs(Lps) ;
                  // les paragraphes sont ordonnés selon leur position dans le texte
  nodeParagraph nodeP;
  nodeParagraph p = Lps.first();
  While (p.notNull()) {
    nodeP = new nodeParagraph() ;
    L = selectSentencesAfter(p.start()) ∩ selectSentencesBefore(p.end());
    For each f∈ L do {
       nodeSentences nodeS = new nodeSentences() ;
       nodeS.set(buildPartsOfPhrase('', f)) ;  // les guillemets simples '' correspondes à une
chaîne de caractère vide
       nodeP.sentences.insert(nodeS) ;
    }//finForEach
    Lps'.insert(nodeP) ;
    p = p.next();
   }//finWhile
  return Lps';
}//finbuildParagraphs
```

La fonction « *buildPartsOfPhrase*» est une fonction récursive qui admet deux paramètres *listTraveled* et *listRemains*, correspondant respectivement à « *la liste des unités linguistiques parcourues d'une phrase* » et à « *la liste des unités linguistiques qui reste à parcourir dans la même phrase*». Les unités linguistiques peuvent être : des mots, des virgules, des points, etc.

L'idée principale de l'algorithme est de parcourir la phrase jusqu'à trouver un délimiteur permettant d'identifier une « ETp ». Ces délimiteurs peuvent être des ponctuations ou encore des relations contextuelles (entre autres les marqueurs de discours).

nodeETp : Le type de nœud de l'arbre des « ETp »

listTraveled, listRemains : les listes d'unités linguistiques

Tree *buildPartsOfPhrase***(**listTraveled : Liste d'unité linguistique, listRemains : Liste d'unité linguistique**) {**

```
    Tree tr ;
    If (listTraveled.notNull()) {
       nodeA = new nodeETp(listTraveled) ;
       tr.add(nodeA);
    }
    If (listRemains.notNull()) {
        unit ← listRemains.getFirst();
        listTraveled.setFirst(unit);
        While ((unit != 'comma') and (unit.hasNext()) and (unit != 'discourse markers')) {
             listTraveled.insert(unit) ;
             unit ← unit.next();
        // Parcourir f tant qu'il n'y a pas une virgule, un marqueur de discours ou fin de phrase
        }
        listRemains.setFirst(unit.next());
        listTraveled.insert(unit) ;
        Tree tr2;
        tr2 = buildPartsOfPhrase(listTraveled,listRemains);
        tr.add(tr2);
   }//endIf
 return tr
}// endBuildPartsOfPhrase
```

3 Manipulation sémantique : génération des annotations contextuelles

3.1 Identification des relations contextuelles

Les relations contextuelles devant être identifiées à ce niveau correspondent aux relations de discours. Nous avons recensé la liste des relations du discours dans la langue anglaise, sans prétendre que cette liste soit exhaustive (annexe 1).

Pour chacune des relations du discours recensées, nous avons généré une règle JAPE associée.

Algorithme 2 : construction des règles JAPE associées aux relations de discours

```
RR : une relation rhétorique (ou de discours)
RRs : la liste des relations rhétoriques
rules ←"";
For each RR ∈ RRs do {
  ruleName ← generateNameOf(RR) ;
      // supprimer tous les espace entre les mots existant dans une RR et concaténer ces
      mots pour les utiliser comme un nom de la règle JAPE générée.
  ruleHeader ← "Rule: JRule"+ generateNameOf(RR) + "(";
  endOfRule ← "):"+ ruleName +"-->:" + ruleName + ".RhetoricalRelation={kind="" + ruleName +"", rule=JRule"+ ruleName + "}";
  corpDeLaRègle ← "";
  For each word ∈ RR do {
    lemmatizedWord ← lemmeOf(word);
    bodyRule ← bodyRule + "{Token.lemma==""+ lemmatizedWord +""} ({SpaceToken})?";
  }//endFor
  rules ← rules + ruleHeader + bodyRule + endOfRule;
} //endFor
//end
```

Le principe de l'algorithme 1 est de générer les trois parties qui constituent la règle, à savoir : l'*entête*, le *corps* et la *fin*. Pour chaque relation rhétorique:

i) Récupérer les mots de la relation rhétorique en les concaténant pour les utiliser comme nom de la règle. Ce nom est utilisé dans l'*entête* et dans la *fin* de la règle.

ii) Lors de la génération du *corps* de la règle, les mots qui constituent les relations rhétoriques sont lemmatisés avant d'être utilisés. La comparaison se fait entre les mots lemmatisés du texte (exprimés dans la règle par *Token.lemma*) et les mots lemmatisés des relations rhétoriques (exprimés dans la règle par *lemmatizedWord*).

Par exemple, pour la relation rhétorique *« as well as »*, la règle associée s'écrit comme suit :

```
Rule:JRuleAsWellAs
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"well"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):AsWellAs -->
:AsWellAs.ContextualRelation= {kind = "aswellas", rule=JRuleAsWellAs}
```

Ces règles JAPE sont utilisées dans un *transducer* permettant d'identifier toutes ces relations de discours ainsi que leur position dans le texte : « début et fin » correspondant à « *StartNode* et *EndNode* ». Le listing qui suit donne le résultat engendré par la règle JAPE *JRuleAsWellAs* à la rencontre dans le texte d'une instance de la relation rhétorique *« as well as »*.

```
 <Annotation Id="12974" Type="RhetoricalRelation" StartNode="14596"
EndNode="14599">
  <Feature>
    <Name className="java.lang.String">rule</Name>
    <Value className="java.lang.String"> JRuleAsWellAs</Value>
  </Feature>
  <Feature>
    <Name className="java.lang.String">kind</Name>
    <Value className="java.lang.String">aswellas</Value>
  </Feature>
 </Annotation>
```

La balise nommée « Annotation »[19] est une balise générée par GATE pour chaque unité linguistique se trouvant dans le texte. Cette balise est générée aussi pour chaque résultat obtenu d'une identification dans le texte à travers une règle JAPE, comme l'exemple cité ci-dessus.

Un des problèmes auxquels nous avons été confrontés est la composition de certaines relations rhétoriques avec d'autres relations rhétoriques. Pour illustrer ce problème, nous prenons la même relation rhétorique que celle de l'exemple précédent *« as well as »*. Lors de

[19] A ne pas confondre avec les **annotations** sémantiques que nous cherchons à construire.

la génération des règles JAPE, à partir de la liste des relations rhétoriques, deux règles sont construites correspondant respectivement aux relations rhétoriques : *« as »* et *« as well as »*. Par conséquent, si la chaîne « as well as » est identifiée dans le texte, les deux règles vont identifier la présence de trois relations rhétoriques, *« as »*, *« as well as »* et *« as »* alors qu'il n'en existe qu'une seule.

Pour régler ce problème, nous avons ajouté une heuristique à prendre en compte lors de l'utilisation des relations rhétoriques identifiées dans le texte. Le but de cette heuristique est de sélectionner une et une seule relation rhétorique à utiliser :

- *Heuristique* 4. Si deux relations rhétoriques ont une partie du texte commune, alors sélectionner celle qui a la chaîne de caractères la plus longue.

3.2 Identification des classes, des relations et des instances

Nous rappelons que notre approche de génération des annotations contextuelles est automatisée et est basée sur l'utilisation d'une ontologie du domaine.

Afin de générer les triplets RDF associés à une annotation, il est nécessaire d'identifier d'abord les *ressources* (*classe* ou *concepts*), les *propriétés* et les *instances* dans le texte. Pour cela, nous proposons de construire des règles JAPE d'une manière automatique.

En effet, l'idée principale est de bénéficier de la propriété *rdfs:label*, dans un schéma RDFS, pour construire des règles JAPE qui détectent les différentes manifestations d'un *concept* (ou *propriété*) dans le texte. La propriété *rdfs:label* représente le nom lisible par un humain d'un *concept* (ou *propriété*) dans le texte.

L'algorithme qui suit a comme objectif de générer automatiquement des règles JAPE permettant d'identifier toute présence d'une classe dans le texte.

Algorithme 3 : Construction des règles JAPE associées aux classes d'une ontologie

ClassJapeRulesBuilding{

Classes ← *ListOfClass(ontology);* Rules ←"";

*// ListOfClass interroge l'ontologie en utilisant des règles **SPARQL** pour avoir la liste des classes avec leurs labels dans l'ontologie*

For each Class ∈ *Classes* **do** {

HeadRule←*HeadRuleBuilding*(GetNameOf(Class));

EndRule←*EndRuleBuilding*(GetNameOf(Class));

BodyRule ←"";

labels ← GetLabelsFrom(Class);

label ← Labels.first();

While label.hasNext() **do** {

BodyRule←BodyRule + "("+*BodyRuleBuilding*(label)+")" + "|";

//in Jape language "|" is the operator "or"

label ← label.Next();

}*//endFor*

BodyRule←BodyRule + "("+*BodyRuleBuilding*(label)+")";

// Pour le dernier label nous n'avons pas besoin d'ajouter l'opérateur « | »

Rules ← Rules +HeadRule + BodyRule + EndRule;

} *//endFor*

} *//End of ClassJapeRulesBuilding*

```
String HeadRuleBuilding(nameOfRule){
  Return "Rule: JRule"+ nameOfRule ;
}
String EndRuleBuilding(nameOfRule){
   Return "):nameOfRule-->:nameOfRule.Class={kind= "nameOfRule",
rule=JRulenameOfRule }";
}
String BodyRuleBuilding(label){
    Body ←""; Words ←GetWordsOf(label);
    For each w ∈ Words do {
       wlemma ← lemmaOf(w);
       Body←Body + "{Token.lemma=="" + wlemma +""} ({SpaceToken})?";
    }//endFor
  Return Body;
}
```

Après avoir interrogé l'ontologie pour récupérer la liste des *classes*, l'algorithme 3 parcourt cette liste et pour chaque classe :

i) Récupère le nom de la classe pour l'utiliser comme nom de règle. Ce nom est utilisé dans l'entête et dans la fin de la règle ;

ii) Construit l'entête et la fin de la règle ;

iii) Pour la même classe, récupère chaque *label* et lance la construction du corps de la règle avec *BodyRuleBuilding* en ajoutant un « | » (qui représente un 'ou') sauf pour le dernier *label* pour cette classe.

iv) Lors de la génération du corps de la règle, les mots qui constituent le *label* sont lemmatisés avant d'être utilisés. La comparaison se fait entre les mots lemmatisés du texte (exprimés dans la règle par *Token.lemma*) et les mots lemmatisés dans un *label* (exprimés dans la règle par *wlemma*).

D'une manière similaire à l' « *algorithme 3* » des règles JAPE sont générées automatiquement permettant d'identifier toute présence d'une *propriété* dans le texte.

3.3 Identification des valeurs candidates numériques

Pour identifier les valeurs candidates numérique associées aux propriétés dans les triplets, nous proposons une règle JAPE permettant d'identifier toutes les valeurs numériques présentes dans le texte. Cette règle s'écrit comme suit :

```
Rule:JRuleNumberValue (

  ({Token.kind == "number"}({SpaceToken})?(({Token.string == "."})|
   ({Token.string == ","})|({Token.string == "-"}))?({SpaceToken})?)+

): NumberValue -->: NumberValue.CandidateValueProperty = {kind
="NumberValue", rule=JRuleNumberValue}
```

Nous soulignons que les valeurs identifiées ne sont que des valeurs numériques candidates à être sélectionnées comme valeurs de propriétés. Cette sélection prend en compte certaines contraintes telles que « ces valeurs ne doivent pas être des indices numériques pour des titres ». Ceci veut dire que les heuristiques 1, 2 et 3 ne doivent pas être vérifiées. La sélection se fait par l'algorithme de génération d'annotation contextuelle.

3.4 Algorithme de génération des annotations contextuelles

La phase de manipulation de la structure du texte nous a permis d'identifier les entités textuelles de type « partie de phrase » que nous notons « **ETp** ». Ces « ETp » doivent être associées à une annotation sémantique. Nous rappelons que la génération d'une annotation sémantique donnée consiste à générer des triplets RDF dont les arguments sont identifiés dans un « ETp ». Cependant, vouloir générer des triplets RDF dans une entité textuelle limitée - telle qu'une « ETp » - engendre plusieurs problèmes et rend la tâche plus difficile. En effet, nous avons constaté certains problèmes que nous voulons citer :

- limiter le cadre de construction des triplets à l'« ETp » peut conduire à l'absence d'instance de classe, de propriété ou de valeur dans cette « ETp » et qui sont nécessaires à la construction des triplets ;
- le contraire pourrait arriver : il y aurait plusieurs instances de classes, propriétés ou valeurs. Dans ce cas, un problème d'ambigüité survient : quelle valeur correspond à quelle propriété qui correspond à quelle classe ?

Pour trouver des solutions satisfaisantes, nous avons mis en œuvre plusieurs heuristiques pour d'une part maximiser le nombre de triplets RDF générés et d'autre part minimiser le taux de triplets erronés.

- *Heuristique* 5. L'« ETp » est la première *entité textuelle* à exploiter mais pas l'unique pour la construction des triplets qui lui seront associés ;

- *Heuristique* 6. un triplet RDF généré est toujours associé à l'« ETp » où sa propriété a été identifiée ;

- *Heuristique* 7. Les entités textuelles englobantes sont choisies dans cet ordre : « ETp », phrase, paragraphe, phrase du titre, section, etc. ;

- *Heuristique* 8. seule l'identification d'une propriété déclenche la possibilité de générer un triplet ;

- *Heuristique* 9. lors de la génération d'un triplet associé à une propriété donnée, s'il n'y a pas d'instance de classe ou de valeur correspondant à la propriété en question dans l'entité textuelle courante, alors une entité textuelle plus englobante est utilisée ;

- *Heuristique* 10. utiliser le *domain* et le *range* d'une propriété pour choisir les instances de classes adéquates pour cette propriété;

- *Heuristique* 11. si une ambigüité persiste entre deux classes pour une propriété, choisir la plus spécifique dans l'ontologie (en terme de profondeur) ;

- *Heuristique* 12. si une ambigüité persiste entre deux instances compatibles avec la même propriété, choisir celle qui est la plus proche de la propriété dans le texte;

La majorité de ces heuristiques ont été regroupées en un seul algorithme permettant de construire les triplets RDF en prenant en compte d'une part l'aspect structure (imbrication hiérarchique) du texte et d'autre part l'aspect sémantique manifesté par l'ontologie du domaine.

Nous soulignons que nous supposons l'existence d'au plus un seul « *domain* » et un seul « *range* » pour chaque propriété. Actuellement, nous ne traitons pas le cas de plusieurs « domain » ou de plusieurs « range » pour une propriété. Nous discutons ce cas plus en détail dans le chapitre V, section 8.

Algorithme 4: génération des triplets RDF associés aux « ETp »

ETp : entité textuelle la plus petite déduite à partir de la hiérarchie de la structure du texte;

ETs : ensemble d'ETp;

ET_max: le plus grand ET considéré *(par exemple : le document en entier, une section, un paragraphe,…);*

AS : annotation sémantique que nous cherchons à générer et à associer aux ET;

Input : ETs, ET_max;

For each ET ∈ ETs do {

ET'← ET;

P ← ET'.getProperties();

AS ← Null;

While ((ET' ≤ ET_max) and (P ≠ ∅)) {

For each pj ∈ P **do** {

AS_temps ← *RDFTriplesGenerating*(pj, ET');

If (!empty(AS_temps)) {

P←P-{pj}; *// si un triplet est généré pour la propriété pj, alors retirer pj*

// de l'ensemble de propriété P

AS ← AS + AS_temps;

}

} *//endFor*

If (P ≠ ∅) {

ET' ← ET'.IncreaseTextualElement();

// choisir un élément textuel plus englobant. De la « ETp » à la phrase, ensuite

// le paragraphe, ensuite la phrase du titre, etc.

} *//endIf*

} *//endWhile*

If(!empty(AS))**then**{AssociateObjects(ET,AS)}

}*//endFor*

Cet algorithme admet en entrée l'ensemble des ETp ainsi que l'entité textuelle la plus englobante considérée ET_max. Nous cherchons à générer le triplet RDF pour chaque propriété « pj » identifiée dans chaque entité textuelle ET. Si la génération du triplet échoue, une autre entité textuelle plus englobante est choisie. Nous répétons ceci jusqu'à aboutir à la génération du triplet associé à la propriété pj ou jusqu'à atteindre l'entité textuelle maximale englobant cette ET considéré sans construire de triplet.

La fonction « *RDFTriplesGenerating* » de l'Algorithme 4

```
RDF_triplets RDFTriplesGenerating(p, ET) {
AS ← Null;
C ← set_of_classes_in(ET) // l'ensemble des classes identifiées dans l'ET
c1 ∈ C ∩ domain(p)//c1 est une classe qui vérifie une contrainte de « domain » pour la
        // propriété « p ». Si plusieurs classes vérifient cette contrainte, choisir la plus
        //spécifique dans l'ontologie ainsi que la plus proche par rapport à la propriété « p »
        //dans le texte. (Heuristiques 10,11 et 12)
I ← set_of_values_in(ET)
inst ∈ I // Sélectionner parmi les valeurs de l'ensemble « I » la valeur inst qui est la plus
          // proche de « p » dans le texte.
c2 ∈ C ∩ range(p) ;
        // c2 est une classe qui vérifie une contrainte « range » pour la propriété « p » autre
        // que « literal ». Si plusieurs classes vérifient cette contrainte, choisir la plus
        // spécifique dans l'ontologie ainsi que la plus proche par rapport à la propriété « p »
        // dans le texte. (Heuristiques 10,11 et 12)
If (!empty(c1)) {subject ← id(c1.getName());}
        // « id » est une fonction qui ajoute un identifiant
If (p.range ="literal") { If (!empty(inst)) {object ← inst ;}
} else { If (!empty(c2)) {object ← id(c2.getName());}
}
If (!empty(subject) and !empty(object)) {
   // typage d'instance : déclarer l'instance de la classes c1 et c2 avec la propriété rdf:type
   AS ← AS + GenRDFInstanceOfClass(subject, "rdf:type", c1) ;
   If (p.range != "literal") { AS←AS + GenRDFInstanceOfClass(object, "rdf:type", c2);}
   AS ← AS + GenRDFtriples(subject, p, object) ;
}
```

La génération des triplets avec la fonction « *RDFTriplesGenerating*» consiste à faire correspondre une propriété donnée à « son instance de classe » et à « sa valeur » appropriées.

Les heuristiques 10, 11 et 12 utilisées dans l'algorithme pour sélectionner la classe c1 du « *sujet* » peuvent s'écrire d'une manière plus formelle de la façon suivante :

Etant donnés :

p : la propriété dont nous cherchons à générer le triplet RDF associé ;

C : l'ensemble des classes identifiées dans un ET ;

C_{domain} : l'ensemble des classes (et sous classes) de C vérifiant la contrainte *domain* de « p ».

$C_{dist\_ontologique}$: l'ensemble des classes de C_{domain} les plus spécifiques dans l'ontologie (heuristique 11) ;

C_{dist_text} : l'ensemble des classes de $C_{dist\_ontologique}$ les plus proches de « p » dans le texte (heuristique 12).

La classe c1 de la fonction « *RDFTriplesGenerating* » est choisie parmi les éléments de l'ensemble C_{dist_text}.

Nous définissons C_{domain} et $C_{dist\_ontologique}$ comme suit :

$$C_{domain} = \{ c \in C / c \leq p.domain \}$$

$$C_{dist\_ontologique} = \{ c' \in C_{domain} / Maximiser [\delta_{Onto} (Racine, c')] \}$$

Nous cherchons à sélectionner les classes de « C_{domain} » ayant la distance « δ_{Onto} » maximale dans l'ontologie par rapport à la racine ontologique. C'est-à-dire, choisir les classes de « C_{domain} » les plus spécifiques dans l'ontologie. La définition de $C_{dist\_ontologique}$ de cette manière nous permet de couvrir les deux cas de figures pouvant exister dans l'ontologie entre deux classes données :

- Le cas où une classe Ci est l'ancêtre d'une autre classe Cj. Dans ce cas de figure et étant donné que le descendant est celui qui maximise la distance avec la racine ontologique par rapport à son ancêtre, c'est le descendant qui va être sélectionné (Cj) ;
- Le deuxième cas concerne deux classes dont l'une n'est pas l'ancêtre de l'autre. Dans ce cas de figure, ayant au moins un ancêtre commun (racine ontologique), la classe qui a la distance maximale par rapport à la racine ontologique sera sélectionnée.

$$C_{dist\_text} = \{ c1 \in C_{dist\_ontologique} / Minimiser [\delta_T (p, c1)] \}$$

Dans l'ensemble C_{dist_text} nous sélectionnons les classes de $C_{dist\_ontologique}$ qui ont la distance (δ_T) minimale dans le texte par rapport à la propriété p. Ce que nous appelons ici par distance dans le texte (δ_T) est la distance par rapport à la « position dans le texte ».

Etant donnés:

wp : le mot identifié dans le texte et associé à la propriété p

ws : le mot identifié dans le texte et associé à la classe c1

$$\delta_T(p, c1) = \begin{cases} wp_{début} - ws_{fin} & si \quad ws_{fin} < wp_{début} \\ ws_{début} - wp_{fin} & si \quad ws_{début} > wp_{fin} \end{cases}$$

Où *début* et *fin* représentent respectivement le début et fin de la position d'un mot dans le texte. Il est à noter que pour tout mot : $\forall w / length(w) \neq 0, \ 0 < w_{début} < w_{fin}$

Avec la définition de C_{domain}, $C_{dist\_ontologique}$ et C_{dist_text}, nous avons expliqué l'utilisation de la contrainte *domain* et des heuristiques 10, 11 et 12. Cependant, même avec ces heuristiques qui réduisent le champ d'ambigüité dans l'attribution des classes à une propriété donnée, des cas rares restent à traiter. Nous avons utilisé le terme « rares » parce que tout au long de nos expérimentations ces cas ne se sont pas manifestés.

Nous exposons dans ce qui suit le cas auquel nous faisons allusion:

Identification, dans une même partie de phrase « ET » dans le texte, d'une propriété « p » et de deux instances qui correspondent à deux classes ci et cj candidates pour le sujet de « p ». Ces deux classes ont une distance « δ_{Onto} » identique et maximale par rapport à la racine ontologique. ci et cj ont une distance identique « δ_T » par rapport à « p » vu qu'elles se positionnent respectivement avant et après la position de « p » dans le texte.

La question qui se pose est: quelle est la classe parmi ci et cj qui est la plus appropriée pour être associée à la propriété « p » ?

Nous proposons deux solutions pour pallier ce problème :

- La première est de faire un choix aléatoire vu que ce cas de figure ne se présente que rarement et que traiter ce cas rendrait l'algorithme de génération plus complexe et plus coûteux en temps de calcul.
- La deuxième proposition est de mettre la génération du triplet associé à la propriété en attente jusqu'au signalement qu'une de ces deux classes « ci ou cj » a été utilisée dans la génération d'un autre triplet associé à une propriété. La classe restante sera sélectionnée pour le triplet de la propriété p. Cette proposition ne paraît pas dans l'algorithme proposé

ci-dessus pour une raison de simplification. Dans le cas où aucune de ces classes n'a été utilisée dans la génération d'un autre triplet, le problème reste non résolu et la première solution est adoptée.

Ces propositions restent en concordance avec notre première hypothèse qui est de proposer des solutions automatiques en premier plan avec des compromis sur la précision du résultat.

Nous soulignerons à toute étape de l'approche d'extraction la possibilité d'intégration, si elle existe, d'un module avec une intervention humaine (expert du domaine) ou une proposition d'amélioration, comme ce fut le cas dans cette partie de l'algorithme.

Nous revenons sur l'« Algorithme 4 » et plus spécialement sur la sélection des valeurs « inst » de type « literal » dans la fonction « *RDFTriplesGenerating* ». Actuellement, le traitement est restreint aux valeurs littérales de type « float » (numérique). Par conséquent, l'ensemble « I » regroupe les valeurs numériques ayant été identifiées dans le texte dans une entité textuelle donnée et candidates à être des valeurs numériques de propriétés pour les triplets générés.

« *Inst* » est utilisé dans le cas où le type de la valeur de la propriété est un numérique. La valeur *inst* est choisie parmi les éléments de l'ensemble « I' » qui regroupe les valeurs numérique de l'ensemble « I » ayant une distance minimale avec la propriété en question « p ».

$$I' = \{ \exists\, i \in I \,/\, Minimiser\, [\, \delta_T\, (p, i)] \}$$

Dans le cas où il y aurait deux valeurs numériques « i » et « i' » se positionnant respectivement avant et après la propriété « p » dans le texte, nous avons opté pour la valeur se trouvant après la propriété. Ce choix est conduit par des observations empiriques sur les textes de l'expérimentation. Nous avons observé aussi que c'est fortement lié à la langue du texte et à la manière dont les phrases sont formulées. Nous ne nions pas la nécessité de faire une étude plus poussée à ce niveau pour argumenter le choix d'une valeur par rapport à une autre selon sa position linguistique ou encore l'intervention d'un expert du domaine pour trancher. Cependant nous avons une fois de plus opté pour un compromis entre l'automatisation et la précision. Nous soulignons que ce choix n'empêche pas de faire intervenir un expert du domaine pour trancher sur un type de documents particulier ou sur un type de rédaction particulière.

Après avoir sélectionné la classe « c1 » comme « *sujet* » de la propriété « p », le choix de l'objet dépend du « range » de la propriété « p ». En effet, deux cas se présentent :

- Dans le cas où le « *range* » de la propriété « p » est un littéral et plus particulièrement un « *float* », nous affectons la valeur numérique « inst » à l'objet de « p ».
- Dans le cas où le « *range* » de la propriété « p » est une classe, nous affectons l'instance de la classe « c2 » à l'objet de « p ».

Nous pouvons ainsi générer le triplet « sujet, p, objet » avec la fonction « *GenRDFtriples* ». Un autre triplet est généré pour le typage de l'instance de la classe c1 et un autre triplet pour le typage de l'instance de la classe c2 dans le cas où le range de « p » est un non « literal ».

Nous soulignons que les critères de sélection de la classe c2 pour l' « *objet* » sont identiques à ceux utilisés pour c1 (le « *sujet* ») à savoir les Heuristiques 10, 11 et 12. Les distances définies plus haut sont les mêmes à être utilisées. Ce qui change, c'est l'utilisation de la contrainte « *range* » au lieu du « *domain* ».

4 Conclusion

Dans le quatrième chapitre nous exposons notre approche de génération d'annotations sémantiques contextuelles à partir de texte. L'architecture générale de l'approche est constituée de deux principales phases : manipulation textuelle et manipulation sémantique.

La première phase consiste à reconstruire la structure physique d'un texte brut donné en entrée. En premier lieu une chaine de traitement est lancée sur le texte pour identifier les différents éléments textuels : tokens (mots, chiffre, etc.), phrases, paragraphes, etc. Des indicateurs numériques sont utilisés pour identifier les titres. Les imbrications sont ensuite construites entre : titres, paragraphes, phrases, etc. Les algorithmes permettant ces manipulations de la structure physiques du texte sont détaillés. A partir des éléments textuels identifiés et leurs imbrications des uns par rapport aux autres, nous déduisons les relations textuelles issues de la structure physique du texte.

La deuxième phase consiste à identifier les relations contextuelles de type rhétorique en utilisant des règles JAPE [Cunningham et al., 2002] générées automatiquement. De même, en utilisant des règles JAPE générées automatiquement, nous identifions les instances de classes et de propriétés. Ensuite, nous générons des annotations sémantiques sous forme de « triplets RDF » en associant un sujet et un objet pour les propriétés identifiées tout en prenant en compte la position des sujets et objets dans les imbrications de la structure physique du texte. Par la suite les annotations générées sont reliées par les relations contextuelles formant ainsi des annotations sémantiques contextuelles.

Tous les algorithmes décrivant les étapes de ces deux phases sont exposés dans ce chapitre.

Chapitre V : Implémentation du système CEEMA-T

1 Introduction

Dans ce chapitre, nous abordons l'implémentation de notre approche CEEMA que nous avons nommée « système CEEMA-T[20] ». Nous exposerons aussi les différents choix techniques adoptés pour son développement.

2 Architecture générale de CEEMA-T

Dans cette partie, nous aborderons les différentes étapes d'extraction d'information et génération des annotations contextuelles avec les choix techniques adoptés pour chaque étape.

D'abord les rdfs:label des classes et propriétés sont récupérés de l'ontologie. Ensuite, un module de génération des règles de grammaire récupère, d'une part, les rdfs:label pour générer des règles JAPE correspondant aux classes et aux propriétés, et d'autre part, une liste des relations contextuelles pour générer des règles JAPE qui leur sont associées.

Les règles JAPE générées avec le module de « génération des règles de grammaire » peuvent être ajoutées avec d'autres règles construites manuellement. Les règles JAPE sont appliquées, avec d'autres traitements, aux documents textuels (bruts) via une chaîne de traitement « pipeline » GATE. Ce dernier génère des documents XML (fichiers de sortie GATE) complétés avec les informations issues de la chaîne de traitement. Une suite d'algorithmes est appliquée sur ces documents XML pour construire les imbrications (la hiérarchie des documents) entre les entités textuelles : titres, sous-titres, paragraphes, phrases, parties de phrase. Après la construction des imbrications, l'étape de génération des annotations contextuelles est lancée avec une suite d'algorithmes appliqués aux documents XML.

La Figure 18 illustre la séquence avec laquelle les différentes étapes sont lancées. Le module « Dev » représente notre module développé avec Java.

[20] CEEMA-T: Contextual Extraction and Exploitation Method of Annotation - Tool

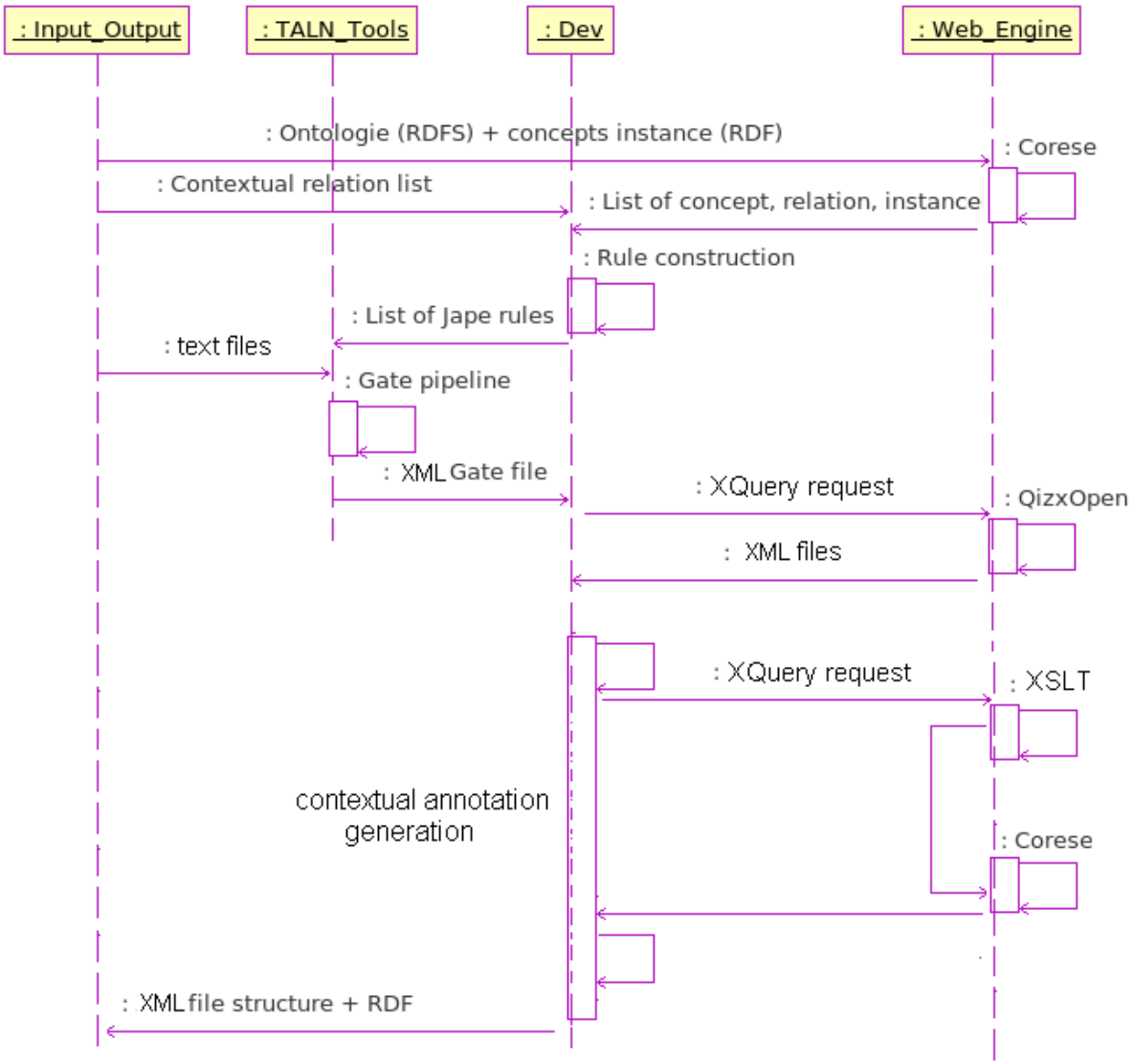


Figure 18 La séquence d'exécution des différentes étapes de CEEMA-T

Cette architecture a été implémentée dans l'outil CEEMA-T qui prend en entrée une ontologie du domaine et des documents textuels (brut) pour la génération d'annotations sémantiques contextuelles. Dans ce qui suit, nous détaillons les fonctionnalités et les composantes de cet outil.

3 Outils de TALN utilisés

3.1 GATE

Nous avons eu l'occasion dans « le Chapitre I, section 2.1 » d'aborder un descriptif de la plateforme GATE. Dans cette partie nous abordons comment GATE a été utilisé dans l'outil CEEMA-T.

GATE peut être utilisé soit comme un environnement de développement, soit comme une bibliothèque. Dans notre cas, nous l'avons utilisé des deux façons possibles.

L'utilisation de GATE comme environnement de développement au travers des ressources développées par ses concepteurs est la plus simple. L'environnement de développement GATE met à disposition des développeurs une interface graphique permettant aux utilisateurs:

- de créer de nouvelles ressources
- de paramétrer des ressources disponibles
- d'appliquer les ressources (crées ou déjà disponibles) aux textes au sein d'une chaîne de traitement.

Nous avons utilisé l'interface graphique de GATE pour tester nos modules d'extraction de concepts, relations, instances, indices numériques, etc. pour visualiser les résultats.

Notre deuxième utilisation de GATE consiste à tirer parti des ressources disponibles dans la plateforme embarquée dans des applications autonomes. Ainsi, il est possible de se passer de l'interface graphique de GATE et de traiter un texte dans un programme autonome hors de l'environnement de développement GATE.

Nous avons ainsi bénéficié, d'une part, de la simplicité d'élaboration des chaînes de traitement via l'interface graphique GATE, et d'autre part, de la bibliothèque de GATE afin d'intégrer nos différents modules et afin de les appliquer en chaîne sur les textes d'une manière embarquée.

3.2 JAPE : un langage d'expression de grammaires pour le TALN

Le langage JAPE (Java Annotation Patterns Engine), est une variante du standard CPSL[21] adapté au langage de programmation Java [Cunningham et al., 2002]. Les grammaires écrites avec JAPE ont pour but, une fois appliquées sur le texte, de permettre d'y ajouter des informations sous forme d'annotations.

Une grammaire de JAPE est constituée d'un ensemble de « phases », chaque phase étant un ensemble de « règles » sous forme de « patron et action ». Dans une règle, si la condition formulée par les patrons est satisfaite, alors une action pourra être déclenchée. Un exemple d'action pourra être l'attribution d'une étiquette d'annotation. La partie gauche d'une règle contenant des patrons doit être écrite en JAPE mais la partie droite, contenant les actions, peut être écrite en JAPE ou en JAVA.

Un exemple de grammaire JAPE est présenté ci-dessous permettant de détecter les adresses IP dans le texte.

```
Rule: IPAddress
  ({Token.kind == number, Token.length <= 3} {Token.string == "."}
   {Token.kind == number, Token.length <= 3} {Token.string == "."}
   {Token.kind == number, Token.length <= 3} {Token.string == "."}
   {Token.kind == number, Token.length <= 3})
:ipAddress --> :ipAddress.Address = {kind = "ipAddress"}
```

Le langage JAPE peut être utilisé en tant que transducteur dans GATE. C'est, d'ailleurs, de cette manière qu'il a été utilisé dans l'implémentation de notre approche.

[21] A Common Pattern Specification Language

3.3 TreeTagger

TreeTagger [Schmid, 1994] est un outil développé au sein de l'institut de linguistique computationnelle de l'université de Stuttgart. TreeTagger est un outil d'étiquetage des textes (en français et en anglais). Il spécifie la catégorie syntaxique pour chaque mot et indique son lemme.

TreeTagger utilise la construction récursive d'arbres de décisions avec un calcul de probabilité pour estimer la catégorie grammaticale d'un mot.

Le Tableau 2 représente un exemple de sortie de TreeTagger pour l'entrée suivante « The TreeTagger is easy to use. ».

Mot	Catégorie	Signification	Lemme
The	DT	Article	The
TreeTagger	NP	Nom propre	TreeTagger
is	VB	Verbe conjugué au présent	Be
easy	JJ	Adjectif	Easy
to	TO	TO : catégorie spéciale	to
use	VBZ	Verbe à l'infinitif	Use
.	SENT	Ponctuation	.

Tableau 2 Résultat de TreeTagger sur l'exemple de la phrase précédente

Nous avons utilisé TreeTagger, d'une part, comme traducteur (wrapper) intégré dans GATE, et d'autre part, comme module intégré dans notre application. Nous avons utilisé TreeTagger pour bénéficier de l'étiquetage grammatical et la lemmatisation des textes. Les informations concernant chaque mot sont intégrées dans la structure XML du document de sortie de GATE.

GATE propose un étiqueteur grammatical par défaut mais ce dernier ne calcule pas la forme lemmatisée des mots. Ceci nous a poussé à utiliser TreeTagger car la forme lemmatisée des mots est une information indispensable dans notre processus.

4 Moteurs utilisés

4.1 Corese (Conceptual Resource Search Engine)

Corese (COnceptual REsource Search Engine) [Corby et al., 2004, 2006] est un moteur de recherche sémantique proposé par l'équipe EDELWEISS.

Corese est un moteur RDF basé sur le formalisme des graphes conceptuels (GC) [Sowa, 1984]. Il permet le traitement de RDFS, une partie d'OWL Lite et des énoncés RDF (RDF

statements) en s'appuyant sur le formalisme des GC. Il peut effectuer des requêtes SPARQL[22] et exécuter des règles sur des graphes RDF.

Corese traduit les classes et relations RDFS en types de concept et en type de relation et les annotations RDF en base de GC. Le Tableau 3 regroupe les différentes traductions entre RDF/RDFS et le formalisme des GC.

RDF/RDFS	GC	
rdfs:Class	Type de concept	Représentation des ontologies
rdf:Property	Type de relation	
rdfs:domain, rdfs:range	Signature des relations	
resource	Concept	Représentation des annotations
resource anonyme	Concept générique	
propriété	Relation	

Tableau 3 Correspondance entre RDFS/RDF et GC

La Figure 19 représente une vue globale sur le fonctionnement interne de Corese. Comme nous pouvons le constater, Corese prend comme entrées des schemas RDFS (ontologie), des descriptions RDF (annotations), des règles et SPARQL (requête). Il construit un graphe conceptuel avec RDFS, RDF, et les règles (en utilisant des inférences). Corese fait des projections[23] du graphe de la requête sur les graphes issus de « l'ontologie + annotations + règles ». On trouve enfin les résultats des graphes conceptuels transformés en RDF/S, ou sous forme de « *XML Result Format* » (un exemple de ce format est donné dans Figure 21).

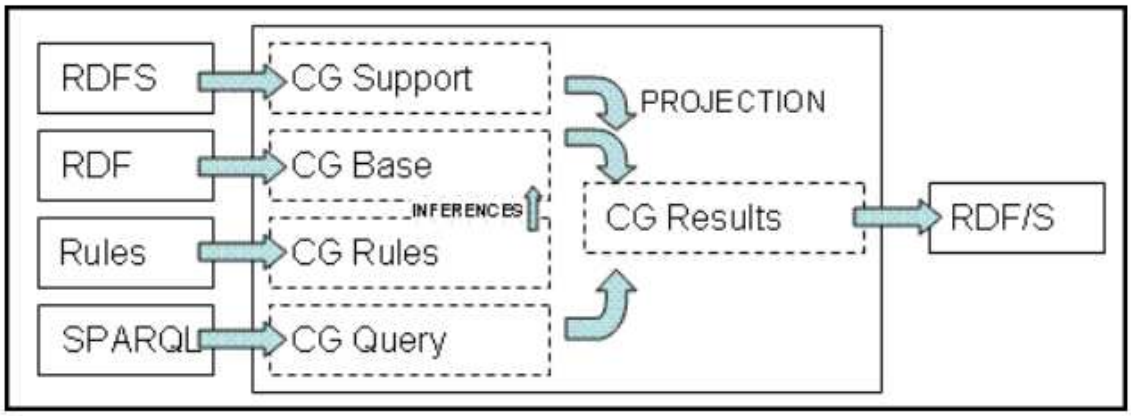


Figure 19 Principe général de Corese

[22] http://www.w3.org/TR/rdf-sparql-query/

[23] Opération de projection dans les GC

Dans la Figure 20 nous donnons un exemple d'une requête SPARQL et le résultat proposé par Corese est dans la Figure 21.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?book ?title
WHERE { ?book dc:title ?title }
```

Figure 20 Exemple de requête SPARQL

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="title"/>
  </head>
  <results>
     <result>
      <binding
name="book"><uri>http://example.org/book/book1</uri></binding>
      <binding name="title">SPARQL</binding>
     </result>
  </results>
</sparql>
```

Figure 21 Résultat proposé par Corese de la requête de l'exemple «Figure 20»

Corese est disponible[24] en bibliothèque de fonctions (API) utilisable dans les applications du Web sémantique.

4.2 Qizx/open

Qizx/open[25] est la première version de Qizx, disponible en « open source » depuis 2003. Qizx/open implémente les spécifications du langage de requête XML « XQuery » qui est une recommandation du W3C depuis janvier 2007.

Qizx/open offre la possibilité de regrouper des documents XML en « collections » (vu aussi comme un entrepôt) sur lesquelles il est possible de lancer des requêtes XQuery. Le résultat est retourné sous forme d'un fichier XML.

[24] http://www-sop.inria.fr/edelweiss/software/corese/

[25] Axyana. Qizx/open. Une implementation Java opensource XQuery: http://www.axyana.com/qizxopen, 2006.

L'utilisation de Qizx/open est possible soit via une interface graphique mise à disposition, soit via une bibliothèque de fonctions (API).

5 Génération des règles de grammaire JAPE

Dans cette phase, le but est de construire des règles JAPE permettant d'identifier les différents éléments nous permettant de générer par la suite des annotations contextuelles (voir Figure 22).

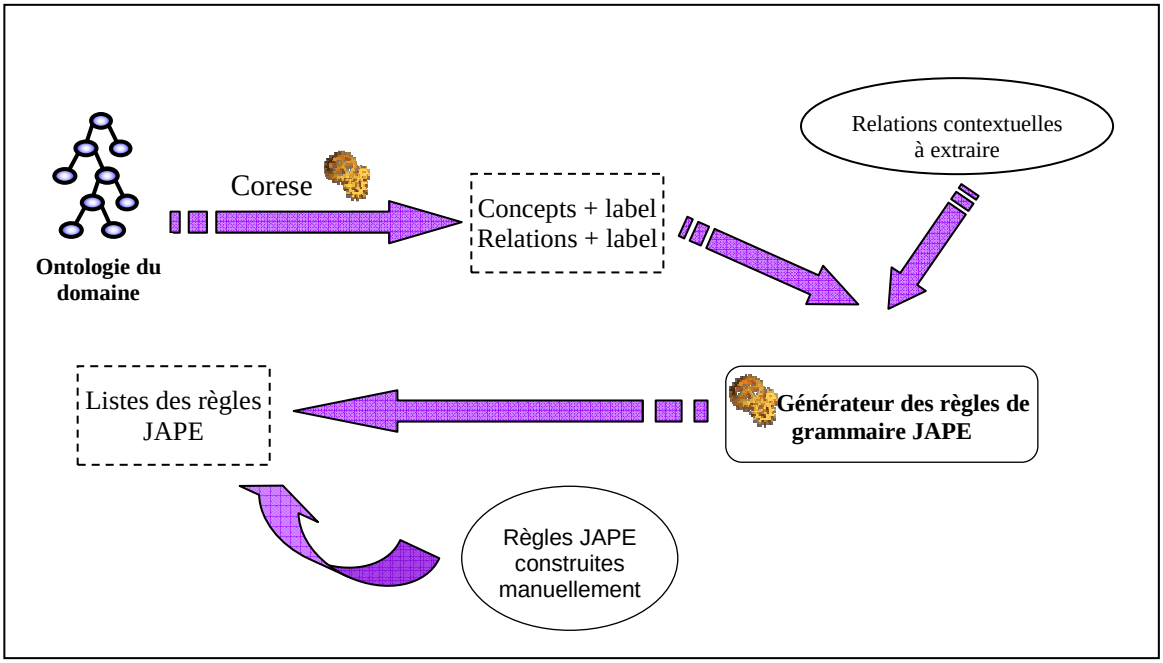


Figure 22 Module de génération des règles JAPE

5.1 Les règles d'identification des éléments de l'ontologie

La première étape exécutée dans notre approche (exposée dans le chapitre VI) est l'identification des éléments de l'ontologie: classes et relations.

Cette étape est entièrement automatisée. En effet, l'ontologie est interrogée via le moteur Corese avec une requête SPARQL pour retourner les *rdfs:label* des classes et des propriétés. Ces labels sont le point d'entrée de l'« Algorithme 3 » permettant la construction des règles JAPE associées aux classes. Idem pour l'algorithme permettant la construction des règles JAPE associées aux propriétés. La Figure 23 représente la requête SPARQL lancée via Corese pour récupérer les *rdfs:label* des classes dans l'ontologie. La requête SPARQL lancée pour récupérer les *rdfs:label* des propriétés est similaire à celle présentée dans la « Figure 23 ». La Figure 24 représente un exemple d'une partie du résultat de la requête SPARQL (exemple de la Figure 23).

```
select ?class
xsd:string(?label) as ?LabelString
where
{
       ?class rdf:type rdfs:Class
       ?class rdfs:label ?label.
       FILTER ( lang (?label) = 'en')
}
ORDER BY ?class
```

Figure 23 Requête SPARQL retournant les rdfs:label des classes dans l'ontologie

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head>
<variable name='class'/>
<variable name='LabelString'/>
</head>
<results>

<result>
<binding name='class'>
<uri>http://www.inria.fr/2008/03/24/estanda.rdfs#LinerMill</uri>
</binding>
<binding name='LabelString'><literal
datatype='http://www.w3.org/2001/XMLSchema#string'>Liner
</literal>
</binding>
</result>

<result>
<binding name='class'>
<uri>http://www.inria.fr/2008/03/24/estanda.rdfs#LinerMill</uri>
</binding>
<binding name='LabelString'><literal
datatype='http://www.w3.org/2001/XMLSchema#string'>Liner Of
Mill</literal></binding>
</result>

</results>
</sparql>
```

Figure 24 Exemple de résultats de la requête de la « Figure 23 » pour la classe « *Liner Mill* »

Après la récupération des *rdfs:label*, les règles JAPE associées à chaque classe sont générées.

La Figure 25 représente un exemple de règle JAPE générée par l'« Algorithme 3 » appliqué sur la classe « *Liner Mill* » de la Figure 24.

```
Rule:JRuleLinerMill (


(( ({Token.lemma == "liner"})| ({Token.string == "Liner"})|
({Token.string == "liner"})| ({Token.lemma == "Liner"}))({Token.kind
== "punctuation"})? ({SpaceToken})?) |


(( ({Token.lemma == "liner"})| ({Token.string == "Liner"})|
({Token.string == "liner"})| ({Token.lemma == "Liner"}))({Token.kind
== "punctuation"})? ({SpaceToken})?( ({Token.lemma == "of"})|
({Token.string == "Of"})| ({Token.string == "of"})| ({Token.lemma ==
"Of"}))({Token.kind == "punctuation"})? ({SpaceToken})?(
({Token.lemma == "mill"})| ({Token.string == "Mill"})| ({Token.string
== "mill"})| ({Token.lemma == "Mill"}))({Token.kind ==
"punctuation"})? ({SpaceToken})?)


): LinerMill -->: LinerMill.Class = {kind ="LinerMill",
rule=JRuleLinerMill }
```

Figure 25 un exemple de règle JAPE générée par l'« Algorithme 3 » appliqué sur la classe «Liner Mill »

5.2 Les règles d'identification des relations contextuelles

Cette étape consiste à récupérer les relations contextuelles à partie d'une liste prédéfinie et de générer les règles JAPE pour chaque relation en appliquant l'« Algorithme 2 ». Un exemple de règle JAPE a été déjà présenté dans le « chapitre VI, section 3.1 ».

5.3 Les règles JAPE construites manuellement

En plus de ces règles engendrées automatiquement, nous avons écrit une règle JAPE additionnelle incorporée dans CEEMA-T. Il s'agit de la règle permettant d'identifier des valeurs numériques candidates à être sélectionnées comme valeurs de propriétés. Cette règle JAPE est déjà présentée dans le « chapitre VI, section 3.3 ».

Par ailleurs, il est tout à fait possible de spécifier d'autres règles JAPE dans CEEMA-T pour un type particulier d'instance (de classe) ou de relation.

6 Chaîne de traitement avec GATE

CEEMA-T utilise la plate-forme GATE pour intégrer plusieurs modules et les appliquer en cascade sur le texte brut (voir Figure 26). Notre processus de traitement avec GATE est constitué de quatre étapes.

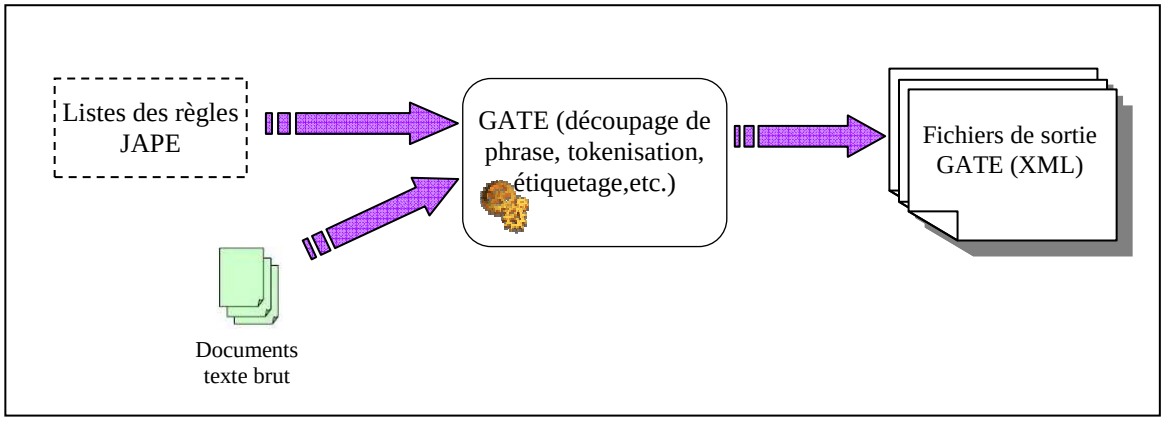


Figure 26 Processus de traitement en cascade sur le texte avec GATE

6.1 Prétraitement

Notons ici qu'une phase de prétraitement est requise dans CEEMA-T. En effet, nous n'utilisons pas les documents textuels sous leurs formats de publication (pdf, ps, etc.), mais une conversion est faite vers un format textuel brut. La seule condition requise pour cette conversion est qu'elle garde la structure physique du texte. Par exemple, conserver les retours à la ligne, les numéros des titres, les distinctions entre paragraphes, etc.

6.2 Analyse morpho-syntaxique des textes

Manipuler automatiquement des textes écrits en langue naturelle nécessite souvent une analyse préliminaire des entités constituant ces textes. Ceci est valable quelle que soit la langue utilisée. Cette phase d'analyse a pour but de récupérer toute information caractérisant le comportement d'un mot dans son contexte d'énonciation.

Dans cette phase, notre analyse comprend trois étapes:

- *Le découpage du texte en phrases* : ce découpage permet de repérer le début et la fin de chaque phrase (les limites des phrases) pour un éventuelle traitement spécifique (aux phrases) ;
- *L'identification des entités linguistiques de base « tokenisation » (tokens)*: ceci nous permet d'identifier aussi la morphologie de ces tokens (nombre, ponctuation, etc.) ainsi que la racine de chaque entité (lemmatisation) ;
- *L'étiquetage grammatical* : l'étiquetage consiste à affecter à chaque *token* une catégorie d'ordre grammatical (Verbe, Nom, Adjectif, etc.).

Les deux premières étapes sont implémentées dans CEEMA-T en utilisant, respectivement, les modules de GATE : le « *Sentence Splitter* » et « l'*English Tokeniser* » (voir Figure 27). La troisième étape (l'étiquetage grammatical) est implémentée en utilisant TreeTagger comme module.

6.3 Détection des propriétés, des instances et des relations contextuelles

Nous soulignons que le but de cette phase est de repérer dans le texte les propriétés, les instances de classe, les valeurs candidates pour les propriétés et les relations contextuelles.

Nous rappelons que les règles JAPE permettant de les identifier sont déjà construites (voir section 5). Ces règles JAPE sont appliquées au texte en utilisant un « JAPE transducer » proposé dans GATE (voir Figure 27).

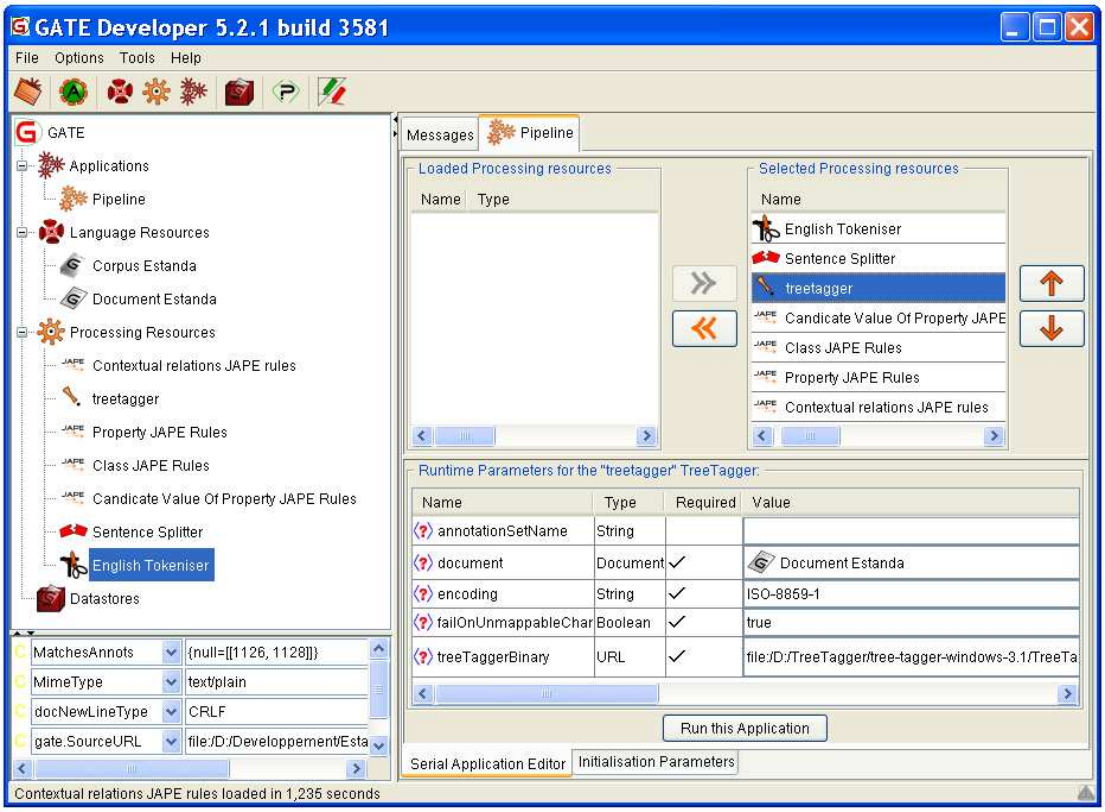


Figure 27 Vue de GATE avec les ressources (modules) utilisées dans le processus de traitement

7 Construction de la structure physique du texte

Dans cette phase, le but est de reconstituer les imbrications entres les différents éléments du texte (titres, sous-titres, paragraphes, phrases, parties de phrase). Etant donné que les documents de sortie de GATE sont au format XML, il nous a paru plus approprié d'implémenter les algorithmes de constructions des imbrications avec un langage évolué adapté, à savoir, le langage de requête XQuery en utilisant le moteur Qizx/Open (voire la Figure 28). Ainsi, les algorithmes et les heuristiques implémentés avec le moteur Qizx/Open dans cette phase sont :

- *Les heuristiques d'identification des titres*, nous rappelons que dans ces heuristiques (décrites dans le Chapitre IV section 2.1) le but est de sélectionner parmi les valeurs numériques (identifiés via la règle de grammaire JAPE décrite dans le chapitre IV, section 2.1) celles qui représentent les numéros des titres ;
- *L'« Algorithme 1» de construction des imbrications hiérarchiques du document*, qui est basé sur la récursivité pour construire les imbrications (décrit dans le Chapitre IV section 2.2).

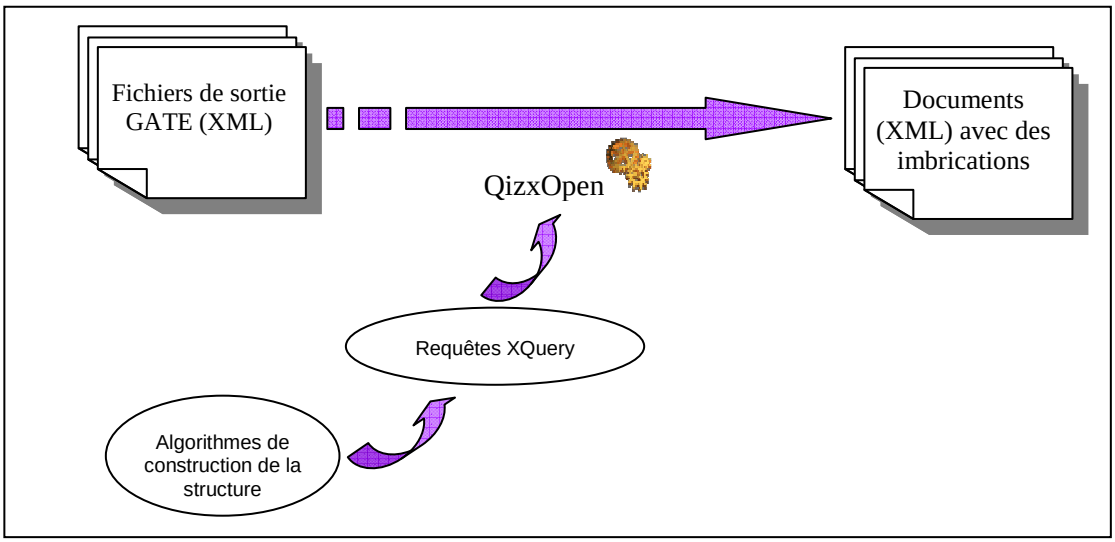


Figure 28 Phase de construction des imbrications dans le texte

```
(: --------------- fonction : CountTitreInInterval ----------:)

declare function local:CountTitreInInterval($start as xs:integer,
$end as xs:integer) as xs:integer
{
     count( for $b in //Annotation[@Type="NumberTitre"]
      where (xs:integer($b/@StartNode)) >= $start
      and (xs:integer($b/@EndNode)) <= $end
      return $b)
};
(: ---------------   fonction : SentenceIsTitre   ----------:)
declare function local:SentenceIsTitre($paragraph as node(),
$sentence as node()) as xs:integer
{
  let $StartParagraph := (for $i in 1 to 1 return
xs:integer($paragraph/@StartNode))
  let $EndParagraph := (for $j in 1 to 1 return
xs:integer($paragraph/@EndNode))
  let $StartSentence := (for $k in 1 to 1 return
xs:integer($sentence/@StartNode))
  let $EndSentence := (for $l in 1 to 1 return
xs:integer($sentence/@EndNode))

  for $i in 1 to 1 return
       if ( (($StartSentence -$StartParagraph) <= 4 ) and
            (($EndParagraph - $EndSentence) <= 2 )    and

(local:CountTitreInInterval($StartParagraph,$StartSentence)>0)
       ) then ($StartSentence - $StartParagraph)
            (: retourner la longueur du NumberTitre qui se trouve
avant la phrase:)
       else (0)
       (: retourner 0 si la phrase n'est pas une phrase de titre:)
};
```

Figure 29 Heuristiques d'identification des titres implémentées avec XQuery

La Figure 29 représente l'implémentation XQuery des heuristiques d'identification des phrases des titres. La fonction « `SentenceIsTitre` » admet comme paramètre deux variables de type nœud « $sentence et $paragraph ». Celles-ci correspondent respectivement à la phrase dont nous cherchons à savoir si elle est une phrase de titre et au paragraphe englobant la phrase. Cette fonction retourne un entier naturel : la longueur du numéro du titre si la phrase est une phrase de titre, sinon « 0 ».

Les quatre premières lignes dans la fonction « `SentenceIsTitre` » permettent de récupérer les débuts et les fins des positions dans le texte de la phrase et du paragraphe. Dans cette fonction, la première condition du test permet de connaître la distance (nombre de caractères) entre le début de la phrase et le début du paragraphe englobant. La distance ne doit pas excéder la longueur maximale des numéros de titres (représentée ici par « 4 »). La deuxième condition permet de vérifier si la fin de la phrase coïncide avec la fin du paragraphe englobant (avec plus ou moins « 2 » caractères). Ces deux premières conditions permettent de vérifier si la phrase est la seule dans le paragraphe. La troisième condition permet de vérifier l'existence d'un numéro de titre candidat entre le début du paragraphe et le début de la phrase. Cette vérification est faite via l'appel de la fonction « `CountTitreInInterval`» qui permet de savoir combien de « `NumberTitre` » (numéros de titres candidats) existent dans un intervalle donné en entrée.

8 Génération des annotations contextuelles

Dans cette partie, nous rappelons que le but est de générer les annotations contextuelles à partir des éléments déjà identifiés dans le texte tels que : les propriétés, les instances de classes, les valeurs candidates des propriétés, les titres, les sous-titres, les partie de phrases, etc.

Cette phase consiste à implémenter l'« Algorithme 4 » de génération des triplets RDF associés aux parties de phrase « ETp » : nous rappelons que « ETp » est l'entité textuelle considérée comme la plus petite.

Nous rappelons que l'idée principale de l'« Algorithme 4 » (Chapitre IV, section 3.2) est d'identifier une propriété « p » pour une « ETp » donnée et sélectionner pour cette propriété son *sujet* et son *objet* associés. Le sujet et l'objet de la propriété « p » sont sélectionnés, d'abord, dans l'«ETp », puis en cas d'échec dans la phrase qui l'englobe, puis continuer à augmenter la portée jusqu'à atteindre l'entité textuelle englobante maximale « ET_max » donnée en entrée de l'algorithme.

L'«Algorithme 4 » est implémenté avec le langage de transformations XSLT. Nous avons choisi XSLT, d'abord parce que les données traitées sont des documents XML, mais ce n'est pas la seule raison. En effet, dans l'« Algorithme 4 » que nous avons proposé, nous avons besoin d'interroger l'ontologie du domaine pour connaître certaines informations:

- Le *domain* et le *range* d'une propriété pour choisir le sujet et l'objet correspondant ;
- La plus spécifique des classes dans l'ontologie (en terme de subsomption), s'il y a plusieurs instances de classe candidates pour une propriété donnée ;

Ces informations doivent être récupérées de l'ontologie pendant le déroulement de l'algorithme. Si cet algorithme avait été implémenté avec le langage de requête XQuery, ces informations n'auraient pas pu être récupérées, car XQuery ne traite pas la subsomption des classes.

En effet, Corese offre une extension permettant d'exécuter une requête SPARQL dans une feuille XSLT. Cette fonctionnalité nous a permis de récupérer les informations de l'ontologie nécessaires au déroulement de l'«Algorithme 4 » (voir Figure 30).

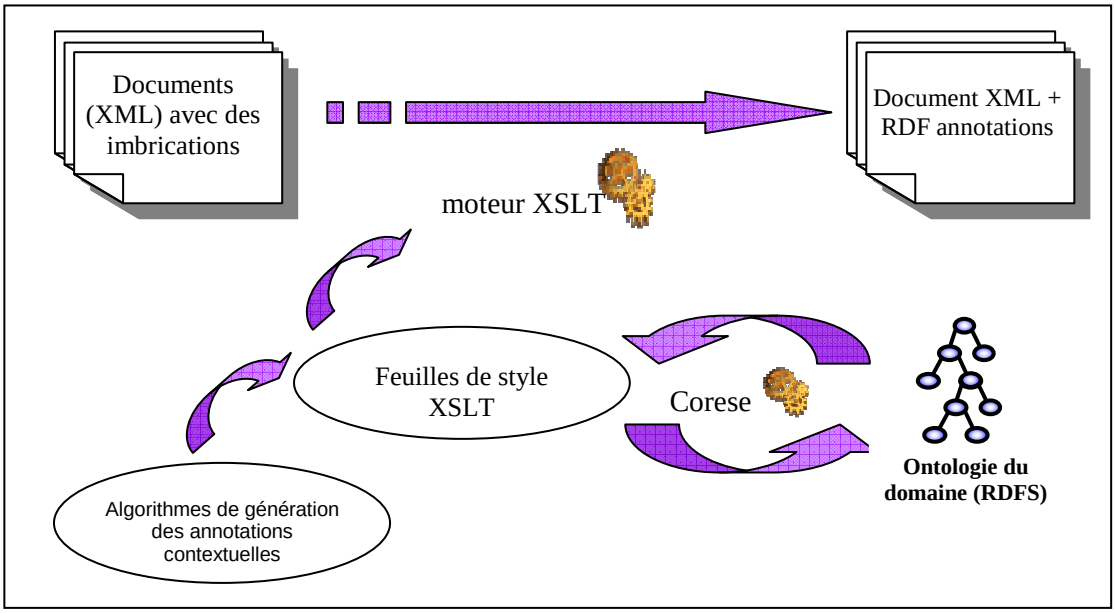


Figure 30 Phase de génération des annotations contextuelles

La Figure 31 montre un exemple d'appel du moteur Corese dans une feuille de style XSLT. La fonction « server :sparql » permet d'exécuter une requête Sparql et le résultat obtenu est sous forme d'un arbre DOM[26]. Ce dernier peut être récupéré dans une variable XSLT ($res) et manipulé ainsi au sein de la feuille de style.

```
<xsl:template name="copyPart">
    <xsl:copy> <xsl:apply-templates select = "@* | node()" />
    </xsl:copy>
</xsl:template>
<xsl:variable name='res'  select='server:sparql($engine,
$sparqlQuery1)'/>
<xsl:variable name="classesVerifyDomainAndDepth">
 <xsl:for-each select='$res//sparql:results/sparql:result'>
  <xsl:if test='sparql:binding[@name = "depth"] =
../sparql:result[1]/sparql:binding[@name = "depth"]'>
   <xsl:variable name="temp" select="sparql:binding[@name
='sub']/sparql:uri">
     <xsl:call-template name="copyPart"/>
   </xsl:variable>
   <xsl:apply-templates select="$temp" />
  </xsl:if>
 </xsl:for-each>
</xsl:variable>
```

Figure 31 Exemple de requête SARQL dans une feuille XSLT

[26] Document Object Model

Ainsi, la requête exécutée dans l'exemple ci-dessus « sparqlQuery1 » permet de récupérer des informations de l'ontologie pendant le déroulement de l'algorithme 4 de génération des annotations. La requête « sparqlQuery1 » est représentée dans la Figure 32.

```
                  ----------- sparqlQuery1 --------------
PREFIX estanda:<http://www.inria.fr/2008/03/24/estanda.rdfs>
SELECT ?sub depth(?sub) as ?depth
WHERE {
{estanda:iProperty rdfs:domain ?sub}
UNION
{estanda:iProperty rdfs:domain  ?c
  ?sub      rdfs:subClassOf ?c}

FILTER (?sub=IdentifiedClasses)
}
ORDER BY DESC (?depth)
```

Figure 32 Requête-template retournant des classes et leur profondeur dans l'ontologie

Pendant l'exécution de l'algorithme de génération des triplets RDF, la requête-template de la figure ci-dessus est utilisée pour générer une requête SPARQL adéquate à une propriété donnée. Dans la requête-template, les chaînes de caractères mentionnées en gras sont remplacées par leur valeur appropriée :

- La chaîne de caractère « iProperty » est remplacée par le nom de la propriété dont nous voulons construire le triplet RDF ;
- La chaîne de caractère « IdentifiedClasses » est remplacée par les classes candidates à être utilisées pour la construction du triplet associé à « iProperty ». Pour chaque classe candidate, la chaîne de caractère « ?sub=IdentifiedClasses » est dupliquée dans la requête et les tests sont composés avec un « ou » booléen.

La requête générée de la requête-Template de la Figure 32 a pour but de retourner les classes (et sous classes) vérifiant la contrainte "domain" de la propriété "iProperty" parmi les classes des instances identifiées (représentées par *IdentifiedClasses* dans FILTER).

Le résultat de la requête « sparqlQuery1 », est récupéré dans la variable « $res » (Figure 31). La suite du code XSLT permet de sélectionner dans la variable « *classesVerifyDomainAndDepth* », les classes ayant la profondeur « $depth » maximale dans l'ontologie.

De la même manière les classes vérifiant la contrainte « range » sont sélectionnées pour un triplet donné.

La question soulevée ici est : dans le cas de plusieurs contraintes « domain » et/ou « range », Est-ce que cela reste valide ?

Nous illustrons la réponse dans l'exemple qui suit :

Etant donnés :

- Le schéma ontologique de la Figure 33, où c1,…, c7 sont des classes ;
- p une propriété, où p.domain =c2 ;
- c2, c3, c5, c6, sont des classes des instances identifiées comme candidates pour un triplet associé à « p » ;

Nous pouvons constater que les classes sélectionnées par la requête « sparqlQuery1 » sont c2 et c6. La classes c6 est celle choisie comme sujet du triplet de la propriété « p », puisque sa profondeur dans l'ontologie est maximale « 2 ».

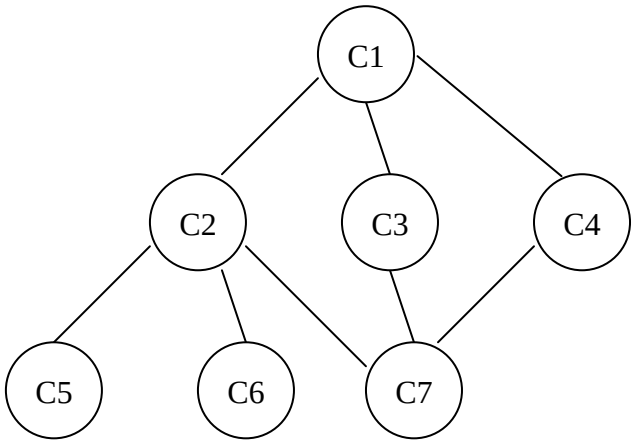


Figure 33 Exemple d'un schéma ontologique

Avec les mêmes données initiales, nous considérons le cas de plusieurs contraintes « domain » :

p.domain = c2 et p.domain =c3

Le « domain » de la propriété « p » devant être considéré est l'intersection des deux « domain ». Par conséquent, la seule classe vérifiant cette contrainte est la classe c7 qui n'est pas candidate comme sujet pour « p ». Par conséquent, deux cas s'offrent à nous :

- Choisir de générer le triplet RDF suivant la sélection proposée par l'exécution de la requête « sparqlQuery1 » (c6 comme sujet). c6 étant une sous-classe de c2, la première contrainte du « domain » est vérifiée. La deuxième contrainte est réalisée par inférence de type (e.g. le système infère que l'instance c6 est de type c3) ;
- Choisir de générer un triplet que si la classe candidate au sujet (c6) vérifie le domain de la propriété. Dans ce cas, puisque c6 ne vérifie pas la contrainte domain « c6 est de type c2 mais pas de c3 », le triplet n'est pas généré.

Nous avons opté pour le deuxième choix, à savoir, gérer le cas de plusieurs « domain ».

Pour résoudre ce problème, nous proposons de tester le nombre de contraintes « domain » de la propriété. Dans le cas de plusieurs domain choisir la requête appropriée. Par exemple, pour une propriété avec deux contraintes « domain », appliquer la requête de la Figure 34 « sparqlQuery2 ».

```
               ----------- sparqlQuery1 --------------
PREFIX estanda: <http://www.inria.fr/2008/03/24/estanda.rdfs#>
SELECT ?sub depth(?sub) as ?depth
WHERE {
{estanda:iProperty rdfs:domain ?c1
 estanda:iProperty rdfs:domain ?c2
     ?sub     rdfs:subClassOf ?c1
     ?sub     rdfs:subClassOf ?c2
}

FILTER (?c1 != ?c2; ?sub=IdentifiedClasses)
}
ORDER BY DESC (?depth)
```

Figure 34 Requête-type pour une propriété avec deux contraintes « domain »

La requête « sparqlQuery2 » retourne les classes « ?sub » qui sont des sous-classes de toutes les classes (dans ce cas deux classes) vérifiant les contraintes « domain ».

Une classe « c » vérifie les deux contraintes « domain » d'une propriété « p » telles que « p.domain = c1 et p.domain=c2 » :

- si « c <c1 et c< c2 », « c » est une sous classe de c1 et de c2 ;
- ou si « c1<c2 », alors « c =c1 », le cas où c1 est une sous classe de c2 alors « c » est la sous-classe c1;

Les deux cas ci-dessus sont gérés par la requête « sparqlQuery2 ».

La requête « sparqlQuery2 » peut être construite automatiquement selon le nombre de contraintes « domain ». Cependant, à ce stade nous n'avons pas mis en œuvre cette construction automatique. Par conséquent, le système CEEMA-T permet de gérer actuellement, au plus, deux contraintes domaine pour une propriété donnée.

Nous rappelons que ce que nous avons détaillé ci-dessus pour la contrainte « domain » est valable aussi pour la contrainte « range » pour la sélection des objets pour la construction des triplets RDF.

9 Conclusion

Dans ce chapitre, nous avons abordé l'implémentation de l'approche d'extraction d'informations et de génération d'annotations sémantiques contextuelles « CEEMA ». Les étapes de l'approche CEEMA sont illustrées en déroulant un exemple de texte. Nous commençons par aborder l'architecture générale de notre système CEEMA-T. Nous avons, d'une part, exploité les outils de TALN « GATE, JAPE, TreeTagger », et d'autre part, les technologies Web et Web sémantique « moteur sémantique Corese, moteur de recherche XQuery Qizx/open et les transformations XSLT ». Par ailleurs, nous exposons les différents choix techniques adoptés ainsi que les problèmes rencontrés et les solutions proposées.

Nous avons détaillé aussi, les différentes étapes de la chaîne de traitement implémentée avec la plate-forme GATE. Cette chaîne est constituée de trois étapes principales :

- *Prétraitement* : cette étape initiale permet de convertir les différents formats texte (pdf, ps, etc.) au format « txt » en conservant la structure physique du texte ;
- *Analyse morpho-syntaxique des textes* : cette étapes consiste à découper du texte (en phrase et en paragraphes), à faire une « tokenisation » (tokens de type nombre, ponctuation, etc.), à faire une lemmatisation ; à affecter à chaque *token* une catégorie d'ordre grammaticale (Verbe, Nom, Adjectif, etc.) ;
- *Détection des propriétés, des instances et des relations contextuelles* : les règles JAPE sont implémentées sous forme de *transducer* dans GATE.

Chapitre VI : Expérimentation de l’approche CEEM

1 Introduction

Ce chapitre présente les expérimentations que nous avons réalisées et qui ont pour objectif de valider notre approche CEEMA. Dans une première partie, nous décrirons les textes sur lesquels l'approche a été expérimentée ainsi que le cadre de leur utilisation. Nous parlerons donc du projet européen SevenPro auquel nous avons participé et situerons nos travaux dans le cadre de ce projet. Nous présenterons par la suite les différents résultats obtenus des expérimentations sur deux plans : en premier sur un corpus de taille réduite, permettant ainsi une validation manuelle et minutieuse des résultats ; ensuite, sur un corpus volumineux en se focalisant sur la validation et l'évaluation des annotations contextuelles générées.

2 Le projet SevenPro

Le projet SevenPro[27] a comme objectif de développer des technologies et des outils basés sur l'ingénierie des connaissances pour l'aide à la conception de produit dans un environnement de réalité virtuelle augmenté 3D.

Le projet européen SevenPro vise à améliorer le processus d'ingénierie de production dans les entreprises de fabrication, au moyen de l'acquisition sémantique (basés sur l'ontologie), la formalisation et l'utilisation des connaissances. À cette fin, des technologies et des outils ont été développés permettant, d'une part, l'annotation et la fouille en profondeur des référentiels d'ingénierie des produits, et d'autre part, une meilleure interaction entre les différentes équipes d'ingénieurs dans le cadre d'un environnement de réalité virtuelle en exploitant l'ingénierie des connaissances.

L'objectif le plus important du projet était un meilleur partage et réutilisation des connaissances du processus de développement (conception et réalisation) des produits. Les améliorations du processus sont mesurées par des critères tels que le temps nécessaire à une innovation de produit, le niveau de réutilisation des connaissances existantes par les entreprises, et la réduction des coûts d'ingénierie.

Dans le cadre du projet SevenPro, deux partenaires de l'ingénierie automobile et de la fabrication en métal coulé collaborent pour identifier leurs besoins communs et étudier des cas de tests réels.

2.1 Vue d'ensemble de l'architecture de SevenPro

L'architecture générale de SevenPro se présente sous forme de modules avec des interfaces clairement définies entre ces modules. Ceci permet d'utiliser les différents modules d'une manière indépendante, ou intégrés dans un environnement commun, dans le but de favoriser le partage et la réutilisation des connaissances.

L'ingénierie des ontologies a été choisie pour être le support de base pour les annotations générées à partir des données existantes, qu'elles soient des documents de bureau, des fichiers

[27] Sevenpro: Semantic Virtual Engineering Environment For Product Design.

de CAO[28] ou des données ERP[29]. Ces annotations sont des informations résumées et utiles contenues dans l'énorme quantité d'information engendrée au cours des activités d'ingénierie.

Les annotations, avec différentes sortes d'informations sémantiques, sont stockées dans une base de connaissances (semantic repository) et accessibles au moyen d'un agent serveur sémantique.

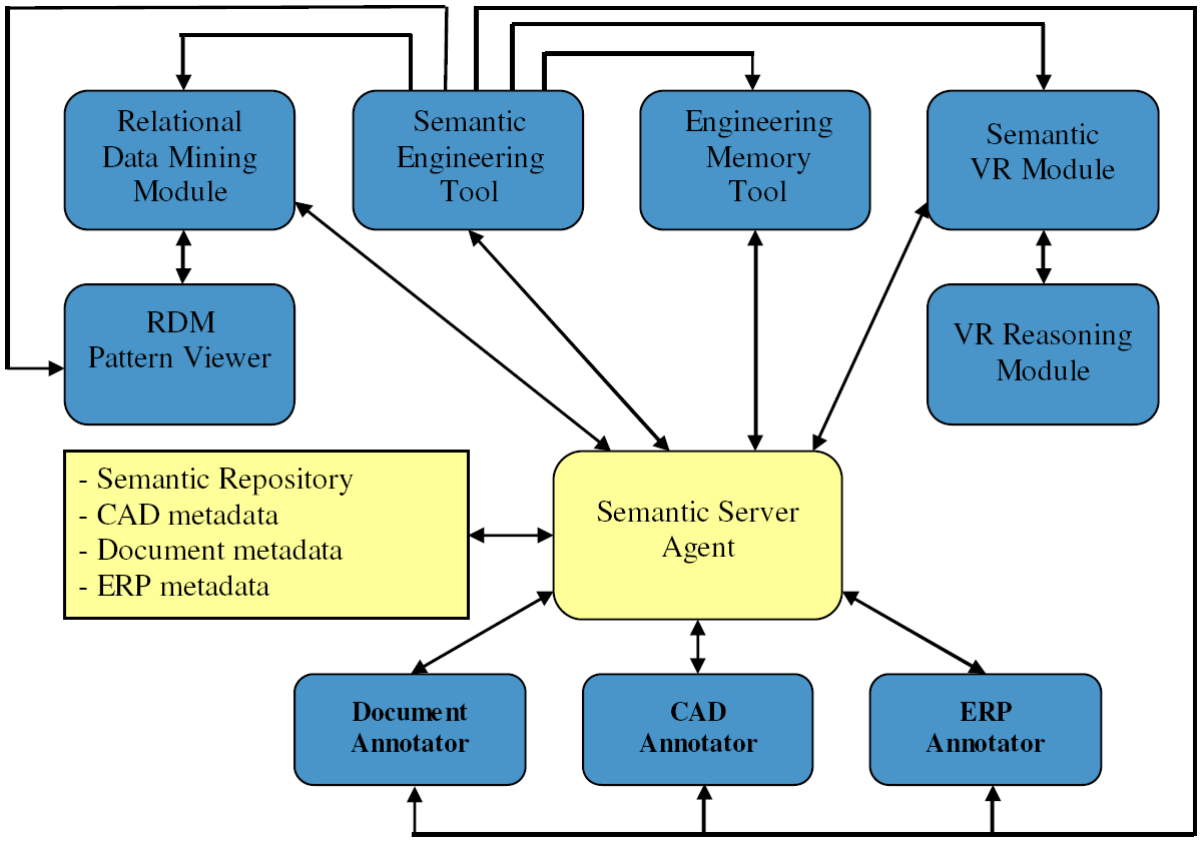


Figure 35 Architecture générale de SevenPro

Différents modules interagissent avec le serveur sémantique, soit pour l'extraction et l'utilisation des informations, soit pour l'enrichir en fournissant de nouvelles connaissances :

- *Un outil d'ingénierie sémantique* (Semantic Engineering Tool) est destiné à faciliter la création de nouveaux produits, la maintenance de la connaissance sur les produits et la recherche d'informations utiles sur les produits à partir de différentes sources de connaissance. Avec la flexibilité que l'utilisation de l'ontologie peut offrir, une entreprise peut modifier et étendre son modèle de produit et voir ces changements se refléter dans l'outil.

[28] Conception assistée par ordinateur.

[29] ERP: Enterprise Resource Planning (Progiciel de gestion intégré).

- *Un module de réalité virtuelle* (Semantic VR Module), renforcé par des technologies sémantiques, permet d'accéder à toutes sortes de connaissance des produits, en plus de ses propres caractéristiques, offrant ainsi un environnement virtuel et interactif pour l'accès et le partage de la connaissance des produits. Ce composant est enrichi d'un module de raisonnement (VR Reasoning Module) qui permet de piloter le comportement des objets selon des règles définies par l'utilisateur sur le modèle du produit.
- *Un outil de recherche sémantique approchée* (Engineering Memory Tool) qui fournit des fonctionnalités de recherche approchée pour récupérer des informations dans des projets similaires, des produits ou des articles déjà présents dans la base de connaissance (semantic repository).
- *Un module de fouille de données relationnelles* (Relational Data Mining Module) dans l'ensemble de la base de connaissances pour extraire des patrons représentant des connaissances de conception. Ces « connaissances sur les connaissances» sont stockées et peuvent être utilisées par les autres modules pour aider les ingénieurs dans leurs différentes tâches.
- *Des modules d'annotation* (Annotator Modules) qui permettent de générer semi-automatiquement des annotations sémantiques à partir des ressources de conception multimédia (CAO, ERP, documents). Les annotations générées sont ajoutées à la base de connaissances. Nos travaux rentrent dans le cadre de ce module d'annotation.

2.2 La collection de texte Estanda

La fonderie ESTANDA « FUNDICIONES DEL ESTANDA », qui est l'un de nos partenaires dans le projet SevenPro, est une compagnie spécialiste dans la fabrication de l'acier à haute résistance depuis 1953. ESTANDA fournit des solutions de qualité aux entreprises dans les secteurs du ciment, automobile, ferroviaire ou les travaux publics.

La fonderie ESTANDA a mis à notre disposition des documents décrivant des pièces de leur production. Nos expérimentations ont porté sur ces documents.

Vu le caractère confidentiel de ces documents, nous avons pris l'initiative de masquer certaines valeurs décrivant la constitution des pièces de production.

2.3 L'ontologie de la fonderie ESTANDA

L'ontologie correspondant à la collection de texte de la fonderie ESTANDA a été développée dans notre équipe de recherche avec la supervision et la validation de nos collaborateurs d'ESTANDA.

L'ontologie est constituée de 65 concepts et 50 propriétés et est écrite en RDFS. La Figure 36 montre un exemple d'une classe « `RingOfDiaphragmMill` » et d'une propriété « `HasPart` » de l'ontologie ESTANDA. Pour des raisons de confidentialité ces exemples sont les seuls que nous pourrons illustrer.

```
<!-- ===========  hasPart  ============== -->
      <rdf:Property rdf:ID="HasPart">
            <rdf:type rdf:resource="&owl;TransitiveProperty"/>
            <owl:inverseOf rdf:resource="&estanda;#partOf"/>

            <domain rdf:resource="#Item"/>
            <range rdf:resource="#Item"/>
            <label xml:lang="en">has Part</label>
            <label xml:lang="en">Is Composed Of</label>

            <comment xml:lang="en">hasPart is transitive and
also reflexive, and anti-symmetrical.</comment>
      </rdf:Property>

<!-- ===========  RingOfDiaphragmMill ============== -->

      <Class rdf:ID="RingOfDiaphragmMill">
            <subClassOf rdf:resource="#Item"/>
            <label xml:lang="en">Ring Of Mill Diaphragm</label>
            <label xml:lang="en">Ring Of Diaphragm</label>
            <label xml:lang="fr">Anneau Diaphragme</label>
            <comment xml:lang="en">denotes a part Of a
diaphragm mill (i.e. ring).</comment>
      </Class>
```

Figure 36 Exemple d'une classe et d'une propriété de l'ontologie ESTANDA

3 Validation et évaluation de la méthodologie

Dans cette partie, nous présentons une étude quantitative et qualitative de la méthodologie de génération des annotations contextuelles, d'une part, pour valider son aspect automatique, et d'autre part, pour juger la qualité des annotations générées.

Dans notre travail, évaluer la qualité des annotations contextuelles générées consiste à évaluer les cinq points suivants :

1. La capacité des grammaires de détection à identifier les instances possibles d'une propriété dans le texte ;
2. La capacité des grammaires de détection à identifier les instances de concepts pouvant être liées par les propriétés;
3. La capacité des grammaires de détection à identifier les relations dites contextuelles (rhétoriques ou autre) ;
4. La capacité de l'« algorithme 1» (décrit dans le chapitre IV, section 2.2) à construire correctement les imbrications hiérarchiques pour un texte donné;
5. La capacité à relier les bonnes instances de concepts par les bonnes propriétés (génération des triplets RDF), tout en garantissant la cohérence avec les contraintes décrites dans l'ontologie du domaine (domain, range) ;

Pour évaluer notre approche nous avons utilisé deux mesures fréquemment utilisées dans l'évaluation des systèmes d'extraction d'informations : la précision (P) et le rappel (R).

$$\text{Précision} = \frac{\text{nombre des bonnes identifications}}{\text{nombre total identifié}}$$

$$\text{Rappel} = \frac{\text{nombre des bonnes identifications}}{\text{nombre total qui aurait dû être identifié}}$$

Nous avons utilisé les définitions de la « précision » et du « rappel » ci-dessus pour évaluer, d'une part, l'identification des titres, des instances de propriétés, des instances de classes et des relations contextuelles, et d'autre part, la génération des triplets RDF.

La précision mesure l'absence de bruit dans l'extraction (le pourcentage des éléments extraits correctement). Le rappel mesure l'absence de silence dans l'extraction (le pourcentage des éléments extraits par rapport aux éléments qui auraient dû être extraits).

Une autre métrique est calculée en combinant les deux mesures, il s'agit de la F-mesure (dite aussi moyenne harmonique) :

$$\text{F-mesure} = \frac{2PR}{P+R}$$

Remarque

Pour le dénominateur du rappel « *nombre total qui aurait dû être identifié* », nous considérons, uniquement les instances (des propriétés et des classes) ayant une description conceptuelle dans l'ontologie, dans le cas de l'évaluation de leur identification.

Notre expérimentation a été élaborée en deux phases : la première phase est faite sur un corpus de taille réduite et la seconde sur un corpus de taille plus grande.

Bien qu'elle soit coûteuse, nous avons opté pour une évaluation manuelle des résultats.

3.1 Phase 1 : corpus de taille réduite

Le texte sur lequel nous avons exécuté cette expérimentation fait partie des collections de documents d'ESTANDA. Le texte est constitué de 313 phrases (3768 mots et 1862 autres unités linguistiques telles que les : chiffres, virgules, parenthèses,…). Un exemple de ce texte est illustré dan la Figure 37. La langue du texte est l'anglais.

Nous nous somme focalisés dans cette phase sur l'évaluation des quatre premiers points (cités plus haut dans la section 3).

```
3 Description of the mill internal elements

3.1 Inlet Headliners


This new design is composed of N thicker bolted rings, compared
to the original design of M rings (larger and bulkier plates
making handling more difficult.). The liners have a thickness of
XXmm. except for the area of most wear (R/2-R/2+R/3), where the
thickness is YYmm.
```

Figure 37 Exemple de partie de texte dans un document ESTANDA

Nous avons exécuté le module de génération de règles de grammaire JAPE et nous avons obtenu : 114 règles de grammaire JAPE générées automatiquement et qui correspondent à, d'une part, 64 concepts et 50 propriétés dans l'ontologie, et d'autre part, 80 autres règles de grammaire JAPE pour identifier les relations contextuelles (entre autres des relations de discours). Pour voir plus d'exemple de règles JAPE, voir l'annexe 1.

Remarque

Certaines informations sur les figures sont masquées pour respecter des clauses de confidentialité. Par exemple, dans la Figure 37, les « XX et YY » remplacent des chiffres qui mentionnent l'épaisseur d'une pièce.

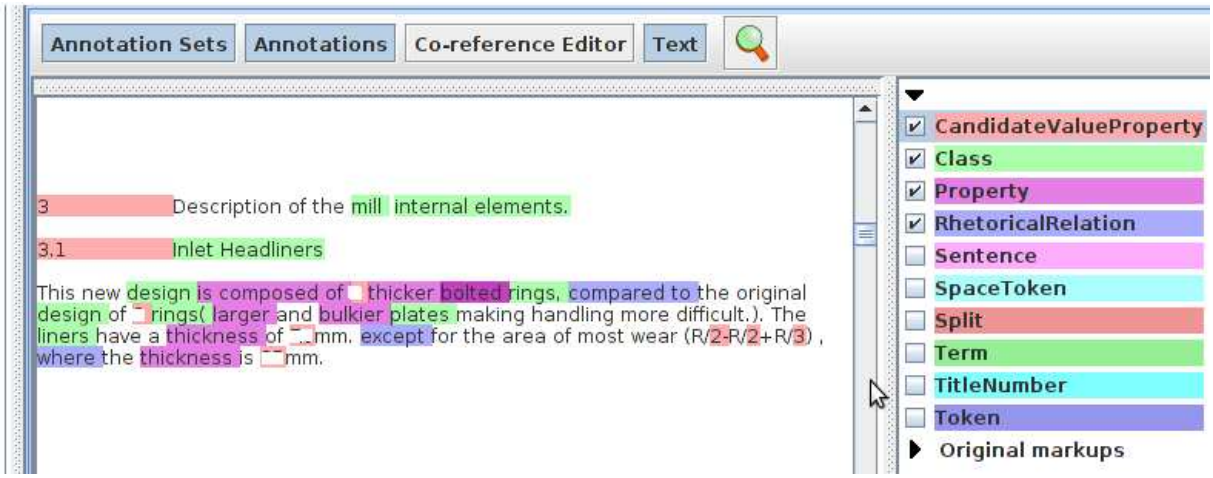


Figure 38 Résultat de l'extraction dans le texte de l'exemple de la Figure 37 (vue dans GATE)

La Figure 38 montre une vue sur GATE de différents éléments identifiés dans le texte : instances de propriétés, instances de classes, des relations contextuelles (entre autres rhétoriques) et les symboles numériques.

Le résultat d'exécution des différents modules de notre approche est listé dans le Tableau 4.

	identification		Total existant	Précision (%)	Rappel (%)	F-mesure (%)
	Bonne	Mauvaise				
Titres et imbrication	26	0	26	100	100	100
Phrases	285	101	313	-	-	-
Paragraphes	205	0	205	-	-	-
Relations contextuelles	113	16	127	87,59	88,87	88,22
Propriétés	174	30	195	85,29	89,23	87,21
Classes	436	126	458	77,58	95,19	85,48
Valeurs candidates pour les propriétés	413	15	413	-	-	-

Tableau 4 Evaluation des résultats d'extraction de la phase 1

Le résultat de la ligne « *Phrases* » du Tableau 4 concerne l'identification des phrases. Nous rappelons que nous utilisons le « transducer » par défaut dans GATE pour identifier les phrases d'un texte. Nous avons jugé inutile de calculer la « précision » et le « rappel » pour les phrases vu que nous n'avons pas développé le « transducer » utilisé, ceci pour ne pas créer une confusion entre ce que nous avons développé et ce qui a été réutilisé.

Le résultat de la ligne « Paragraphes » du tableau concerne l'identification des paragraphes dans le texte. De même que pour les phrases, le « transducer » utilisé est celui offert par défaut dans GATE. Nous n'avons pas calculé la « précision » et le « rappel » pour les mêmes raisons citées dans le cas de l'identification des phrases.

Le résultat illustré par la ligne « *Valeurs candidates pour les propriétés* » du tableau concerne toute les valeurs numérique pouvant exister dans le texte. En effet, ceci est le résultat de l'exécution de la règle de grammaire JAPE cité dans « chapitre IV, section 3.3 ». Nous nous assurons pendant la génération des triplets RDF que ces identifications ne doivent pas vérifier les heuristiques « 1, 2 et 3 » pour ne pas les confondre avec les valeurs numériques précédant les titres.

Nous détaillerons dans ce qui suit le résultat du tableau ci-dessus.

3.1.1 Evaluation de l'identification des instances de propriétés

En appliquant les règles de grammaire JAPE correspondant aux identifications des instances de propriétés, nous avons obtenu un assez bon score: 85,29 % de précision et 89,23 % de rappel.

Nous listons dans le Tableau 5 les instances de propriétés que nous avons pu identifier pour l'exemple de texte de la Figure 37. Elles sont triées selon leur ordre d'apparition dans le texte :

Texte indicateur (label) d'instance de propriétés dans le texte	***Noms des propriétés associées dans l'ontologie***
is composed of	*HasPart*
larger	*Larger*
bulkier	*bulkier*
thickness	*thickness*
bolted	*RingTypeBolt*
	BoltLinerMill
thickness	*thickness*

Tableau 5 Instances de propriétés identifiées et « labels » ayant permis leur identification

Nous soulignons que nous avons pu identifier ici sept propriétés alors qu'il ne devait y en avoir que six. Nous expliquons ceci par l'existence de « Bolt » comme valeur de *rdfs:label* pour deux relations différentes. En effet, le *rdfs:label* de la propriété *RingTypeBolt* (anneau boulonné) est « Bolted » et le *rdfs:label* de la propriété *BoltLinerMill* est « Bolt », et puisque nous lemmatisons les mots des *rdfs*:*label* avant de les comparer avec le texte, nous avons le même texte à comparer pour deux propriétés différentes (pour plus de détail voir chapitre IV, section 3.2).

Pour résoudre cette ambigüité nous proposons d'appliquer dans l'ordre ce qui suit :

- Dans le cas où plusieurs instances de propriétés identifiées partagent une portion de texte, choisir celle qui a la plus longue chaîne de caractères. Solution détaillée dans « Chapitre IV, section 3.1, heuristique 4 » ;
- Appliquer la politique de l'« autruche » qui consiste à ne rien faire de particulier: laisser tourner l'algorithme permettant de générer les triplets RDF avec les deux propositions d'instances de propriétés. Puis, conserver le bon triplet (voire ci-dessous) ;
- Si aucune des deux propositions ne résolvent le problème, faire un retour sur expérience aux experts du domaine responsables de la construction de l'ontologie pour affiner les descriptions des *rdfs:label* dans l'ontologie.

En appliquant la politique de l'« autruche», l'algorithme de génération des triplets RDF (Chapitre IV, section 3.4 : Algorithme 4) génère un triplet pour chaque instance des deux propriétés *RingTypeBolt* et *BoltLinerMill*. Nous rappelons que nous prenons en compte les contraintes (*domain* et *range*) dans cet algorithme. En sachant que les deux propriétés ont respectivement comme « domain » les classes « RingOfDiaphragmMill » et « LinerMill », nous avons :

- Le triplet généré de la propriété *RingTypeBolt* est :

```
{:RingOfDiaphragmMill  :RingTypeBolt  'bolted' }
```

Ce dernier est généré en utilisant les arguments se trouvant dans l'entité textuelle la plus petite (ETp) de la propriété ;

- Le triplet généré de la propriété *BoltLinerMill* est :

  `{:Mill :BoltLinerMill 'bolted'}`

  Ce dernier est généré en utilisant les arguments se trouvant dans l'entité textuelle de la propriété ainsi que la phrase du titre (voir le texte de l'exemple).

Nous constatons que le sens du texte va à l'encontre du second triplet qui est erroné. Par conséquent, nous pouvons généraliser ce cas et ajouter une heuristique pour choisir le bon triplet à conserver s'il y a une ambiguïté entre deux propriétés associées à un même *token* (ou une même séquence de token).

Heuristique 13 : si plusieurs propriétés candidates sont identifiées pour un token donné (ou une même séquence de *token*), ne considérer que celle dont le triplet généré est construit en utilisant l'entité textuelle minimale.

Remarque

Nous ne traitons pas le cas où l'ambiguïté persiste après l'application de l' « heuristique 13 » C'est-à-dire, dans le cas où les triplets associés aux propriétés sont générés en utilisant une même entité textuelle minimale. Nous considérons que ce cas est très peu probable.

La Figure 39 montre la liste des propriétés identifiées pour l'exemple de texte de la Figure 37 avec :

Start : le début de la position de la propriété dans le texte (premier caractère de la propriété);

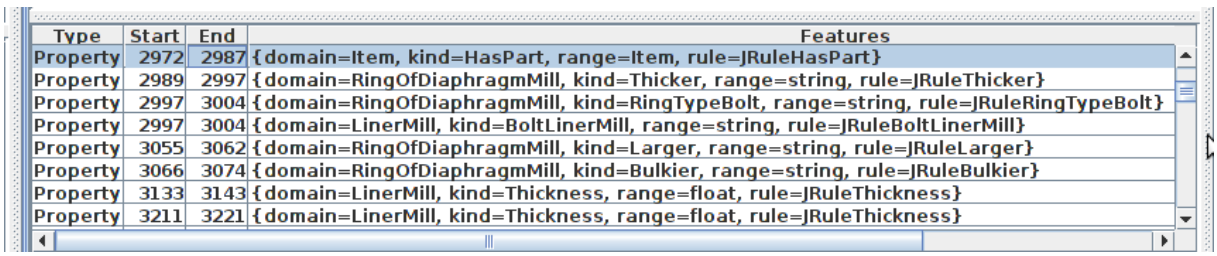
End : la fin de la position de la propriété dans le texte ;

domain : la propriété « rdfs:domain » récupérée de l'ontologie ;

range : la propriété « rdfs:range » récupérée de l'ontologie ;

kind : le nom de la propriété dans l'ontologie ;

rule : le nom de la règle JAPE qui a permis de l'identifier.



Type	Start	End	Features
Property	2972	2987	{domain=Item, kind=HasPart, range=Item, rule=JRuleHasPart}
Property	2989	2997	{domain=RingOfDiaphragmMill, kind=Thicker, range=string, rule=JRuleThicker}
Property	2997	3004	{domain=RingOfDiaphragmMill, kind=RingTypeBolt, range=string, rule=JRuleRingTypeBolt}
Property	2997	3004	{domain=LinerMill, kind=BoltLinerMill, range=string, rule=JRuleBoltLinerMill}
Property	3055	3062	{domain=RingOfDiaphragmMill, kind=Larger, range=string, rule=JRuleLarger}
Property	3066	3074	{domain=RingOfDiaphragmMill, kind=Bulkier, range=string, rule=JRuleBulkier}
Property	3133	3143	{domain=LinerMill, kind=Thickness, range=float, rule=JRuleThickness}
Property	3211	3221	{domain=LinerMill, kind=Thickness, range=float, rule=JRuleThickness}

Figure 39 Liste des propriétés identifiées avec leurs caractéristiques détaillées (vue dans GATE)

3.1.2 Evaluation de l'identification des instances de classes

En appliquant les règles de grammaire JAPE correspondant aux identifications des instances de classes, nous avons obtenu le score : 77,58% de précision et 95,19% de rappel.

Nous listons dans le Tableau 6 les instances de classes que nous avons pu identifier pour l'exemple de texte de la Figure 37. Elles sont triées selon leur ordre d'apparition dans le texte :

Texte indicateur (label) d'instance de classes dans le texte	***Noms de leurs classes associées dans l'ontologie***
mill	*Mill*
internal elements	*InternalElements*
Inlet Headliners	*InletHeadliner*
design	*Design*
rings	*RingOfDiaphragmMill*
design	*Design*
rings	*RingOfDiaphragmMill*
plates	*PlateOf DiaphragmMill*
liners	*LinerMill*

Tableau 6 Instances de classes identifiées et « labels » ayant permis leur identification

Dans l'exemple ci-dessus, toutes les instances de classes sont identifiées avec succès. Cependant, ce n'est pas toujours le cas. En effet, principalement les mauvaises identifications des classes sont dues à l'identification de plusieurs instances de classes différentes alors qu'il ne devait y en avoir qu'une seule. Ceci explique le grand écart entre la précision et le rappel (respectivement 77,58% et 95,19%).

A titre d'exemple, pour une partie de texte « Outlet diaphragm », nous avons identifié les instances de classes : « OutletDiphragmMill » et « DiaphragmMill ». Nous expliquons ceci par l'existence du texte *diaphragm* (ou son lemme) dans le *rdfs:label* des deux classes.

Nous soulignons dans cet exemple que « OutletDiphragmMill » est une sous classe de « DiaphragmMill ».

Par ailleurs, ce manque de précision n'a pas d'impact sur la génération des triplets. En effet, les classes des instances identifiées sur une même séquence de tokens ont généralement des relations de subsomption entre elles (classes/sous-classes). Par conséquent, l'instance de classe sélectionnée pour une propriété donnée est celle dont la classe est la plus spécifique dans la hiérarchie de l'ontologie (voir Chapitre IV, section 3.4, heuristique 11).

3.1.3 Evaluation de l'identification des titres et de leur imbrications

Le résultat de la ligne « *Titres et imbrication* » du Tableau 4 concerne l'identification des titres dans un document ainsi que la constitution des imbrications entre titres, sous titres, paragraphes et phrases.

L'algorithme de construction des imbrications entre titres (Chapitre IV, section 2.2, algorithme 1) fonctionne sans erreurs. La vérification manuelle des identifications des titres à été faite sur 14 documents différents. Le résultat de l'un d'eux est affiché dans le Tableau 4 (ligne *titres et imbrication*). Tous les titres du document ont été identifiés avec succès (26 titres) et les imbrications sont constituées sans erreur.

Par ailleurs, la construction des imbrications entres titres est très sensible aux erreurs d'identification. En effet, tout titre identifié par erreur pourra élargir ou réduire la portée d'un autre titre. Si nous n'avons eu aucune erreur d'identification de titre, c'est parce que tous les titres sont précédés par leurs indicateurs numériques.

La Figure 40 montre un exemple de reconstitution des imbrications entres titres, paragraphes, phrases et entités textuelles. Dans cette figure, seule la première phrase de l'exemple de la Figure 37 est détaillée.

```
<titre Id="37" StartNode="2887" EndNode="2932">
   Description of the mill  internal elements.
 <titre Id="38" StartNode="2934" EndNode="2954">
   Inlet Headliners
   <paragraphe Id="39" StartNode="2956" EndNode="3229">
    <sentence Id="15139" StartNode="2956" EndNode="3112">
      <entiteTextuelleETp>
        This new design is composed of N thicker bolted rings,
      </entiteTextuelleETp>
      <contextualRelation Markers="comparedto">
       <entiteTextuelleETp>
         the original design of M rings( larger and bulkier plates
         making handling more difficult.).
       </entiteTextuelleETp>
      </contextualRelation>
    </sentence>
    ...
   </paragraphe>
  </titre>
 </titre>
```

Figure 40 Exemple d'imbrication de la première phrase de l'exemple de la Figure 37

3.1.4 Evaluation de l'identification des relations contextuelles

En appliquant les règles de grammaire JAPE correspondant aux identifications des relations contextuelles, nous avons obtenu le score « 87,59% de précision et 88,87% de rappel » affiché dans la ligne « *Relations contextuelles* » du Tableau 4.

Pour le texte de l'exemple de la Figure 37, nous avons pu identifier toutes les relations contextuelles à savoir : *compared to, except, where*.

Les relations contextuelles considérées dans cette expérimentation sont principalement des marqueurs de discours. En plus de ces marqueurs de discours, nous avons considéré (c'est-à-dire, ajouté dans la liste des relations contextuelles à identifier) d'autres relations telles que la relation « *compared to* ».

Lors de l'identification des relations contextuelles, nous avons rencontré le même problème récurrent, à savoir, l'identification de plusieurs relations contextuelles dans une séquence de token alors qu'il ne devait y avoir qu'une seule.

Nous avons apporté des solutions à ce problème que nous avons détaillé dans « Chapitre IV, section 3.1 ». La solution est de ne garder que la relation ayant la plus longue chaîne de caractères.

La difficulté pour les relations contextuelles réside dans l'identification de leurs arguments. Identifier ces arguments revient à définir quelles sont les parties de texte concernées par une relation contextuelle.

Pour l'exemple de la Figure 37, les arguments concernés par la relation « *compared to* » sont indiqués entre crochets dans ce qui suit :

[This new design is composed of N thicker bolted rings,] **compared to** [the original design of M rings(larger and bulkier plates making handling more difficult.)]

Ceci est représenté par la structure arborescente illustré dans la « Figure 40 ».

Si plusieurs relations contextuelles sont identifiées dans une même phrase, par exemple une phrase du type « text1 *RC1* text2 *RC2* text3 *RC3* text4 *RC4* text5» alors nous nous limitons à constituer la portée sémantique de ses relations comme suit :

[text1 *RC1* [text2 *RC2* [text3 *RC3* [text4 *RC4* text5]]]

où les RCi sont des relations contextuelles

Pour plus de détails, voir le Chapitre IV, section 2.2, Algorithme 1, fonction *buildPartsOfPhrase.*

Nous soulignons que nous ne traitons pas certains cas complexes de marqueurs de discours dont le découpage de leur portée prend d'autres formes. Ceci requiert une étude approfondie de chacun des marqueurs de discours.

En plus des marqueurs de discours comme relations contextuelles, la reconstitution des imbrications (structure physique des documents) nous permet de déduire les relations contextuelles telles que : *est imbriqué dans, est le successeur de, est le prédécesseur de.*

3.2 Phase 2 : corpus de taille plus large

Le corpus de texte sur lequel nous avons fait notre deuxième expérimentation fait partie des collections de documents de la fonderie ESTANDA. Ce corpus est sept fois et demie plus important que celui de la première phase d'expérimentation et est constitué de 2422 phrases.

Nous nous somme focalisés dans cette expérimentation au cinquième point de l'évaluation (cité dans la section 3), à savoir, l'évaluation de la génération des triplets RDF. La Figure 41 montre un exemple d'une annotation contextuelle générée sur la première phrase de l'exemple de la Figure 37.

Nous avons choisi de représenter le résultat obtenu de la génération des annotations contextuelles avec le formalisme des graphes nommés avec les contraintes suivantes:

- Regrouper les triplets RDF identifiés dans une entité textuelle (ETp) dans un même graphe nommé ;
- Relier les graphes nommés entre eux par les relations contextuelles ;
- Garder les imbrications de la structure physique entre les graphes nommés ;

```
:G11 { :Design_16175 rdf:type :Design .
       :Design_16175 :HasPart :RingOfDiaphragmMill_16001 .

       :RingOfDiaphragmMill_16001 rdf:type :RingOfDiaphragmMill .
       :RingOfDiaphragmMill_16001 :Thick "thicker".

       :Mill_16048 rdf:type :Mill .
       :Mill_16048 :BoltLinerMill "bolted" }

:G12 { :RingOfDiaphragmMill_16002 rdf:type :RingOfDiaphragmMill .
       :RingOfDiaphragmMill_16002 :Larger "larger"

       :RingOfDiaphragmMill_16003 rdf:type :RingOfDiaphragmMill .
       :RingOfDiaphragmMill :Bulkier "Bulkier" }

:G1  { :G11 rc:comparedto :G12 }

:G21 { :LinerMill_15140 rdf:type :LinerMill .
       :LinerMill_15140 :Thickness "XX"^^xsd:decimal }

:G22 { :LinerMill_15999 rdf:type :LinerMill .
       :LinerMill_15999 :Thickness "YY"^^xsd:decimal  }

:G2  { :G21 rc:except _:w1 .
      _:w1 rc:where :G22 }
```

Figure 41 Annotations contextuelles générées pour l'exemple de texte de la Figure 37 et représentées par des graphes nommés

Nous nous sommes contentés de représenter les relations contextuelles de type « marqueurs de discours ». En effet, d'autres relations contextuelles entre les graphes nommés peuvent être déduites de la structure physique du texte. Par exemple :

```
:G12 rc:estLeSuccesseurDe  :G11
:G11 rc:estLePredecesseurDe :G12
:G2  rc:estLeSuccesseurDe   :G1
:G1  rc:estImbriqueDans     :G0        (G0 étant le graph nommé englobant)
:G2  rc:estImbriqueDans     :G0
```

Le *préfixe* « rc » correspond à l'ontologie des relations contextuelles en cours de construction (l'ontologie n'est pas achevée) « voir Annexe 2 ». Le but était de proposer un support ontologique pour l'exploitation des relations contextuelle dans des moteurs sémantiques existants, tels que Corese. Cependant, des propositions d'ontologies plus exhaustives sur les relations rhétoriques sont disponibles, telles que l'ontologie SALT[30] [Groza, T., et al. 2007], et nous comptons les réutiliser dans nos travaux futurs.

La représentation des annotations contextuelles avec les graphes nommés dépend étroitement de la bonne reconstitution de la structure physique des documents textuels. Vu que le score obtenu pour la construction des imbrications est très satisfaisant, la génération des graphes nommée est de bonne qualitée.

Il nous reste à discuter des résultats obtenus pour la génération des triplets RDF. Nous rappelons que le principe de l'« Algorithme 4 » (Chapitre IV, section 3.4) est de générer des annotations contextuelles et d'identifier le « sujet » et l'« objet » associés à une propriété donnée dans le texte pour constituer un triplet. Les sujets et les objets candidats pour la propriété sont sélectionnés parmi les sujets et les objets identifiés dans l'élément textuel où se trouve la propriété. Si les sujets et/ou les objets candidats ne correspondent pas à la propriété, la sélection des sujets et des objets se fait à un niveau plus englobant : la phrase, le paragraphe, le titre, etc.

Dans cette expérimentation, nous avons non seulement évalué l'algorithme de génération des annotations contextuelles, mais aussi identifié le niveau (phrase, paragraphe, etc.) permettant d'obtenir le meilleur rapport entre précision et rappel. Ceci revient à définir le meilleur « ET_max » dans l' « Algorithme 4 ». Les résultats de cette expérimentation sont regroupés dans le Tableau 7.

[30] http://salt.semanticauthoring.org/ontologies/sro

Génération des triplets RDF en utilisant comme resources textuelles (ET_max) :	Identifications		Total existant	Précision (%)	Rappel (%)	F-mesure (%)
	Bonne	Mauvaise				
(1) : ETp sans phrases des titres	191	29	756	86,82	25,26	39,13
(2) : (1) + phrases des titres	539	113	756	82,67	71,30	76,56
(3) : (2) + phrases	546	118	756	82,23	72,22	76,90
(4) : (3) + Paragraphe	593	130	756	82,02	78,44	80,19
(5) : (4) + Section	641	208	756	75,50	84,79	79,87

Tableau 7 Evalution de l'algorithme de génération des annotations contextuelles

La première exécution de l'algorithme a été faite en limitant l'entité textuelle maximale (ET_max) à la partie de phrase (entité textuelle la plus petite « ETp »). Nous constatons que le résultat de la première exécution (ligne « (1) : ETp sans phrases des titres » du tableau) n'est pas satisfaisant (rappel à 25,26%).

La deuxième ligne du tableau « (2) : ETp + phrases des titres » illustre l'importance des « phrases des titres » et l'impact qu'elles peuvent avoir sur le résultat de l'expérimentation. En effet, nous avons constaté que les phrases des titres peuvent comporter des instances de classes pouvant être le sujet ou l'objet d'une propriété identifiée dans le texte qui est dans la portée du titre. Ceci nous a poussé à incorporer les sujets et les objets (instances de classes) identifiés dans la phrase d'un titre, dans toute les sélections des sujets et des objets candidats pour les propriétés existantes dans la portée du titre. Ceci est valable pour tous les titres englobants l'ETp de la propriété. En effet, il peut y avoir plusieurs titres pour un ETp.

La ligne « (3) : (2) + phrases » du tableau correspond au résultat pour un ET_max limité à la phrase où se trouve la propriété concernée avec la prise en compte des instances de classes existantes dans les phrases des titres englobant. Nous avons remarqué une amélioration peu significative au niveau du rappel « 72,22 % » et une régression minime dans la précision « 82,23 » en comparaison avec la deuxième expérimentation. Nous pouvons constater que les phrases des titres, ainsi exploitées, jouent un rôle important dans l'amélioration des résultats.

La quatrième expérimentation est de porter la limitation de « ET_max » au paragraphe qui englobe la propriété concernée en plus des titres englobants. Le résultat de cette expérimentation est présenté dans la ligne « (4) : (3) + Paragraphe » du tableau et est le meilleur rapport obtenu entre précision et rappel dans la génération des triplets RDF.

La cinquième expérimentation consiste à augmenter le « ET_max » à la section. Cette portée assez large a fait augmenter considérablement le rappel « 84,79 %» mais au détriment de la précision « 75,50 % » (voir la ligne « (5) : (4) + Section» du tableau).

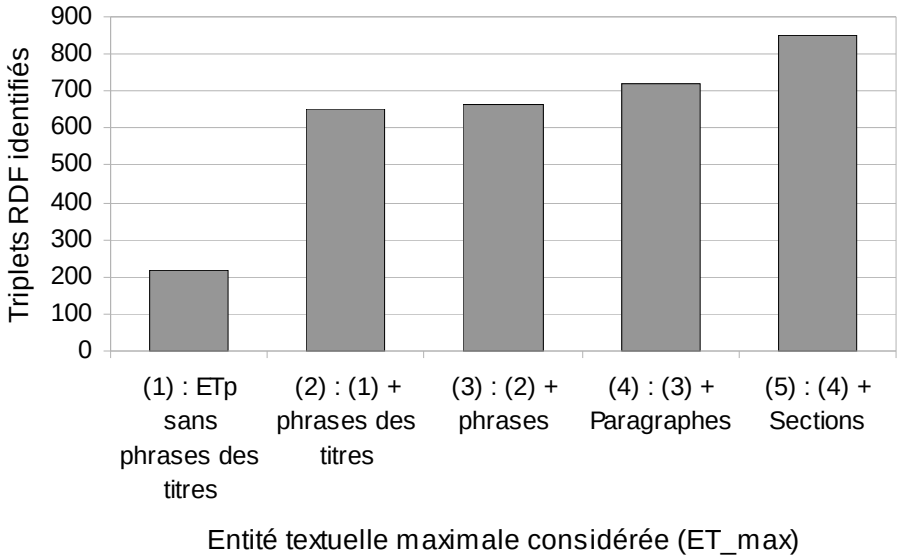


Figure 42 Impact de ET_max sur la génération des triplets RDF

La Figure 42 montre l'impact de l'entité textuelle maximale (ET_max) dans la génération de triplets RDF. Dans cette figure nous pouvons constater d'une manière claire le rôle important que joue l'exploitation des phrases des titres. En effet, le nombre des triplets générés avec la prise en compte des phrases des titres est presque le triple du nombre des triplets générés sans prise en compte des phrases des titres (premier et second histogramme).

Nous rappelons que le meilleur score obtenu entre précision et rappel était dans la quatrième expérimentation où le « ET_max » est limité au paragraphe (F-mesure 80,19 %)

Par ailleurs, nous soulignons d'un autre côté que notre approche dépend étroitement de la qualité descriptive de l'ontologie du domaine utilisée. En effet, nous avons rencontré dans nos expérimentations certains documents où très peu d'annotations sont générées. Nous avons constaté que ceci arrive lorsque la description de l'ontologie du domaine ne correspond pas aux textes. Après avoir fait un retour d'expérience à nos partenaires industriels sur les documents qui n'ont pas été annotés, nous avons eu la confirmation que ces documents concernaient la méthodologie d'assemblage des produits et non une description des produits. Par ailleurs, la dépendance de notre approche vis à vis de la qualité de l'ontologie et son adéquation avec les textes pourra offrir une information supplémentaire. En effet, le nombre d'annotations (triplets RDF) pourra être perçu comme un indicateur qui permet d'identifier les documents correspondant (en adéquation) à une ontologie donnée.

4 Conclusion

Nous avons présenté nos travaux qui s'inscrivent dans le cadre du projet européen SevenPro dans lequel nos partenaires industriels ont mis à notre disposition des données réelles afin de valider et d'évaluer notre approche. Les objectifs tracés dans un milieu industriel nous ont permis de mettre à l'épreuve notre approche CEEM et de pousser le développement de l'outil CEEMA-T à un niveau d'automatisation avancé. En effet, nous rappelons que notre approche admet en entrée une ontologie du domaine et des textes bruts et donne en sortie des annotations sémantiques contextuelles.

Nous avons pu constater à travers l'expérimentation que les annotations générées sont de qualité satisfaisante bien que l'approche soit entièrement automatisée.

Dans nos expérimentations, nous nous sommes restreints aux documents de notre partenaire « la fonderie ESTANDA ». Ceci est dû au manque de corpus textuels disponibles ayant une ontologie du domaine de bonne qualité. Notre approche requiert des ontologies avec une forte composante « terminologique ». En effet, nous avons essayé de faire d'autres expérimentations sur d'autres corpus avec leurs ontologies, mais le problème qui revenait à chaque fois était la qualité de l'ontologie et son adéquation avec le texte. Souvent les « rdfs:label » sont inexistants dans les ontologies ou ne reflètent pas la présence, par exemple, des instances de concepts dans le texte.

Conclusions et perspectives

Contributions scientifiques

Le besoin pour lequel ce travail a été initié est « d'offrir une plateforme d'annotation sémantique pour nos partenaires industriels pour capitaliser leurs connaissances issues des corpus textuels ».

Dans un cadre industriel et particulièrement dans le cas de nos partenaires, le besoin de qualité et de précision d'information s'impose. Les annotations sommaires ne sont pas pertinentes et sont inadaptées aux besoins réels.

Afin de traiter cette problématique, nous nous somme retournés vers l'annotation sémantique avec ce quelle peut offrir. Nous rappelons nos hypothèses initiales qui peuvent être résumées dans ce qui suit :

- **La modélisation des connaissances du domaine** : les connaissances du domaine peuvent être capitalisées par les annotations sémantiques et les ontologies ;
- **L'annotation sémantique pour la recherche d'informations :** l'annotation sémantique de documents apporte un plus à la recherche d'information ;
- **Les technologies du Web sémantique au service de l'annotation sémantique :** l'annotation sémantique peut s'appuyer sur les technologies du Web sémantique et ainsi tirer profil d'un support puissant de représentation et de raisonnement sémantique ;
- **La modélisation du contexte pour le texte et les annotations sémantiques :** l'absence de consensus sur la définition du « contexte » nous amène à faire une étude poussée des différentes définitions et proposer une modélisation du « contexte » pour le texte et sa sémantique afin de l'exploiter au mieux;
- **De l'annotation sémantique à l'annotation sémantique contextuelle** : l'annotation sémantique est de meilleure qualité et plus efficace en introduisant la notion de contexte.

Afin d'apporter des solutions à cette problématique, nous nous somme inspirés, d'une part, des travaux liés au TALN et au Web sémantique, et d'autre part, des travaux liés à l'exploitation de la notion de « contexte » dans le domaine de l'informatique pervasive. Ceci nous a amené à proposer l'approche CEEMA permettant de générer des annotations en prenant en compte leur contexte d'apparition dans le texte.

CEEMA apporte les réponses aux problèmes exposés au début de nos travaux et qui sont :

La modélisation des connaissances du domaine

Nous utilisons, d'une part, les ontologies offrant ainsi un cadre formel pour la représentation et l'organisation des connaissances du domaine, et d'autre part, les annotations sémantiques décrivant le contenu sémantique des ressources textuelles ;

L'approche CEEMA est basée sur une ontologie du domaine qui est donnée en entrée. Cette ontologie est la représention formelle de la connaissance du domaine pour notre approche.

Certains chercheurs restent réticents quant à la construction ou la réutilisation d'une ontologie existante, à cause de l'influence de l'application dans le choix de modélisation de l'ontologie.

Dans le cadre de nos expérimentations, l'ontologie de la fonderie ESTANDA a été construite dans notre équipe de recherche Edelweiss, avec une étroite collaboration des ingénieurs d'ESTANDA. Dans le cadre de nos recherches nous n'avons pas participé à sa construction. Mais nous l'avons utilisée comme entrée dans le cadre de l'expérimentation de notre approche.

L'annotation sémantique pour la recherche d'informations

Les besoins en recherche d'information sont divers. Une recherche par mots clés pourra répondre à un besoin simple et courant. Cependant, dans notre cas, le cadre industriel nous dicte certaines exigences telles que la précision et la qualité dans la recherche d'information. L'annotation sémantique associée à une ontologie du domaine ajoute une richesse supplémentaire pour la recherche d'information, où la formulation de requêtes sémantiques complexes est permise.

Néanmoins, dans la littérature, les annotations sémantiques telles qu'elles sont extraites à partir du texte et sous leur forme la plus sémantique (triplet : sujet, prédicat, objet), constituent une base d'annotations « peu structurée ». En effet, tous les triplets générés ont le même niveau dans la base d'annotations où les dépendances entre triplets sont faibles. Dans ce type de base d'annotations, les triplets générés sont tous supposés être vrais au même niveau. Par conséquent, nous pouvons trouver des triplets contradictoires dans la base d'annotations, alors que leur sémantique devait être vraie que dans un contexte particulier.

L'approche CEEMA permet de construire une base d'annotation où les triplets RDF générés sont emboités dans des graphes nommés selon la structure physique du texte. En outre, ces graphes nommés sont reliés entre eux par des relations contextuelles (rhétoriques, spatiales, temporelles).

Cette modélisation des annotations sémantiques contextuelles ouvre des perspectives dans la recherche d'information contextuelle. L'effort qui reste à faire dans ce cadre est le calcul de la portée sémantique d'une annotation. C'est-à-dire, une annotation donnée est-elle vraie dans toute la base d'annotation sans restriction ou est-elle vraie dans un contexte précis.

Les technologies du Web sémantique au service de l'annotation sémantique

La simplicité du choix de modélisation du contexte d'une annotation, à savoir, des annotations liées par des relations contextuelles, permet d'exploiter les moyens de raisonnement dans le domaine du Web sémantique. En effet, la représentation que nous avons adoptée est celle de RDF avec les graphes nommés où des moteurs sémantiques existant tels que Corese en permettent l'exploitation.

En outre, la description conceptuelle du domaine, à savoir l'ontologie, permet de décrire, entre autres, des concepts, des propriétés, des contraintes (domain et range) et des labels caractérisant le nom lisible par un humain d'un concept (ou propriété) dans le texte. Dans le processus de génération des annotations contextuelles, nous avons pu exploiter l'ontologie du domaine à trois niveaux :

- *Au niveau des labels* : nous avons pu exploiter les labels des concepts et des propriétés pour construire automatiquement des règles de grammaire JAPE permettant d'identifier des instances de concepts et de propriétés dans le texte. Ceci nous a permis d'automatiser une étape fastidieuse dans le processus de génération des annotations et connue pour être, usuellement, une tâche manuelle ;
- *Au niveau des contraintes* : nous avons pu exploiter les contraintes « domain et range » afin d'associer à une propriété donnée son sujet et son objet adéquat. Ceci nous permet de générer des annotations en adéquation avec le schéma ontologique ;
- *Au niveau de la hiérarchie de l'ontologie* : nous avons pu exploiter la hiérarchie de l'ontologie pour résoudre certaines ambigüités. Par exemple, pour une propriété donnée, choisir le sujet (instance de concept) ayant le concept le plus spécifique dans la hiérarchie de l'ontologie;

Par ailleurs, nous avons opté pour les technologies du Web sémantique dans notre approche du fait de leur standardisation. En effet, en utilisant les technologies du Web sémantique comme support, nous assurons une plus large utilisation de la base d'annotation générée. Ainsi, nous favorisons la réutilisation des annotations sémantique générés.

La modélisation du contexte pour le texte et les annotations sémantiques

Nous avons pu explorer l'utilisation du contexte dans d'autres domaines, et plus spécialement le domaine de l'informatique pervasive. Ceci nous a permis de mettre en relief certains aspects dans la modélisation des informations sur le contexte de type textuel tels que l'importance de l'aspect spatial qui peut être assimilé à la structure physique du texte.

Par ailleurs, plusieurs propositions de définition de la notion de « contexte » existent dans la littérature. La plupart d'entres elles assurent couvrir la majorité des aspects de cette notion. De notre côté, nous avons proposé une modélisation qui est le résultat d'une projection de plusieurs recommandations issues de ces définitions sur notre cadre de travail, à savoir, le texte et les annotations sémantiques.

Nous avons pu tirer profit de la richesse sémantique de la notion de « contexte », tout en gardant l'aspect applicatif en vue. De plus, nous avons retenu la majorité des caractéristiques exposées dans ces définitions avec une description des différents types de relations

contextuelles pouvant exister. Ces dernières concernent des relations issues de la structure physique du document, de la structure logique du document et d'autres relations sémantiques (temporelles et spatiales).

Du point de vue du texte, nous avons défini le contexte d'une entité textuelle (segment de texte, phrase, paragraphe, document, etc.) par les autres entités textuelles qui sont liées à elle par des relations contextuelles. Dans ce cas, les relations contextuelles peuvent être à titre d'exemple pour une phrase donnée, des relations d'*imbrications* avec le paragraphe englobant, de *successions* avec la phrase suivante, etc.

Du point de vue de la sémantique du texte, nous avons défini le contexte d'une annotation sémantique (représenté par des triplets) par les autres annotations sémantiques qui sont liées à elle par des relations contextuelles. Dans ce cas, les relations contextuelles peuvent être à titre d'exemple, des relations de discours (rhétoriques).

Cette modélisation des annotations contextuelles offre la possibilité de sélectionner, selon le besoin, les relations contextuelles à identifier et à exploiter.

De l'annotation sémantique à l'annotation sémantique contextuelle

Pour ce qui est de la génération des annotations, nous avons proposé une approche innovante et générique basée sur l'ontologie et les techniques de traitement automatique de la langue naturelle.

Le système que nous avons développé et qui implémente cette approche permet d'avoir en entrée des ressources textuelles brutes et une ontologie du domaine et retourne des instances de concepts reliées par des instances de relations de l'ontologie formant ainsi des triplets RDF (annotation sémantique). Les annotations sémantiques sont elles aussi liées entre elles par des relations contextuelles.

Les annotations sémantiques contextuelles sont donc représentées concrètement par des triplets RDF (annotations sémantiques) avec des imbrications de graphes nommés et des relations contextuelles entre ces graphes. Les relations contextuelles sont représentées par des propriétés (au sens ontologique) qui admettent des graphes nommés comme sujet et objet.

CEEMA est une approche qui permet de générer des annotations sémantiques contextuelles. L'exploitation du contexte pour les annotations sémantiques se situe à deux niveaux :

- *Au niveau de la génération des annotations sémantiques* : dans le processus de génération des annotations, le sujet et/ou l'objet d'une propriété donnée ne sont pas toujours présents dans le même segment de texte que la propriété. Par conséquent, nous avons proposé d'augmenter la portée de sélection des sujets candidats (ou objets candidats) sur une portion de texte plus englobante. L'aspect du contexte exploité dans cette partie concerne la structure physique du texte ;
- *Au niveau des relations contextuelles entre les annotations sémantiques* : les annotations sémantiques générées (triplets RDF) sont emboitées dans des graphes nommés suivant la localisation de la propriété de chaque triplet et suivant la structure physique du texte. En outre, les graphes nommés imbriquant les annotations sémantiques sont liés entre eux par des relations contextuelles (par exemple rhétorique). Les relations contextuelles entre graphes nommés et les imbrications hiérarchiques entre eux illustrent,

respectivement, deux aspects exploités du contexte, à savoir, la structure logique et physique du texte.

Les annotations sémantiques prennent en compte la notion de « contexte », nous détenons donc des annotations plus riches sémantiquement qui exploitent la structure physique et logique du texte ainsi que d'autres relations sémantiques (temporelles et spatiales).

Afin de montrer l'apport du « contexte » pour les annotations sémantiques (triplets RDF), nous avons effectué plusieurs expérimentations sur le processus de génération des annotations sémantiques pour le même corpus textuel. Le paramètre que nous avons modifié dans ces expérimentations est la portée maximale à considérer (ETp_max) pour construire un triplet donné (associer à une propriété son sujet et son objet adéquats). Nous avons effectué ces expérimentation avec les portées « ETp-max » suivantes : i) le segment de texte où une propriété est identifiée; ii) le segment de texte où une propriété est identifiée avec la prise en compte de la phrase du titre ; ii) la phrase englobant le segment de texte avec la prise en compte de la phrase du titre ; iii) le paragraphe englobant la phrase précédente avec la prise en compte de la phrase du titre ; vi) le section englobant le paragraphe segment précédant avec la prise en compte de la phrase du titre.

Nous avons calculé la précision et le rappel pour chaque expérimentation et nous avons déduit de ces expérimentations que le changement de la portée maximale a un impact direct et significatif dans les résultats de génération des annotations sémantiques. En effet, nous avons constaté, entre autres, qu'une portée restreinte (segment de texte) limite le nombre d'annotations générées. En parallèle, une large portée (section) ne conduit pas forcément à plus d'annotations générées mais produit plus d'annotations erronées.

Avec ces expérimentations nous avons pu, entre autres :

- Montrer l'impact que peut avoir la notion de contexte sur la qualité des annotations sémantique ;
- Déduire un juste milieu dans la détermination de la portée maximale considéré, à savoir, le paragraphe ;
- Constater que la phrase du titre dans le texte peut avoir un impact significatif dans l'amélioration de la qualité des résultats de génération des annotations sémantiques.

Nos travaux et contribution on fait l'objet de quatre publications dont trois internationales [Mokhtari et al. 2008][Mokhtari et al. 2009a][Mokhtari et al. 2009b] [Mokhtari et al. 2009c]. Nous avons pu faire connaître ces travaux auprès de communautés spécialisées d'ingénierie des connaissances et du Web sémantique.

En plus des publications scientifiques, nos travaux ont été exposés, dans le cadre du projet européen SevenPro, à nos partenaires industriels où ils ont participé avec leurs remarques et en manifestant leurs besoins réels dans le domaine d'extraction d'information et annotations sémantiques à partir de texte.

Limites et perspectives

Nous avons pu respecter la majorité des exigences par rapport à la modélisation du contexte issues du domaine de l'informatique pervasive. Néanmoins, certaines exigences sont difficiles à mettre en œuvre pour le texte et sa sémantique. En effet, nous citons en particulier la gestion des historiques du texte. Cette exigence entraine la gestion du cycle de vie du document et de pouvoir gérer l'évolution de la base d'annotation tout en respectant la cohérence sémantique. Ceci reste un problème scientifique ouvert et qui mérite un intérêt particulier.

L'approche CEEMA proposée pour la génération automatique d'annotations sémantiques contextuelles à partir de textes est générique et réutilisable pour n'importe quel domaine. Néanmoins, les bons résultats obtenus par CEEMA-T, le système implémentant cette approche, sont positivement influencés par la forte composante « terminologique » de l'ontologie. En effet, toutes les classes et propriétés de l'ontologie ont des descriptions « rdfs:label » dans la langue correspondant à la langue des textes.

Nous pensons donc qu'une réutilisation de CEEMA-T en utilisant une autre ontologie devrait nécessiter un travail en amont sur l'enrichissement et le peuplement de cette ontologie. Ce travail pourrait être facilité pour les experts du domaine en utilisant des outils de TALN tel que Nomino [David et al. 1990] ou Likes [Ouesleti et al., 1996] pour le peuplement de la hiérarchie des concepts de l'ontologie, ou Syntex [Bourigault et al., 2005] pour enrichir la hiérarchie de propriétés à travers l'extraction de syntagmes verbaux.

Nous avons pu valider notre approche avec des expérimentations sur un corpus textuel issu de nos partenaires industriel. Cependant, des expérimentations sur d'autres corpus serait un plus significatif pour notre approche. D'autant plus que notre approche est générique et facilement exploitable pour d'autre corpus textuel. La difficulté réside dans l'étape de validation. En effet, cette étape est fastidieuse si elle est faite manuellement par des experts du domaine comme c'était le cas dans notre expérimentation. La difficulté réside dans le fait que les annotations générées, même si elles sont erronées, respectent le schéma ontologique. C'est-à-dire, une annotation est erronée si le sujet ou l'objet d'une propriété ne sont pas ceux que la sémantique du texte voulait transmettre, même si le triplet généré respecte le schéma de l'ontologie. Dans ce cas, l'automatisation de la validation des résultats, par exemple, par des requêtes lancées sur la base d'annotation ne sera pas suffisante, et l'intervention humaine pour la validation des annotations générées est requise.

L'approche CEEMA est une approche qui produits des annotations sémantiques contextuelles qui prend en compte la structure physique et logique du texte. Elle est basée, en partie, sur une chaîne de traitement automatique implantée par GATE. Ce type de chaîne de traitement est connu pour être séquentiel. D'un point de vu performance, cette approche séquentielle représente un frein pour une utilisation à plus grande échelle de notre approche. En effet, Dans une perspective d'adaptation de notre approche pour des pages Web, la performance dans cette étape pèse lourd dans le temps de traitement global.

Une solution applicative est apparue récemment, proposée par Apache, dite « UIMA » (Unstructured Information Management Applications) pour l'analyse du texte. L'avantage initial d'UIMA est la possibilité de paralléliser l'exécution des composants de traitements. Nous avons eu la possibilité de participer, dans le cadre de « First French-speakins meeting around the framework Apache UIMA »[31], à un tutorial nous permettant ainsi de constater les

[31] http://uima-fr.org/RMLL-cfp.html

avantages de l'utilisation d'UIMA. Cependant, au moment où l'implantation de notre approche (CEEMA-T) a été effectuée, les composants développés autour de cette technologie n'étaient pas suffisant pour mettre en œuvre notre méthodologie. A l'heure actuelle, beaucoup de travaux ont émergé et qui couvrent largement la chaîne de traitement dont nous avons besoin pour notre approche. C'est pourquoi, les orientations futures de nos travaux exploitant l'analyse du texte évolueraient vers la technologie UIMA.

Perspectives à court et moyen terme

Une première perspective à court terme pour ce travail consiste à proposer un module d'enrichissement de l'ontologie. Ce module pourrait être basé sur des outils de TALN tels que Faster [Jacquemin et al., 1997] afin d'extraire des termes candidats pour enrichir la hiérarchie des concepts de l'ontologie. L'idée que nous étudions actuellement est la suivante :

- Lors du déroulement de l'algorithme de génération des triplets RDF, certaines propriétés ne sont associées à aucun sujet et/ou objet, même en utilisant les entités textuelles englobantes. Nous proposons d'élargir les candidats sujets (ou objets) d'une propriété donnée « p » aux termes supplémentaires extraits (non existants dans l'ontologie). Un terme est choisi comme sujet candidat pour une propriété « p » s'il est proche de cette propriété dans le texte ;
- La validation des sujets candidats (termes) peut être soumise aux experts du domaine. Les experts du domaine peuvent valider un terme supplémentaire extrait comme instance d'une (ou plusieurs) classe dans l'ontologie, ou créer une nouvelle classe dont le terme est une instance de cette classe.

Une autre amélioration de CEEMA mérite une attention particulière telle que l'étude des anaphores grammaticales (e.g. *ce dernier*, *le mien*, etc.). En effet, nous avons constaté que nombreuses sont les instances de classes potentielles qui se présentent sous forme d'anaphores grammaticales dans le texte. Ces instances peuvent représenter des sujets (ou objets) dans de nouveaux triplets générés.

La représentation que nous avons adoptée pour les annotations sémantiques contextuelles est RDF et les graphes nommés. Cette représentation permet d'exploiter ces annotations contextuelles ainsi que les relations contextuelles de la même manière que les annotations sémantiques et les relations du domaine. Cependant, une étude plus approfondie est nécessaire pour exploiter la sémantique de chaque relation contextuelle. Notamment, la portée sémantique des annotations :

- Est ce qu'une annotation sémantique donnée (ensemble de triplets RDF) est valable (ou vraie) relativement au reste de la base d'annotations ?
- Ou n'est-elle valable que lorsque certaines annotations le sont ?

Une complexité supplémentaire s'ajoute si les relations contextuelles sont de type temporel ou spatial.

Perspectives à long terme

Une première perspective à long terme concerne l'importance de la gestion de l'évolution des ontologies et des annotations. En effet, une évolution de l'ontologie, au niveau de sa structure hiérarchique ou au niveau des connaissances qu'elle modélise, peut générer des incohérences dans la base d'annotations. Ces incohérences peuvent induire en erreur les inférences effectuées sur les connaissances contenues dans ces annotations. Une étude intéressante concernerait l'apport des annotations sémantiques *contextuelles* à la gestion de l'évolution des ontologies et de la base d'annotations.

La visualisation des annotations contextuelles prend une part importante dans l'exploitation des informations contextuelles. En effet, une étude est nécessaire afin d'adopter le meilleur moyen pour visualiser ces annotations contextuelles.

L'approche d'extraction d'annotations sémantiques contextuelles « CEEMA » basée sur les ontologies que nous avons proposée peut être utilisée pour d'autres tâches que la recherche d'informations et l'aide au raisonnement. En effet, nous pouvons, par exemple, proposer des résumés de textes guidés par les ontologies et basés sur les annotations sémantiques contextuelles générées.

Finalement, il serait intéressant de tester l'approche dans d'autres domaines d'application, où la validation des résultats serait, d'une part, basée sur des requêtes lancées dans un module de recherche d'information contextuelle, et d'autre part, orchestrée par les experts du domaine.

Bibliographie

[Abrouk, 2006] L. Abrouk., Annotation de documents par le contexte de citation basée sur une ontologie, *Thèse de doctorat en informatique*, Montpellier, 2006.

[Adrian, et al. 2008] B. Adrian and A. Dengel: Believing finite-state cascades in knowledge based information extraction; In *Proc. of KI 2008: Advances in Artificial Intelligence, LNCS*, pages 152–159. Springer, 2008.

[Adrian et al. 2009a] B. Adrian, J. Hees, L. van Elst, and A. Dengel. iDocument: Using ontologies for extracting and annotating information from unstructured text. *KI 2009: Advances in Artificial Intelligence*, 5803/2009:249–256, 2009.

[Adrian et al. 2009b] B. Adrian, H. Maus, M. Kiesel and A. Dengel, Towards ontology-based information extraction and annotation of paper documents for personalized knowledge acquisition. In: *Proceedings of the First International Workshop on Personal Knowledge Management*, (Bonner Köllen Verlag, Solothurn, Switzerland, 2009).

[Adrian et al. 2009c] B. Adrian, J. Hees, L. van Elst and A. Dengel, iDocument: using ontologies for extracting and annotating information from unstructured text. In: *Proceedings of the 32nd Annual German Conference on AI*, (Springer-Verlag, Heidelberg, 2009).

[Allen, 1984] J.F. Allen, Towards a general theory of action and time, *Artificial Intelligence*, 1984.

[Amardeilh, 2007] F. Amardeilh, Web Sémantique et Informatique Linguistique: propositions méthodologiques et réalisation d'une plateforme logicielle, *Thèse de doctorat, Discipline : Informatique, Université Paris X* – Nanterre, Mai 2007.

[Bateman, 1990] J.A. Bateman, A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model, *tech. report, Information Sciences Inst*., Univ. of Southern California, Marina del Rey, Calif., 1990.

[Berners-Lee et al., 2001] T. Berners-Lee, J. Hendleret, O. Lassila, The Semantic Web, *In Scientific American*, May 2001, p35-43

[Berri, 96] J. Berri, Mise en œuvre de la méthode d'exploration contextuelle pour le résumé automatique de textes : Implémentation du système SERAPHIN. *Actes du colloque CLIM'96*, Montréal, Canada, 8-11 Juin 1996.

[Bettini et al. 2009] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, June 2009.

[Brin, 1998] S. Brin, Extracting Patterns and Relations from the World Wide Web in *WebDB Workshop at 6th International Conference on Extending Database Technology*, (1998).

[Borkar et al. 2001] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. *Proc. ACM SIGMOD, International conference Management Of Data*,. pp.175-186. 2001.

[Buitelaar et al. 2006] P. Buitelaar and M. Siegel, Ontology-based Information Extraction with SOBA. *In: Proceedings of the Fifth International Conference on Language Resources and Evaluation, European Language Resources Association*, Genoa, Italy, 2006.

[Buitelaar et al. 2008] P. Buitelaar, P. Cimiano, A. Frank, M. Hartung, S. Racioppa: Ontology-based Information Extraction and Integration from Heterogeneous Data Sources. *Int. Journal of Human-Computer Studies* (11), 759–788 (2008)

[Castelli et al. 2007] G. Castelli, A. Rossi, M. Mamei and F. Zambonelli, A simple model and infrastructure for context-aware browsing of the world, *Proceedings of the 5th IEEE Conference on Pervasive Computing and Communications*, IEEE Computer Society (2007).

[Cimiano et al. 2004] P. Cimiano, S. Handschuh and S. Staab, Towards the self-annotating web. In: *Proceedings of the 13th International Conference on World Wide Web* (ACM, New York, 2004).

[Cimiano et al. 2005] P. Cimiano, G. Ladwig and S. Staab, Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In: *Proceedings of the 14th International Conference on World Wide Web* (ACM, New York, 2005).

[Ciravegna et al. 2002] F. Ciravegna, A. Dingli, Y.Wilks, and D. Petrelli. Amilcare: adaptive information extraction for document annotation. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 367–368, New York, NY, USA, 2002. ACM Press.

[Croft et al. 1995] W.B. Croft, Lab report special section: *the University of Massachusetts Center for Intelligent Information Retrieval*, SIGIR Forum 29(1) (1995) 1–7.

[Corcho, 2006] O. Corcho, Ontology based document annotation: trends and open research problems, *in International Journal of Metadata, Semantics and Ontologies*, 1(1), Inderscience, 2006, pp. 47-57.

[Cunningham et al. 2000] H. Cunningham, D. Maynard and V. Tablan, JAPE: A Java Annotation Patterns Engine, *Department of Computer Science, University of Sheffield*, 2000.

[Cunningham et al. 2002] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 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), 2002.

[David et al. 1990] S. David et P. Plante: De la nécessité d'une approche morphosyntaxique dans l'analyse de textes. *Intelligence artificielle et sciences cognitives au Québec*, 3(3), pp.140-154, 1990.

[Declerck et al. 2008] T. Declerck, C. Federmann, B. Kiefer and H-U. Krieger. Ontology-based information extraction and reasoning for business intelligence applications. In: *Proceedings of the 31st Annual German Conference on Advances in Artificial Intelligence* (Demos), Springer-Verlag, Berlin, 2008.

[Desclés et al. 1994] J.-P Desclés and J.-L. Minel. L'exploration contextuelle, in Le résumé par exploration contextuelle, rapport interne du CAMS n 95/1, *recueil des communications effectuées aux rencontres Cognisciences-Est*, 25 novembre 1994, Nancy, pp.3-17.

[Desmontils et al. 2002a] E. Desmontils and C. Jacquin. Indexing a web site with a terminology oriented ontology. The Emerging Semantic Web, I.F. Cruz, S. Decker, J. Euzenat and D. L. McGuinness Ed, pages 181–197, 2002.

[Desmontils et al. 2002b] E. Desmontils, C. Jacquin, and E. Morin. Indexation sémantique de documents sur le web : application aux ressources humaines. In Proceedings of Journées de l'AS-CNRS Web sémantique, Octobre 2002.

[Dey et al. 2001] A. Dey, Understanding and using context, *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing* 5, 1, pp. 4–7. (2001).

[Dieng-Kuntz, 2005] R. Dieng-Kuntz. Corporate Semantic Webs. *In Encyclopaedia of Knowledge Management*, D. Schwartz ed, Idea Publishing Group, September 2005.

[Dobson et al. 2006] S. Dobson and J. Ye, Using fibrations for situation identification, *Proceedings of Pervasive 2006 Workshops*, Springer, 2006.

[Egenhofer et al., 1991] M. Egenhofer et R. Franzosa, Point-Set Topological Spatial Relations, *International Journal of Geographical Information Systems*, vol. 5, no. 2, pp. 161-174, 1991.

[Euzenat, 2005] J. Euzenat. L'annotation formelle de documents en 8 questions, *in Ingénierie des connaissances,* Teulier R., Charlet J & Tchounikine P., L'Harmattan, Paris, 2005, pp. 251-271.

[Groza et al. 2007] T. Groza, S. Handschuh, K. Moeller, S. Decker. SALT: A Semantically Annotated Latex for Scientific Publications. *In ESWC*, vol. 4519 of lecture notes in Computer Science, pp. 518-532. Springer, *2007*, Innsbruck, Austria.

[Guarino et al. 1999] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web. IEEE Intelligent Systems, May-June 4-5:70–80, 1999.

[Goujon, 1999a] B. Goujon. Competitive Intelligence: extraction of relevant information with the contextual exploration method. *In P. Lenca, editor, Proceedings of the Human Centered Processes Conference*, Brest, France, pages 233-240, September 1999.

[Goujon, 1999b] B. Goujon. Extraction d'information technique pour la veille par l'exploitation de notions indépendantes d'un domaine, *TIA'99, in Terminologies Nouvelles*. 1999.

[Embley et al. 2004] D.W. Embley, Toward semantic understanding: an approach based on information extraction ontologies. In: *Proceedings of the 15th Australasian Database Conference* (Australian Computer Society, Darlinghurst, Australia, 2004).

[Handschuh, 2005] S. Handschuh. Creating Ontology-based Metadata by Annotation for the Semantic Web, *Thèse de doctorat, Université de Karlsruhe*, 2005, 225 p.

[Henricksen et al. 2006] K. Henricksen and J. Indulska, Developing context-aware pervasive computing applications: Models and approach, *Pervasive and Mobile Computing* 2 (1) (2006), pp. 37–64.

[Hernandez, 2005] Hernandez N., Ontologies de domaine pour la modélisation du contexte en recherche d'information, *Thèse de doctorat en informatique*, Toulouse, 2005.

[Humphreys et al. 1993] B. Humphreys and D. Lindberg: The UMLS project: making the conceptual connection between users and the information they need. *Bull. of the Medical Library Assoc.*81(2): 170, 1993.

[Indulska et al. 2003] J. Indulska, R. Robinson, A. Rakotonirainy and K. Henricksen, Experiences in Using CC/PP in Context-Aware Systems. In: M.-S. Chen, P.K. Chrysanthis, M. Sloman and A.B. Zaslavsky, Editors, *Mobile Data Management, Lecture Notes in Computer Science* vol. 2574**,** Springer (2003).

[Jacquemin et al., 1997] C. Jacquemin, J. L. Klavans et E. Tzoukermann, Expansion of multi-word terms for indexing and retrieval using morphology and syntax. *In Proceedings, 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL – EACL'97)*, Madrid, 1997.

[Khelif et al. 2007] K. Khelif, R. Dieng-Kuntz and P. Barbry, An ontology-based approach to support text mining and information retrieval in the biological domain, *Special Issue on Ontologies and their Applications of the Journal of Universal Computer Science (JUCS)*, Vol. 13, No. 12, pp. 1881-1907, 2007.

[Kiryakov et al. 2005] A. Kiryakov, B. Popov, I. Terziev, D. Manov, A. Kirilov, M.Goranov. Semantic annotation, indexing, and retrieval. In *J. Web Semantics, Science, Services and Agents on the WWW*, 2(1), Elsevier. p. 49-79. 2005.

[Klyne et al. 2004] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M.H. Butler, L. Tran, Composite capability/preference profiles (CC/PP): Structure and vocabularies 1.0, *W3C Recommendation*, Tech. Rep., W3C, January 2004.

[Kshitija et al. 2008] P. Kshitija, N. Patil, S. Patankar, C. Das, "A Survey on Web Content Mining and Extraction of Structured and Semistructured Data," ICETET, pp.543-546, 2008 *First International Conference on Emerging Trends in Engineering and Technology*, 2008.

[Laclavik et al. 2007] M. Laclavik, M. Seleng, E. Gatial, Z. Balogh, L. Hluchy: Ontology based Text Annotation OnTeA; Information Modelling and Knowledge Bases XVIII. IOS Press, Amsterdam, Marie Duzi, Hannu Jaakkola, Yasushi Kiyoki, Hannu Kangassalo (Eds.), Frontiers in *Artificial Intelligence and Applications*, Vol. 154, ISBN 978-1- 58603-710-9, ISSN 0922-6389, (2007) 311315.

[Laclavik et al. 2009] M. Laclavik, M. Seleng, M. Ciglan, L. Hluchy: Ontea: Platform for Pattern Based Automated Semantic Annotation. *Computing and Informatics*, Vol. 28, 2009, No. 4, pp. 555–579.

[Li et al. 2007] Y. Li and K. Bontcheva, Hierarchical perceptron-like learning for ontology-based information extraction. *In: Proceedings of the 16th International Conference on World Wide Web. ACM*, New York, 2007.

[Mann et al. 1987] W. C. Mann et S. A. Thompson. Rhetorical Structure Theory: A theory of Text Organization, *Rapport technique, ISI-RS-87-190, Information Sciences Institute*, Marina Del Rey, Ca, 1987.

[Maynard et al. 1999] B. Maynard, S. Ananiadou. Identifying Contextual Information for Multi-Word Term Extraction. *In Proceedings of Terminology and Knowledge Engineering Conference*. 99. pp. 212-221. Innsbruck, Austria, 1999.

[Maynard et al. 2001] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. Named Entity Recognition from Diverse Text Types. *In Recent Advances in Natural Language Processing 2001 Conference*, pages 257-274, Tzigov Chark, Bulgaria, 2001.

[McCarthy, 1993] J. McCarthy, *Notes on formalizing context*, Proc. 13th IJCAI, pp.555–560, California. 1993.

[McDowell et al. 2006] L. McDowell and M.J. Cafarella, Ontology-driven information extraction with OntoSyphon. *In: Proceedings of the 5th International Semantic Web Conference*. Springer, Berlin, 2006.

[Miller, 1990] G. A. Miller, WordNet: an Online Lexical Database, International Journal of Lexicography, 3(4), 1990, pp. 235-312.

[Mitchell et al. 1982] T.M. Mitchell, Generalization as search, *Artificial Intelligence*, 18(2) 203–226. 1982.

[Mokhtari et al. 2008] N. Mokhtari et R. Dieng-Kuntz, Extraction et exploitation des annotations contextuelles, *Extraction et gestion des connaissances (EGC'2008)*, 7-18, Volumes, Sophia-Antipolis, France, 29 janvier au 1er février 2008.

[Mokhtari et al. 2009a] N. Mokhtari et O. Corby, Modelling and automatic extracting of contextual semantic annotations. *Proceedings of I-KNOW'09, the 9th International Conference on Knowledge Management*, Graz, Austria, September 2-4, 2009.

[Mokhtari et al. 2009b] N. Mokhtari et O. Corby, Contextual semantic annotations: modelling and automatic extraction. *In Poster at K-CAP 2009 The Fifth International Conference on Knowledge Capture*, California, USA, September 2009.

[Mokhtari et al. 2009c] N. Mokhtari et O. Corby, Contextual semantic annotations: modelling and automatic extraction. *Proceedings of Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM 2009), co-located with the 5th International Conference on Knowledge Capture (K-CAP 2009)*, September 2009.

[Nédellec, 2004] C. Nédellec, Machine Learning for Information Extraction in Genomics - State of the Art and Perspectives, *In Text Mining and its Applications: Results of the NEMIS Launch Conference Series: Studies in Fuzziness and Soft Computing*, Sirmakessis, Spiros (Ed.), Springer Verlag, 2004.

[Njmogue, 2004] W. Njmogue, D. Fontaine et P. Fontaine, Identification des thèmes d'un document relativement à un référentiel métier, *In Proceedings of MAJECSTIC'04*, Calais, France, 2004.

[Ouesleti et al. 1996] R. Ouesleti, P. Frath, F. Rousselot, A corpus-based method for acquisition and exploitation of terms, *CLIM-96, Student Conference in Computational Linguistics in Montreal*, Universite de Montreal, Canada, 1996.

[Pascoe, 1998] J. Pascoe. Adding generic contextual capabilities to wearable computers. *In: Proceedings of 2nd International Symposium on Wearable Computers*, 1998; 92–99.

[Prié et al. 2005] Y. Prié, S. Garlatti: Métadonnées et annotations dans le Web Sémantique, *In. Charlet J., Laubet P., Reynaud C. eds. Web sémantique, numéro hors-série de la revue « interaction, information, intelligence »*. Editions Cépaduès/Toulouse. 2005.

[Popov et al. 2003] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff and M. Goranov, KIM – semantic annotation platform. In: *Proceedings of the 2nd International Semantic Web Conference.* Springer-Verlag, Berlin, 2003.

[Popov et al. 2004] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov and A. Kirilov, KIM – a semantic platform for information extraction and retrieval, *Journal of Natural Language Engineering* 10(3–4) (2004) 375–392.

[Reeve et al. 2005] L. Reeve, H. Han, "Survey of semantic annotation platforms", Proc. of the 2005 *ACM Symposium on Applied Computing*, ACM Press, 2005, pp. 1634-1638.

[Rinaldi et al., 2003] F. Rinaldi, J. Dowdall, M. Hess, J. Ellman, G.-P. Zarri, A. Persidis, Bernard L., Karanikas H., Multilayer annotations in Parmenides, *in Proceedings of the Knowledge Markup and Semantic Annotation Workshop*, Sanibel, Florida, USA, 2003, pp.33-40.

[Saggion et al. 2007] H. Saggion, A. Funk, D. Maynard and K. Bontcheva, Ontology-based information extraction for business intelligence. In: *Proceedings of the 6th International and 2nd Asian Semantic Web Conference*. Springer, Berlin, 2007.

[Schilit et al. 1994] B. Schilit, N. Adams and R. Want, Context-aware Computing Applications, *Xerox Corp., Palo Alto Research Center*. 1994.

[Schilit et al. 1994] B. Schilit, M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network 1994*; 8: 22–32. 1994.

[Strang et al. 2004] T. Strang and C. Linnhoff-Popien, A Context Modeling Survey. In: J. Indulska and D.D. Roure, Editors, *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management, in conjunction with UbiComp 2004*, University of Southampton. 2004.

[Séguéla et al. 1999] P. Séguéla et N. Aussenac-Gilles: Extraction de relations sémantiques entre termes et enrichissement de modèles du domaine. *In IC'99*, Paris, 79-88.

[Sowa, 1984] J. Sowa, Conceptual Structures: Information Processing in Mind and Machine, *Addison-Wesley, Reading, Mass*., 1984.

[Soderland et al. 1995] S. Soderland, D. Fisher, J. Aseltine and W.G. Lehnert, CRYSTAL: inducing a conceptual dictionary. *In: Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Montreal, Canada, 1995.

[Todirascu et al. 2002] A. Todirascu, L. Romary and D. Bekhouche, Vulcain – an ontology-based information extraction system. In: *Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers*. Springer-Verlag, London, 2002.

[Vargas-Vera et al. 2001] M. Vargas-Vera, E. Motta, J. Domingue, S.B. Shum and M. Lanzoni, Knowledge extraction by using an ontology-based annotation tool. In: *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation*. ACM, New York, 2001.

[Vargas-Vera et al. 2002] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup in *The 13th International Conference on Knowledge Engineering and Management (EKAW 2002)*, Spain, 379-391, 2002.

[Wanget al. 2004] X.H. Wang, T. Gu, D.Q. Zhang and H.K. Pung, Ontology based context modeling and reasoning using OWL, *Proceedings of Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society. 2004.

[Weld et al. 2008] D.S. Weld, R. Hoffmann and F. Wu, Using Wikipedia to bootstrap open information extraction, *SIGMOD Record* 37(4) 62-68. 2008.

[Wimalasuriya et al. 2010] D. C. Wimalasuriya and D. Dou. Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches. *Journal of Information Science (JIS)*. Vol. 36 no. 3, pp 306-323, 2010.

[Wu et al. 2007] F. Wu and D. S. Weld, Autonomously semantifying Wikipedia. *In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, (ACM, New York, 2007).

[Wu et al. 2008] F. Wu, R. Hoffmann, and D. S. Weld, Information extraction from Wikipedia: moving down the long tail. *In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, 2008.

[Yildiz et al. 2007] B. Yildiz and S. Miksch, OntoX – a method for ontology-driven information extraction. In: *Proceedings of the 2007 International Conference on Computational Science and its Applications*, Springer, Berlin, 2007.

[Zimmermann et al. 2007] A. Zimmermann, A. Lorenz et R. Oppermann. An Operational Definition of Context, Modeling and Using Context, pp. 558-571. 2007.

Annexe 1 : Relations contextuelles et exemples de règles JAPE associées

After
Although
Also
As Well Conversely As
As Long As
As A Consequence
As A Result
As Soon As
As Far As
As Well As
As
Because
But
By Contrast
By Compare
Compare To
Conversely
Despite The Fact
Even Though
Even So
Except
It Follows That
It May Be Concluded
If
In Contrast
Indeed
Instead
In This Way
In Short
In So Doing
In So Far As
In Spite Of
In Other Words
First
For Example
For The Reason That
For This Reason
For That Reason
For One Thing
Furthermore
Further
Having Said That
However
Nevertheless
Nonetheless
Nor
Notwithstanding
Much As
Regardless Of
Second
Since
So That
So
Such As
Summing Up
Till
That Said
This Be Why
Therefore
Thereafter
Then
Then Again
Though
To Sum Up
To Summarise
To Summarize
Otherwise
On The Contrary
On The Grounds That
On The Other Hand
Put Another Way
Unless
Until
While
When
Which Be Why
Which
Where
Whereas
Yet

Exemples de règles JAPE engendrées automatiquement correspondant à la liste ci-dessus.

```
phase: first
options: control = appelt
Rule:JRuleAfter
(
{Token.lemma == "after"}({SpaceToken})?
 ):After -->
:After.ContextualRelation= {kind = "after", rule=JRuleAfter}

phase: first
options: control = appelt
Rule:JRuleAlthough
(
{Token.lemma == "although"}({SpaceToken})?
 ):Although -->
:Although.ContextualRelation= {kind = "although", rule=JRuleAlthough}


phase: first
options: control = appelt
Rule:JRuleAsWellConverselyAs
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"well"}({SpaceToken})?{Token.lemma ==
"conversely"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):AsWellConverselyAs -->
:AsWellConverselyAs.ContextualRelation= {kind = "aswellconverselyas",
rule=JRuleAsWellConverselyAs}

phase: first
options: control = appelt
Rule:JRuleAsLongAs
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"long"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):AsLongAs -->
:AsLongAs.ContextualRelation= {kind = "aslongas", rule=JRuleAsLongAs}

phase: first
options: control = appelt
Rule:JRuleAsAConsequence
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"a"}({SpaceToken})?{Token.lemma == "consequence"}({SpaceToken})?
 ):AsAConsequence -->
:AsAConsequence.ContextualRelation= {kind = "asaconsequence",
rule=JRuleAsAConsequence}

phase: first
options: control = appelt
Rule:JRuleAsAResult
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"a"}({SpaceToken})?{Token.lemma == "result"}({SpaceToken})?
 ):AsAResult -->
:AsAResult.ContextualRelation= {kind = "asaresult", rule=JRuleAsAResult}

phase: first
options: control = appelt
```

```
phase: first
options: control = appelt
Rule:JRuleAsWellAs
(
{Token.lemma == "as"}({SpaceToken})?{Token.lemma ==
"well"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):AsWellAs -->
:AsWellAs.ContextualRelation= {kind = "aswellas", rule=JRuleAsWellAs}

phase: first
options: control = appelt
Rule:JRuleAs
(
{Token.lemma == "as"}({SpaceToken})?
 ):As -->
:As.ContextualRelation= {kind = "as", rule=JRuleAs}

phase: first
options: control = appelt
Rule:JRuleBecause
(
{Token.lemma == "because"}({SpaceToken})?
 ):Because -->
:Because.ContextualRelation= {kind = "because", rule=JRuleBecause}

phase: first
options: control = appelt
Rule:JRuleBut
(
{Token.lemma == "but"}({SpaceToken})?
 ):But -->
:But.ContextualRelation= {kind = "but", rule=JRuleBut}

phase: first
options: control = appelt
Rule:JRuleByContrast
(
{Token.lemma == "by"}({SpaceToken})?{Token.lemma ==
"contrast"}({SpaceToken})?
 ):ByContrast -->
:ByContrast.ContextualRelation= {kind = "bycontrast", rule=JRuleByContrast}

phase: first
options: control = appelt
Rule:JRuleByCompare
(
{Token.lemma == "by"}({SpaceToken})?{Token.lemma ==
"compare"}({SpaceToken})?
 ):ByCompare -->
:ByCompare.ContextualRelation= {kind = "bycompare", rule=JRuleByCompare}

phase: first
options: control = appelt
Rule:JRuleCompareTo
(
{Token.lemma == "compare"}({SpaceToken})?{Token.lemma ==
"to"}({SpaceToken})?
 ):CompareTo -->
:CompareTo.ContextualRelation= {kind = "compareto", rule=JRuleCompareTo}

phase: first
```

```
options: control = appelt
Rule:JRuleExcept
(
{Token.lemma == "except"}({SpaceToken})?
 ):Except -->
:Except.ContextualRelation= {kind = "except", rule=JRuleExcept}

phase: first
options: control = appelt
Rule:JRuleIf
(
{Token.lemma == "if"}({SpaceToken})?
 ):If -->
:If.ContextualRelation= {kind = "if", rule=JRuleIf}

phase: first
options: control = appelt
Rule:JRuleHavingSaidThat
(
{Token.lemma == "having"}({SpaceToken})?{Token.lemma ==
"said"}({SpaceToken})?{Token.lemma == "that"}({SpaceToken})?
 ):HavingSaidThat -->
:HavingSaidThat.ContextualRelation= {kind = "havingsaidthat",
rule=JRuleHavingSaidThat}

phase: first
options: control = appelt
Rule:JRuleHowever
(
{Token.lemma == "however"}({SpaceToken})?
 ):However -->
:However.ContextualRelation= {kind = "however", rule=JRuleHowever}

phase: first
options: control = appelt
Rule:JRuleNevertheless
(
{Token.lemma == "nevertheless"}({SpaceToken})?
 ):Nevertheless -->
:Nevertheless.ContextualRelation= {kind = "nevertheless",
rule=JRuleNevertheless}


phase: first
options: control = appelt
Rule:JRuleNotwithstanding
(
{Token.lemma == "notwithstanding"}({SpaceToken})?
 ):Notwithstanding -->
:Notwithstanding.ContextualRelation= {kind = "notwithstanding",
rule=JRuleNotwithstanding}

phase: first
options: control = appelt
Rule:JRuleMuchAs
(
{Token.lemma == "much"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):MuchAs -->
:MuchAs.ContextualRelation= {kind = "muchas", rule=JRuleMuchAs}

phase: first
```

```
options: control = appelt
Rule:JRuleRegardlessOf
(
{Token.lemma == "regardless"}({SpaceToken})?{Token.lemma ==
"of"}({SpaceToken})?
 ):RegardlessOf -->
:RegardlessOf.ContextualRelation= {kind = "regardlessof",
rule=JRuleRegardlessOf}

phase: first
options: control = appelt
Rule:JRuleSecond
(
{Token.lemma == "second"}({SpaceToken})?
 ):Second -->
:Second.ContextualRelation= {kind = "second", rule=JRuleSecond}

phase: first
options: control = appelt
Rule:JRuleSince
(
{Token.lemma == "since"}({SpaceToken})?
 ):Since -->
:Since.ContextualRelation= {kind = "since", rule=JRuleSince}

phase: first
options: control = appelt
Rule:JRuleSoThat
(
{Token.lemma == "so"}({SpaceToken})?{Token.lemma == "that"}({SpaceToken})?
 ):SoThat -->
:SoThat.ContextualRelation= {kind = "sothat", rule=JRuleSoThat}

phase: first
options: control = appelt
Rule:JRuleSo
(
{Token.lemma == "so"}({SpaceToken})?
 ):So -->
:So.ContextualRelation= {kind = "so", rule=JRuleSo}

phase: first
options: control = appelt
Rule:JRuleSuchAs
(
{Token.lemma == "such"}({SpaceToken})?{Token.lemma == "as"}({SpaceToken})?
 ):SuchAs -->
:SuchAs.ContextualRelation= {kind = "suchas", rule=JRuleSuchAs}

phase: first
options: control = appelt
Rule:JRuleSummingUp
(
{Token.lemma == "summing"}({SpaceToken})?{Token.lemma ==
"up"}({SpaceToken})?
 ):SummingUp -->
:SummingUp.ContextualRelation= {kind = "summingup", rule=JRuleSummingUp}

phase: first
options: control = appelt
Rule:JRuleTherefore
```

```
(
{Token.lemma == "therefore"}({SpaceToken})?
 ):Therefore -->
:Therefore.ContextualRelation= {kind = "therefore", rule=JRuleTherefore}

phase: first
options: control = appelt
Rule:JRuleThereafter
(
{Token.lemma == "thereafter"}({SpaceToken})?
 ):Thereafter -->
:Thereafter.ContextualRelation= {kind = "thereafter", rule=JRuleThereafter}

phase: first
options: control = appelt
Rule:JRuleThen
(
{Token.lemma == "then"}({SpaceToken})?
 ):Then -->
:Then.ContextualRelation= {kind = "then", rule=JRuleThen}

phase: first
options: control = appelt
Rule:JRuleThenAgain
(
{Token.lemma == "then"}({SpaceToken})?{Token.lemma ==
"again"}({SpaceToken})?
 ):ThenAgain -->
:ThenAgain.ContextualRelation= {kind = "thenagain", rule=JRuleThenAgain}

phase: first
options: control = appelt
Rule:JRuleUntil
(
{Token.lemma == "until"}({SpaceToken})?
 ):Until -->
:Until.ContextualRelation= {kind = "until", rule=JRuleUntil}

phase: first
options: control = appelt
Rule:JRuleWhile
(
{Token.lemma == "while"}({SpaceToken})?
 ):While -->
:While.ContextualRelation= {kind = "while", rule=JRuleWhile}

phase: first
options: control = appelt
Rule:JRuleWhen
(
{Token.lemma == "when"}({SpaceToken})?
 ):When -->
:When.ContextualRelation= {kind = "when", rule=JRuleWhen}


phase: first
options: control = appelt
Rule:JRuleWhich
(
{Token.lemma == "which"}({SpaceToken})?
 ):Which -->
```

```
:Which.ContextualRelation= {kind = "which", rule=JRuleWhich}

phase: first
options: control = appelt
Rule:JRuleWhere
(
{Token.lemma == "where"}({SpaceToken})?
 ):Where -->
:Where.ContextualRelation= {kind = "where", rule=JRuleWhere}
```

Annexe 2 : Partie de l'ontologie des relations contextuelles

```
<?xml version="1.0"?>

<!DOCTYPE RDF [
   <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns">
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
   <!ENTITY rdfg "http://www.w3.org/2004/03/trix/rdfg-1">
   <!ENTITY es   "http://www.inria.fr/2008/03/24/estanda.rdfs">
   <!ENTITY rc   "http://www.inria.fr/2008/03/24/contextualRelation.rdfs">
]>

<rdf:RDF
   xml:base="&rc;/"
   xmlns:rdf="&rdf;#"
   xmlns:rdfs="&rdfs;#"
   xmlns:rdfg="&rdfg;#"
   xmlns:es="&es;#"
   xmlns:rc="&rc;/">

  <rdf:Property rdf:about="&rdfg;#TypeOfGraph">
      <rdfs:label>type of graph</rdfs:label>
      <rdfs:comment>
The graphs generated in title, paragraph, sentence or smallest textual
entity (ETp).
      </rdfs:comment>
      <rdfs:domain rdf:resource="&rdfg;#Graph"/>
      <rdfs:range rdf:resource="&rdfs;#Literal"/>
   </rdf:Property>


  <rdf:Property rdf:about="&rc;#contextualRelation">
      <rdfs:domain rdf:resource="&rdfg;#Graph"/>
      <rdfs:range rdf:resource="&rdfg;#Graph"/>
   </rdf:Property>

  <rdf:Property rdf:about="&rc;#temporalContextualRelation">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
   </rdf:Property>
  <rdf:Property rdf:about="&rc;#structuralContextualRelation">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
   </rdf:Property>
  <rdf:Property rdf:about="&rc;#discoursContextualRelation">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
   </rdf:Property>

              <!--         temporalContextualRelation           -->

  <rdf:Property rdf:about="&rc;#when">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>when</rdfs:label>
   </rdf:Property>
```

```
   <rdf:Property rdf:about="&rc;#since">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>since</rdfs:label>
   </rdf:Property>

   <rdf:Property rdf:about="&rc;#thenAgain">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>Then Again</rdfs:label>
   </rdf:Property>

   <rdf:Property rdf:about="&rc;#until">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>Until</rdfs:label>
   </rdf:Property>

   <rdf:Property rdf:about="&rc;#while">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>While</rdfs:label>
   </rdf:Property>

   <rdf:Property rdf:about="&rc;#asFarAs">
      <rdfs:subPropertyOf rdf:resource="#temporalContextualRelation"/>
      <rdfs:label>As far as</rdfs:label>
   </rdf:Property>

           <!--         ContextualRelation           -->


  <rdf:Property rdf:about="&rc;#comparedto">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
      <rdfs:label>compared to</rdfs:label>
   </rdf:Property>

  <rdf:Property rdf:about="&rc;#where">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
      <rdfs:label>where</rdfs:label>
   </rdf:Property>
  <rdf:Property rdf:about="&rc;#except">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
      <rdfs:label>except</rdfs:label>
   </rdf:Property>
  <rdf:Property rdf:about="&rc;#comparedto">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
      <rdfs:label>compared to</rdfs:label>
   </rdf:Property>
  <rdf:Property rdf:about="&rc;#comparedto">
      <rdfs:subPropertyOf rdf:resource="#contextualRelation"/>
      <rdfs:label>compared to</rdfs:label>
   </rdf:Property>


</rdf:RDF>
```

Résumé

Cette thèse entre dans le cadre du projet européen SevenPro (Environnement d'ingénierie virtuel sémantique pour la conception des produits) dont le but est d'améliorer le processus d'ingénierie de production dans les entreprises de fabrication, au moyen de l'acquisition, de la formalisation et de l'exploitation des connaissances. Nous proposons une approche méthodologique et logicielle pour générer des annotations sémantiques contextuelles à partir de texte. Notre approche est basée sur des ontologies et sur les technologies du Web sémantique.

Dans une première partie, nous proposons une modélisation de la notion de « contexte » pour le texte. Cette modélisation peut être perçue comme une projection des différents aspects du «contexte» abordés par ses définitions dans la littérature. Nous proposons également une modélisation des annotations sémantiques contextuelles, avec la définition des différents types de relations contextuelles pouvant exister dans le texte. Ensuite, nous proposons une méthodologie générique pour la génération d'annotations sémantiques contextuelles basées sur une ontologie du domaine qui exploite au mieux les connaissances contenues dans les textes.

L'originalité de la méthodologie est qu'elle utilise des techniques de traitement automatique de la langue ainsi que des grammaires d'extraction (engendrées automatiquement) de relations de domaine, de concepts et de valeurs de propriété afin de produire des annotations sémantiques reliées avec des relations contextuelles. De plus, nous prenons en compte le contexte d'apparition des annotations sémantiques pendant leur génération. Un système supportant cette méthodologie a été implémenté et évalué.

***Mots-clés* :** Web sémantique, TALN, annotations sémantiques, extraction d'information, annotations contextuelles, contexte, ontologie, RDF(S).

Abstract

This thesis falls within the framework of the European project SevenPro (Semantic Virtual Engineering Environment for Product Design) whose aim is to improve the engineering process of production in manufacturing companies, through acquisition, formalization and exploitation of knowledge. We propose a methodological approach and software for generating contextual semantic annotations from text. Our approach is based on ontologies and Semantic Web technologies.

In the first part, we propose a model of the concept of "context" for the text. This modeling can be seen as a projection of various aspects of "context" covered by the definitions in literature. We also propose a model of contextual semantic annotations, with the definition of different types of contextual relationships that may exist in the text. Then, we propose a generic methodology for the generation of contextual semantic annotations based on domain ontology that operates at best with the knowledge contained in texts.

The novelty in the methodology is that it uses language automatic processing techniques and grammar extraction (automatically generated) field relations, concepts and values of property in order to produce semantic annotations associated with contextual relations. In addition, we take into account the context of occurrence of semantic annotations for their generation. A system that supports this methodology has been implemented and evaluated.

***Keywords* :** semantic Web, NLP, semantic annotations, information extraction, contextual annotations, context, ontology, RDF(S).