Une architecture de services pour mieux spécialiser les processus d'acquisition terminologique

Farid Cerbah* — **Béatrice Daille****

** Dassault Aviation, DPR/ESA, 78, quai Marcel Dassault, 92552 Saint-Cloud cedex 300 — farid.cerbah@dassault-aviation.fr*

*** LINA - Université de Nantes, 2, rue de la Houssinière - BP 92208, 44322 Nantes cedex 03 — beatrice.daille@lina.univ-nantes.fr*

RÉSUMÉ. *L'acquisition terminologique est souvent considérée comme une technologie proche de la maturité dans les domaines du traitement automatique des langues et de l'ingénierie des connaissances. Elle peut jouer un rôle essentiel dans la constitution de lexiques spécialisés pour la recherche d'information et pour diverses applications du TAL. Il existe des outils robustes pour réaliser les tâches centrales de ce processus d'acquisition de ressources à partir de corpus. Cependant, mettre en place un assemblage cohérent d'outils pour supporter un processus aussi étendu reste une entreprise difficile, et les concepteurs ne peuvent guère prendre appui sur des architectures de référence. Le besoin de repères, tant méthodologiques que logiciels, est d'autant plus sensible que la nature même de l'acquisition terminologique requiert la mise en place de chaînes de traitement facilement reconfigurables. Pour répondre à ces besoins, nous proposons une architecture de services où chaque application est conçue par* réutilisation *de services génériques et* adaptation *de services dédiés à des tâches plus sensibles au contexte de l'application. Ce cadre structurant est mis en œuvre dans la plateforme d'acquisition HyperTerm, et nous montrons comment il a été exploité pour intégrer l'extracteur de termes* ACABIT.

ABSTRACT. *It is widely recognized that terminology acquisition is about to reach the stage of a mature technology. Robust tools have been developed to support these corpus-based acquisition processes. However, practitioners in this field cannot yet benefit from reference architectures that may greatly help to build large-scale applications. We propose a service oriented architecture that ease the development of applications through reuse of generic services and adaptation of domain-specific services. This architecture is implemented in the HyperTerm acquisition platform. We show how it has been used for coupling HyperTerm and Acabit term extractor.*

MOTS-CLÉS : *acquisition terminologique, architectures logicielles et TAL, SOA et web services.*

KEYWORDS: *terminology acquisition, software architecture for NLP, SOA and web services.*

1. Introduction

Dans le domaine du traitement automatique des langues (TAL), l'acquisition terminologique à partir de corpus (Bourigault *et al.*, 2001; Kageura *et al.*, 2004) est considérée comme l'une des technologies les plus matures. Durant ces dix dernières années, des méthodes et outils ont été élaborés pour offrir un support logiciel étendu aux processus bien identifiés que sont l'extraction et la structuration terminologique, la détection de variantes, ou encore l'alignement de termes. L'expérience montre que ces différentes techniques d'analyse de corpus spécialisés peuvent servir de nombreuses visées applicatives comme l'aide à la constitution d'ontologies, l'identification de descripteurs conceptuels pour la recherche d'information, ou encore l'assistance à la traduction technique. Il est toutefois surprenant de constater que tous ces traitements fortement automatisés n'ont que très rarement été envisagés sous l'angle de l'ingénierie logicielle. Contrairement à d'autres domaines du TAL, comme l'extraction d'information (*Gate* (Cunningham *et al.*, 1996), *Tipster* (Grishman, 1996)) et le dialogue homme-machine multimodal (*Miamm* (Reithinger *et al.*, 2003)), on compte encore peu de tentatives de définition d'architectures de référence. De tels points de repère pourraient sensiblement faciliter la mise en place d'applications en vraie grandeur. Ce travail est une première étape dans la définition et le développement d'une telle architecture. Nous montrons comment la technologie des *web services*[1] a été utilisée pour élaborer un cadre structurant permettant une mise en œuvre aisée d'applications d'acquisition terminologique par *réutilisation* de composants génériques et *adaptation* de composants dédiés à des traitements plus sensibles aux particularités des corpus à traiter.

Cette architecture est mise en œuvre dans l'outil d'acquisition HyperTerm et l'apport en adaptabilité qui en résulte peut être mis en exergue à travers un scénario typique d'utilisation. Supposons que le laboratoire *FRI* de l'université de Ljubljana souhaite réaliser une application d'analyse terminologique pour offrir un accès navigationnel centré sur la terminologie à des corpus médicaux en Slovène. HyperTerm implémente de manière robuste un certain nombre de services requis pour bâtir une telle application, tels que les prétraitements documentaires et textuels, les services d'indexation/navigation, et la gestion/validation des termes. Ils pourront donc être réutilisés dans la nouvelle chaîne de traitement. Mais, cette plateforme d'acquisition ayant été conçue pour traiter des corpus français et anglais, elle ne fournit aucun composant linguistique pour le Slovène. Pour cette nouvelle application, on peut toutefois tirer profit des efforts de l'institut Josef Stefan qui a développé l'étiqueteur *SLOP*, exploitant les ressources MULTEXT-East, et qui est déployé sous forme de web service sur

1. Derrière ce choix terminologique ambigu et réducteur se cache une technologie mature, issue de la mouvance *XML*, dédiée au développement d'applications distribuées. Outre le recours au formalisme *XML-Schema* pour encoder les objets structurés à échanger, l'approche web service propose un certain nombre de formalismes et protocoles pour encoder les messages (*SOAP*), décrire les services et leurs interfaces (*WSDL*), publier des services (*UDDI*), et bâtir des chaînes de traitement complexes par composition de services (*WS-BPEL*). Le triplet *SOAP/WSDL/UDDI* constitue le socle de cette technologie (cf. § 2.2).

le site de l'institut. Au lieu d'invoquer son étiqueteur interne, HyperTerm sera configuré de manière à invoquer *SLOP* dynamiquement et à distance, lors de l'exécution du processus d'acquisition. En outre, avant d'exploiter les résultats de l'étiquetage, un « guesser » lexical dédié au domaine médical sera sollicité pour réduire le nombre de mots inconnus. Comme aucun extracteur réutilisable n'est disponible, un extracteur spécifique sera implémenté. Le guesser et l'extracteur prendront aussi la forme de web services, mais, grâce à l'infrastructure d'intégration fournie par cette architecture générique, aucun développement lié à la communication distribuée n'est requis. C'est là un scénario fictif, mais qui n'en est pas moins réaliste compte tenu de la maturité des technologies mises à contribution.

Comme l'illustre ce scénario, la mise en œuvre d'une application complète implique souvent une articulation complexe de traitements où des tâches proprement terminologiques très sensibles au type de corpus sont encadrées par des tâches complémentaires, d'ordre linguistique et documentaire, qui correspondent quant à elles à des fonctionnalités plus stables d'une application à l'autre. C'est dans une telle conception étendue de l'acquisition terminologique que les problèmes d'architecture et d'intégration des traitements prennent toute leur importance. Pour faciliter la mise en œuvre de telles applications, il apparaît donc essentiel d'élaborer un support logiciel adaptable offrant la possibilité de spécialiser certains traitements tout en préservant une forte capacité de réutilisation de traitements génériques.

La suite de l'article est organisée de la façon suivante. En section 2, nous passons en revue les propositions les plus significatives en matière d'infrastructures de développement et d'intégration en TAL et nous examinons dans ce contexte l'apport potentiel des concepts d'architecture de services. Nous introduisons en section 3 la plateforme HyperTerm que nous avons restructurée pour valider l'architecture de référence proposée dans cette article. La section 4 est consacrée à la problématique d'adaptabilité des processus d'acquisition terminologique. Puis, nous exposons en section 5 notre proposition d'architecture et, en section 6, le couplage HyperTerm-ACABIT que nous a permis de réaliser cette nouvelle infrastructure d'intégration. Nous concluons avec quelques perspectives.

2. Architectures logicielles pour le TAL

Les exigences de réutilisabilité et d'adaptabilité, longtemps restreintes en TAL aux seules ressources linguistiques, ont fini par concerner aussi – et avec autant d'acuité – les composants de traitement (Cunningham, 2000). En effet, on a vu se multiplier ces dix dernières années les initiatives de développement d'applications en vraie grandeur avec une forte tendance à la réutilisation de composants massifs préexistants, souvent conçus à d'autres fins. La facilité d'intégration des composants externes a une incidence forte sur les coûts de développement de ces systèmes complexes. Il n'est dès lors pas surprenant de percevoir dans les modèles d'architecture conçus pour le TAL la mise en œuvre de principes de conception proches de ceux qui sous-tendent les architectures orientées services. On retrouve ainsi, de manière plus ou moins explicite,

la volonté d'évoluer vers des méthodologies et des infrastructures pour le développement de systèmes conçus comme des assemblages de composants, éventuellement hétérogènes et distribués, et implémentant des interfaces standardisées.

2.1. *Synthèse des infrastructures existantes*

Les initiatives *Tipster* (Grishman, 1996), *Gate* (Cunningham *et al.*, 1996; Cunningham *et al.*, 2002) et *UIMA* (Ferrucci et Lally, 2004) comptent parmi les propositions les plus significatives. Il convient de souligner que même si ces différents travaux affichent une vocation généraliste, ils ont toujours une application type en ligne de mire. Il s'agit dans la majorité des cas de faciliter le développement d'applications en extraction d'information.

Certaines caractéristiques nous paraissent bien indiquées pour appréhender les apports des infrastructures proposées. Nous citerons sans être exhaustif les caractéristiques suivantes : la richesse et la facilité d'utilisation des procédés d'intégration de composants en contexte distribué et hétérogène, le niveau d'interopérabilité et de standardisation des interfaces, la complexité des « objets » échangés par les composants, ou encore la richesse des modèles et bibliothèques de composants mis à disposition dans l'infrastructure.

Tipster est à retenir comme la première initiative d'ampleur visant à standardiser les interfaces (API) des composants « métiers » de l'extraction d'information et des procédés d'enrichissement de données textuelles par annotations. Dans le modèle *Tipster*, un document textuel traverse une chaîne de traitement où chaque maillon inscrit de nouvelles annotations dans une base de données centralisée accessible par tous les maillons de la chaîne. *Tipster* reste une référence importante dans la mesure où ses spécifications ont été adoptées (ou adaptées) dans plusieurs systèmes dont *Gate* et *Corelli* (Zajac *et al.*, 1997).

Gate et *UIMA* sont sans conteste les propositions les plus abouties, avec la réalisation d'infrastructures logicielles matures et largement déployées. Dans l'infrastructure *Gate*, toute application possède une ossature développée en Java articulant des composants inscrits dans un canevas dérivé du modèle des *java beans*. *Gate* gère de manière uniforme trois types de composants ou *ressources* : les ressources linguistiques, les composants de traitement et les composants de visualisation. Ces ressources peuvent être manipulées interactivement pour assembler des chaînes de traitement et la transmission d'informations entre composants est effectuée par le biais d'annotations centralisées dans une base commune conformément au modèle *Tipster*. De plus, la plateforme est dotée d'une bibliothèque de composants réutilisables.

De par ses principes, *UIMA* est très proche de *Gate* et s'inspire sans en être une implémentation du modèle d'annotation de *Tipster*. On notera parmi les principaux apports d'*UIMA* une plus grande richesse dans la définition de la logique d'enchaînement des traitements, avec notamment des possibilités de parallélisation et de recouvrement d'erreurs.

En matière de réutilisation et d'intégration de composants *Gate* et *UIMA* offrent des cadres facilitateurs cohérents qui se démarquent clairement des solutions artisanales communément adoptées dans le domaine du TAL. Toutefois, le choix de langages de développement comme socle des procédés d'intégration peut s'avérer inadéquat lorsque l'intégrateur doit faire face à une forte hétérogénéité de langages et de plateformes d'exécution, souvent doublée d'une distribution des ressources et des traitements. Une telle complexité structurelle est aujourd'hui inhérente aux systèmes d'information et, dans ce contexte, les applications de TAL n'échappent pas à la règle. Notons que ces deux plateformes de référence intègrent déjà un support minimal des web services leur permettant de dépasser en partie ces limites.

Les infrastructures *Corelli* (Zajac *et al.*, 1997) et *Trevi* (Basili *et al.*, 1999) présentent la particularité de fonder leurs procédés d'intégration de composants sur le standard de bus à objets *Corba*. Si cela constitue indéniablement une avancée sur le plan de l'interopérabilité, la complexité de mise en œuvre des mécanismes d'intégration à base de bus à objets ne favorise pas l'adoption de telles infrastructures. Par ailleurs, il semble qu'en dépit de leur richesse conceptuelle, ces deux initiatives n'aient pas bénéficié du même effort de développement et de diffusion que *Gate* et *UIMA*.

Parmi les développements plus récents, il convient de citer la plateforme *LinguaStream* (Widlöcher et Bilhaut, 2005) qui permet de spécifier des chaînes de traitement complexes en procédant aussi par annotations cumulatives de documents. L'un des apports majeurs de cette plateforme est le recours à un langage de contraintes pour supporter une description unifiée et déclarative des traitements linguistiques (Widlöcher, 2006).

2.2. *L'apport des web services*

En TAL, les formalismes de représentation de données et de méta-données recommandés par le consortium W3C (XML, XML Schema, RDF/S et OWL) ont été massivement adoptés par les chercheurs et praticiens de ce domaine. Ils sont intimement associés aux problématiques d'échange et de standardisation de ressources linguistiques (Ide et Romary, 2001). Mais, outre ces aspects représentationnels, les recommandations et outils orientés traitement issus de cette technologie – dont les spécifications XSLT, SAX/DOM et leurs implémentations – sont très rapidement devenus des références tout aussi incontournables. XML est également communément utilisé pour enchaîner les traitements linguistiques. La boîte à outils LT TTT (Mikheev *et al.*, 1999) est représentative de cette pratique courante qui confère accessoirement à XML le rôle de langage pivot pour l'intégration de composants.

C'est là une des idées fondatrices des web services que d'utiliser XML, et plus précisément XML Schema, pour échanger des données structurées entre composants dans des applications hétérogènes et distribuées. Cette approche standard de l'intégration de composants a suscité une large adhésion (la relative simplicité du modèle initial ainsi que son caractère consensuel ont beaucoup contribué à son acceptation).

De plus, les web services garantissent un haut niveau d'interopérabilité tout en permettant à des composants distants d'échanger des objets relativement complexes, et les infrastructures associées prennent totalement en charge les opérations d'encodage et de décodage requises pour transporter ces objets à travers les réseaux.

Sous l'impulsion d'organismes de standardisation tels que le W3C, OASIS et WS-I, le socle technologique des web services s'est construit par élaboration progressive et cumulative de recommandations techniques fondées sur le formalisme XML, pour couvrir les différentes facettes de la communication inter-logicielle, de la formalisation des messages échangés entre services distants (SOAP), à la composition de services en passant par la description d'interfaces (WSDL) et la publication de services (UDDI)[2].

Ce socle constitue la principale concrétisation du concept d'*architectures de services* (SOA[3]) qui est en passe de s'imposer comme une référence incontournable en matière de développement de grands systèmes en environnement fortement distribué. Parmi les caractéristiques saillantes de ce modèle, citons :

– une standardisation systématique des protocoles de communication et des interfaces de services pour garantir un haut niveau d'interopérabilité ;

– une méthodologie de construction de systèmes complexes par couplage faible de composants distribués aux frontières clairement définies ;

– la définition d'un cadre non-propriétaire d'intégration d'applications, potentiellement supporté par plusieurs infrastructures logicielles.

Ces caractéristiques nous laissent penser que ce modèle devrait faire de plus en plus d'adeptes parmi les architectes de systèmes de TAL, même si on compte encore peu de réalisations en TAL exploitant la technologie des web services (Dalli *et al.*, 2004; Biemann *et al.*, 2004). Notons que des plateformes orientées composants telles que *Gate* et *UIMA* évoluent progressivement vers le modèle SOA.

L'essor de ce nouveau concept d'architecture distribuée pourrait faire émerger une nouvelle forme de partage des ressources et services linguistiques. En effet, une publication systématique de services selon ce modèle rendrait possible le développement d'applications complexes où l'exploitation de services externes se traduirait par l'invocation distante de versions de référence hébergées par les concepteurs, et non par l'appropriation de copies locales. Des gains considérables pourraient en résulter, en termes de mutualisation des ressources, de facilité de développement et de maintenance.

2. SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration).
3. SOA (Service Oriented Architecture).

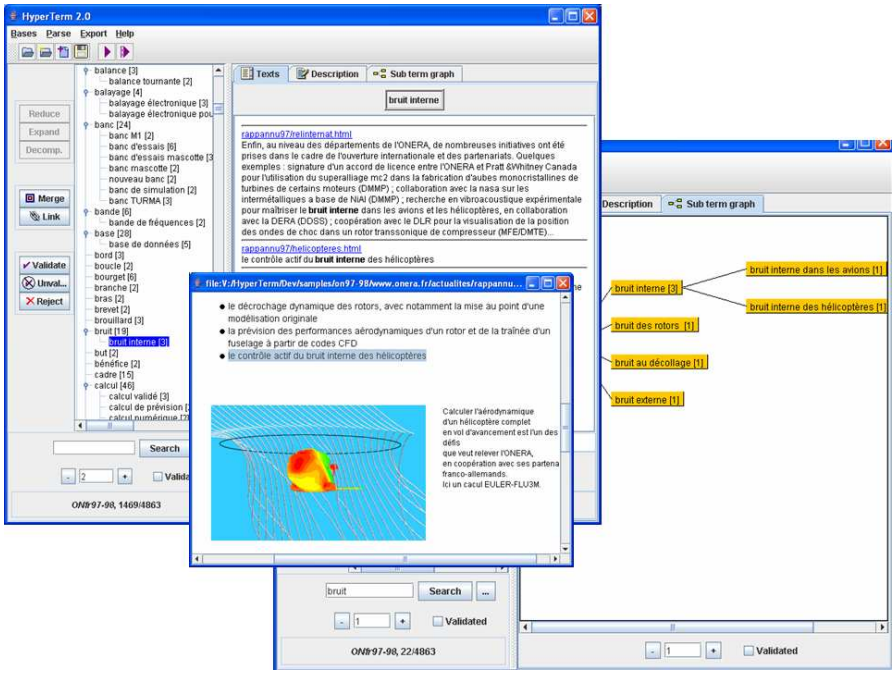

Figure 1. *Vues de la plateforme d'acquisition terminologique HyperTerm*

3. La plateforme HyperTerm

Nous décrivons ici la plateforme HyperTerm qui nous a servi de base d'expérimentation pour valider l'architecture de services (cf. § 5). Dans cette partie, nous commençons par donner une vue d'ensemble de la plateforme et de sa structure initiale. Nous décrivons ensuite plus en détail le composant d'extraction terminologique.

3.1. *Description générale de la plateforme*

La plateforme HyperTerm permet non seulement de constituer et gérer des terminologies métiers, mais aussi d'exploiter les réseaux de termes produits pour explorer finement, par navigation guidée, les corpus de documents analysés (cf. figure 1). Ce support logiciel couvre l'intégralité du processus, de la constitution de corpus de référence à la validation interactive des termes, en passant par les différentes phases d'analyse et d'extraction.

Le noyau de l'application assure la gestion et le stockage des larges collections de données relationnelles manipulées dans ce processus étendu. Il supervise l'exécu-

tion et l'enchaînement des différentes phases de traitement. Autour de ce noyau, sont articulés les différents composants dédiés aux principales étapes du processus :

– Le gestionnaire de documents

Ce composant assure les fonctions de constitution, de prétraitement et d'indexation de corpus hétérogènes. Les prétraitements documentaires et paratextuels visent à s'affranchir des différents formats pour mettre les documents sources, éventuellement répartis sur différents sites, sous une forme qui les rend directement exploitables par les traitements linguistiques.

Les services d'indexation ont pour rôle d'assurer un repérage précis en corpus des unités produites lors des différentes phases d'analyse et, en particulier, des termes extraits. Ils constituent le socle des mécanismes de navigation hypertextuelle décrits ci-après.

– L'étiqueteur grammatical

Les fonctions d'analyse morpho-syntaxique sont assurées par l'étiqueteur MultAna. Ce composant associe un module de désambiguïsation à base de règles à l'analyseur morphologique MMORPH (Petitpierre et Russell, 1995). Dans le processus d'étiquetage, MMORPH attribue à chaque mot de la séquence à traiter toutes les analyses morphologiques envisageables. L'application des règles de désambiguïsation a pour but de restreindre cet ensemble de solutions potentielles pour associer à chaque mot l'analyse morphologique la plus plausible. MultAna offre un certain de nombre de fonctionnalités qui s'avèrent particulièrement utiles dans un contexte de manipulation de ressources terminologiques, comme la possibilité d'exploiter plusieurs lexiques terminologiques hiérarchisés, des mécanismes robustes d'identification de mots composés, et l'attribution de catégories aux mots inconnus (très fréquents dans les textes à forte technicité).

– L'extracteur de termes

C'est le composant central de la plateforme qui a pour fonction l'identification des unités lexicales susceptibles de représenter des termes du domaine. Ce composant, décrit en détail ci-après (§ 3.2), produit par exploration des textes étiquetés un réseau de candidats termes, dont une grande partie correspond à des unités lexicales composées.

– L'explorateur

Dans l'activité de constitution de ressources terminologiques, la part de travail manuel peut être colossale compte tenu des volumes de données générées par les phases automatisées d'extraction. Il est dès lors essentiel d'attacher une attention particulière aux procédés de manipulation interactive des termes et des données associées. L'explorateur d'HyperTerm offre de multiples fonctions interactives pour faciliter la mise en œuvre des opérations de filtrage et de gestion des termes requises pour la finalisation d'une base terminologique. De plus, des mécanismes de navigation permettent une remise en contexte documentaire immédiate des termes extraits.

Il faut souligner que les services avancés d'indexation et de navigation centrées sur la terminologie font d'HyperTerm une application en soi. En effet, du point de vue de la recherche d'information, HyperTerm peut être vu comme un *navigateur conceptuel*

permettant d'explorer finement des fonds documentaires techniques et de personnaliser des réseaux hypertextuels. Sur cet aspect, il présente des points communs avec le système *IndDoc* (Nazarenko et Aït El Mekki, 2005) qui exploite des techniques d'acquisition terminologique avec validation interactive pour élaborer des index de fin de livre.

On constate que cette plateforme d'acquisition est construite selon une architecture modulaire. Cependant, les composants étant assemblés par *couplage fort* pour former un applicatif monolithique, elle offre en définitive peu de flexibilité aux utilisateurs qui souhaiteraient étendre ou modifier ses fonctionnalités. L'architecture n'a pas été pensé pour faciliter l'adaptation de certains composants par des utilisateurs experts (autres que les concepteurs de la plateforme). C'est l'une des lacunes que la nouvelle architecture est censée combler.

3.2. *Le composant d'extraction*

Les traitements terminologiques sont évidemment centraux dans le processus d'acquisition supporté par cet outil, et les méthodes mises en œuvre s'inscrivent dans un champ disciplinaire bien établi du TAL. En effet, déterminer quels sont les termes d'un domaine à partir de documents textuels a été un champ de recherche très actif à partir des années 90 (Bourigault et Jacquemin, 2000) qui a conduit quelques années plus tard à la production d'un certain nombre de méthodes et d'outils (cf. (Cabré *et al.*, 2001) pour une étude comparative d'un échantillon des réalisations les plus représentatives). Les différentes méthodes peuvent impliquer, en entrée, sur les textes à traiter des traitements linguistiques plus ou moins sophistiqués, de la segmentation à l'analyse syntaxique. Toutes les approches se rejoignent dans le fait de proposer des procédés d'identification de syntagmes – majoritairement composés – susceptibles de représenter des termes pertinents de par leurs propriétés lexico-syntaxiques. Elles se différencient néanmoins sur un certain nombre de caractéristiques. Nous retiendrons ici celles qui nous semblent les plus significatives :

– Le degré de complexité de l'unité terminologique admis, généralement un groupe nominal : nom seul, groupe nominal de taille limitée, groupe nominal maximal. Ainsi, pour la séquence *diffraction des neutrons sur poudre à basse température*, certains systèmes ne retiendront que le termes simple *diffraction*, d'autres le terme complexe *diffraction de neutron*, d'autres encore la séquence entière.

– La prise en compte ou non de la variabilité de la forme linguistique des termes complexes et le degré de variabilité admis : orthographique, morphologique, syntaxique ou sémantique (Daille, 2005; Jacquemin, 2001; Hamon et Nazarenko, 2001). Par exemple, selon les systèmes, les occurrences *acidité du sang* et *acidité sanguine* seront considérées soit comme termes variants, soit comme termes différents.

– Le filtrage automatique mis en place pour réduire le nombre d'unités lexicales extraites et ne retenir que celles à haute « valeur terminologique » : filtrage statistique incorporé à la méthode d'identification pouvant avoir lieu avant ou après l'identifica-

tion des groupes nominaux, filtrage utilisant des ressources lexicales terminologiques existantes et des listes d'anti-termes, éventuellement extraites automatiquement de corpus généraux (Drouin, 2003).

– L'organisation de ces listes de candidats-termes par l'ajout de relations relevant de la sémantique lexicale ou le regroupement par classes (Nazarenko et Hamon, 2002; L'Homme, 2004; Cerbah, 2000).

Le composant d'extraction intégré dans HyperTerm implémente une méthode de repérage de termes fondée sur des patrons morpho-syntaxiques définissant une grammaire locale des termes composés. À des fins d'efficacité, ces patrons sont encodés sous la forme d'automates à états finis appliqués sur des séquences de mots étiquetés. Il est possible de situer plus précisément la méthode d'extraction implémentée en regard des quatre caractéristiques définitoires mentionnées ci-avant :

– La grammaire d'extraction est conçue pour rendre compte des termes longs, mais la longueur maximale autorisée est l'un des paramètres d'entrée de l'extraction. L'extension de la grammaire est contrainte par ce paramètre qui peut varier selon les besoins de l'application.

– La méthode détecte les variations flexionnelles et les variations par omission de mots grammaticaux. Par exemple, les trois candidats termes *message du calculateur*, *message calculateur*, et *message en provenance du calculateur* seront automatiquement réunis sous une même entrée. En revanche, la méthode n'est pas dimensionnée pour détecter des variations fondées sur des transformations complexes de niveau morphologique ou syntaxique. Cela constitue l'une des limites majeures du composant d'extraction d'HyperTerm en comparaison à un système comme ACABIT (Daille, 2005) qui intègre un traitement plus précis des phénomènes de variation lui permettant d'associer des termes comme *arbre forestier*, *arbre en forêt*, *arbre semi-forestier*, et *arbre fruitier ou forestier*.

– Des procédés de filtrage sont utilisés pour réduire le nombre de candidats. Outre les filtrages classiques basés sur les effectifs et sur la longueur des termes, nous exploitons un procédé d'*extension lexicale maximale* apparenté à certaines techniques d'identification de segments répétés (Salem, 1997) et qui permet d'atteindre des réductions très significatives sur certains corpus. Le principe consiste à éliminer tout candidat terme englobé par un autre candidat de même base et de même effectif. Par exemple, si les termes extraits *augmentation du nombre*, *augmentation du nombre d'avions*, et *augmentation du nombre d'avions livrés* ont les mêmes effectifs, seul le dernier sera maintenu. L'application de ce procédé est conditionné par un effectif seuil qui constitue un paramètre d'entrée (le procédé n'est appliqué qu'aux termes d'effectifs supérieurs à la valeur seuil).

– Enfin, une structure hiérarchique est dérivée des termes extraits. Des relations de spécialisation entre termes sont identifiées sur la base de propriétés de recouvrement lexical. Par exemple, dans une hiérarchie de termes calculée à partir d'un corpus aéronautique, le terme *plan de vol* aurait les unités *plan de vol de référence* et *plan de vol modifié en cours de décollage* pour termes fils. Cette structuration n'intègre pas les relations sémantiques complexes dépendantes d'un domaine induites par les varia-

tions morphologiques et syntaxiques et traitées par ACABIT comme, par exemple, celle reliant un processus avec son résultat : *filetage du saumon/saumon fileté*, ni les relations d'antonymie : *levure floculante/levure non floculante* (Daille, 2003). Elle n'exploite pas non plus les relations sémantiques induites par des marqueurs contextuels (Condamines, 2002; Aussenac-Gilles et Séguéla, 2000) comme par exemple le patron lexico-syntaxique « *X est un composant de Y* » induisant une relation de méronymie entre X et Y.

4. Le besoin en adaptabilité

On peut voir dans HyperTerm une réponse possible au besoin, maintes fois exprimé, de doter le processus étendu d'acquisition terminologique d'un support logiciel finalisé. Cependant, la mise à l'épreuve d'un tel système dans différents domaines techniques fait rapidement apparaître les limites d'un support générique unique. En particulier, HyperTerm a été mis au point et rodé sur des corpus du domaine aéronautique et les procédés d'extraction ont été optimisés pour ce type de corpus. On observe que les expériences menées hors de ce domaine privilégié aboutissent à des résultats moins concluants.

Certaines limites de ce support logiciel peuvent certes s'expliquer par l'incomplétude de la méthode générique d'extraction implémentée. Nous avons souligné précédemment certaines faiblesses de cette méthode, comme l'absence d'un traitement étendue des phénomènes de variation transverses à différents domaines. Nous montrerons comment l'intégration de l'extracteur ACABIT permet de combler en partie ces limites. Mais, force est de constater que même si les méthodes d'acquisition terminologique peuvent atteindre un niveau de généricité appréciable, elles nécessitent d'être adaptées ou de faire appel à des traitements spécifiques pour gérer des particularités liées au domaine traité. Cet argument peut être appuyé par quelques exemples dans des domaines riches en néologismes.

– **Agro-alimentaire** : Les composés latins comme *Geotrichum candidum* sont très nombreux dans les textes relevant de l'agriculture ou de l'agro-alimentaire. Les mots latins employés sont analysés comme des mots inconnus auxquels aucune catégorie grammaticale n'est associée. Ces composés ne sont généralement pas détectés par les moteurs d'extraction terminologique.

– **Biologie** : Les entités nommées utilisées dans le domaine de la biologie moléculaire regroupent les noms de gènes, de bactéries et leurs zones d'activités comme les noms de cellules ou d'organismes (Kim *et al.*, 2004). La reconnaissance d'un nom de gène comme *IL-2 receptor alpha* ou d'une zone comme *CD4-CD8-murine T lymphocyte precursors* nécessite des programmes de reconnaissance et de catégorisation au même titre que les entités nommées relevant du domaine général comme les noms propres de personnes ou de lieux. Ces entités subissent les mêmes variations que les termes complexes mais font intervenir d'autres règles de formation : *T- and B-lymphocyte* est une variante de coordination de *T-lymphocyte* et *B-lymphocyte* obtenue à l'aide d'une règle relevant à la fois de la morphologie et de la syntaxe.

– **Chimie** : Les textes relevant du domaine de la chimie contiennent de nombreux composés issus de la nomenclature des corpus chimiques comme *6-di-0-isopropylidène-alpha-D-allofuranose-3spiro-3* (Tartier, 2004). Ces composés posent problème aux programmes informatiques du fait de leur longueur et leur découpage est requis. Celui-ci est le plus souvent arbitraire alors qu'une décomposition guidée par une analyse des composants chimiques serait plus appropriée.

– **Médical** : Dans le domaine médical, les composés savants contiennent dans leur grande majorité des racines gréco-latines (*gastr-*, *-phage*, *-hydr-*). Une racine partage sa catégorie et son sens avec l'entrée lexicale contemporaine auquel elle supplée (ainsi, *gastr-* signifie *estomac*) (Namer, 2005). Un traitement morphologique particulier comme celui proposé par l'analyseur DériF (Namer et Zweigenbaum, 2004) permet de regrouper des variantes synonymiques de termes comme *hépatique* et *foie*.

Devant cette diversité de phénomènes, on comprend aisément le besoin d'évoluer vers un support logiciel plus flexible offrant la possibilité de spécialiser le cœur du processus d'acquisition terminologique.

5. Une architecture de référence pour l'acquisition terminologique

Nous proposons une infrastructure de services à contrôle centralisé qui nous paraît bien adaptée à la mise en œuvre de processus d'acquisition d'informations à partir de corpus. Une application construite sur cette architecture est définie structurellement comme un assemblage de services locaux ou distants dont l'invocation est orchestrée par un superviseur.
Cette infrastructure a permis de réarchitecturer la plateforme HyperTerm, et les apports en termes de réutilisabilité et d'adaptabilité ont été validés par le développement de plusieurs applications.
Dans ce cadre, sont mis à disposition un certain nombre de moyens facilitant la composition de nouvelles chaînes de traitement, en particulier : (a) une typologie de services où chaque type de services adaptables est accompagné d'une spécification d'interface et de classes prédéfinies sur lesquelles le développeur peut prendre appui pour dériver de nouveaux services (§ 5.1), et (b) un modèle de données stratifié qui facilite l'inclusion de nouveaux algorithmes d'extraction (§ 5.2).

5.1. *Typologie des services*

Nous adoptons une typologie de services qui peut être vue comme une extension du modèle de composants de *Gate*. Au premier niveau de la typologie, nous distinguons trois grands types de services :

– Les services fixes

Il s'agit des services qui assurent des fonctionnalités stables d'une application à l'autre. On compte dans cette catégorie les prétraitements documentaires et

para-textuels, les mécanismes d'indexation ainsi que les services de stockage. Pour couvrir ces besoins génériques, il est concevable d'inclure dans l'infrastructure des implémentations robustes réutilisables des services correspondants (i.e. une seule instance pour chaque service type).

– Les services instanciables

Ces services correspondent à des phases de traitement identifiées comme sensibles au contexte de l'application et qui doivent dès lors pouvoir être adaptées aisément. Dans le cadre de l'acquisition terminologique (§ 4), les services centraux du processus, dédiés à l'extraction, la structuration et la détection de variantes, sont à inscrire en priorité dans cette catégorie[4]. Chaque type de services instanciables est représenté par une interface décrite en WSDL et peut être instancié par un service interne, intégré dans la plateforme, ou par un service externe déployé sur une infrastructure de web services. Le tableau 1 donne une description simplifiée de quatre services instanciables. Par exemple, l'interface du service-type `Extractor` est composé de trois méthodes : `initExtractor`, appelée une fois par le superviseur au début de la phase d'analyse, `extract` qui est invoquée autant de fois que le corpus compte de documents et enfin `getExtractedData` appelée en fin d'extraction pour récupérer les termes identifiés. Toute instance du service-type doit fournir une implémentation de ces méthodes.

Extractor	Tagger	Adapter	Refiner
`init(nbDocs)`	`init(lang, params)`	`process(docUrl)`	`process(docUrl, corpusUrls)`
`extract(taggedDocUrl)`	`tag(textDocUrl)`		
`getExtractedData()`			

Tableau 1. *Interfaces de services instanciables (ces interfaces sont décrites en WSDL)*

La plupart des services instanciables sont associés à des phases de traitement bien identifiées et récurrentes dans la classe d'applications visée. Ils répondent à des spécifications fonctionnelles explicitées dans des interfaces précises. En complément de ces services sémantiquement bien cadrés, nous distinguons deux autres types de services instanciables d'interfaces plus génériques et qui se prêtent à des usages plus diversifiés :

– Les adaptateurs de données : L'assemblage d'une chaîne de traitement requiert souvent des opérations d'adaptation des données échangées par les services mis bout à bout, en particulier lorsque la chaîne comporte des services « recyclés ». Pour assurer une interopérabilité syntaxique et sémantique, des adaptateurs doivent être insérés entre les services adjacents dont les interfaces

4. Comme nous l'a fait remarqué l'un des relecteurs, le besoin d'adaptation peut, dans certains contextes, concerner des services que nous considérons ici comme fixes, tels que les services documentaires. Une telle typologie « discrète » des services est inévitablement approximative. Elle rend néanmoins compte du fait que certaines phases de traitement font souvent – voire systématiquement – l'objet d'adaptation d'une application à l'autre, alors que d'autres ne le sont que dans des contextes très marginaux.

reposent sur des modèles de données distincts. La résolution des différences peut se limiter à une simple conversion lexicale entre deux ensembles sémantiquement équivalents d'étiquettes formelles (par exemple, convertir les structures de traits d'un analyseur morphologique pour les rendre interprétables par un service de désambiguïsation)[5].

– Les affineurs : Ces services permettent d'insérer des traitements complémentaires pour finaliser une chaîne d'acquisition. Cette fonctionnalité apporte une certaine flexibilité à l'infrastructure, évitant en particulier de restreindre son champ d'application à un processus d'acquisition prédéfini et figé dans sa composition. Pour l'acquisition terminologique, il est possible de greffer sur un processus classique des traitements complémentaires pour enrichir une base de termes (par exemple, pour ajouter des relations lexico-sémantiques).

D'un point de vue plus technique, Les services instanciables dédiés aux procédés d'adaptation et d'affinement suivent des modèles d'interface proches, où le traitement à réaliser est matérialisé par un processus de transformation de documents XML. Ainsi, une instance de type `Adapter` (cf. tableau 1) fournit une implémentation de la méthode `process` qui prend en argument un document XML encodant les données à adapter, et produit un document XML contenant le résultat de la conversion.

– Les services de visualisation

Le caractère partiel des résultats produits par les systèmes actuels de TAL nécessite de mettre à disposition des moyens interactifs efficaces pour supporter le travail manuel de finalisation des résultats. Par exemple, en traduction assistée, pour faciliter l'inévitable travail de correction, une bonne aide logicielle doit fournir au minimum, comme assistance interactive, une visualisation alignée des textes sources et cibles, de manière à permettre un repérage et une reformulation rapides des fragments de textes mal traduits. Comme évoqué en § 3, un outil finalisé d'acquisition terminologique ne peut se concevoir sans procédés performants de filtrage et de validation des termes, et sans une aide à la navigation permettant une exploration souple des documents sources et une remise en contexte des nombreux termes extraits. Les outils interactifs déjà présents dans HyperTerm garantissent de fait à cette nouvelle infrastructure une certaine richesse en matière de services de visualisation. Il reste que, dans l'état actuel de nos développements, ces services présentent le défaut de s'apparenter à des services fixes. Sur cet aspect, *Gate* offre des possibilités d'adaptation des interfaces, même si les services proposés n'ont pas la richesse fonctionnelle de ceux intégrés dans HyperTerm.

Outre ces services de visualisation pour l'utilisateur final, il est utile de prévoir des outils interactifs orientés développeur. *Gate* et *LinguaStream* se distinguent sur ce plan par la mise à disposition de moyens spécifiques pour la visualisation et la composition interactive des chaînes de traitement, là où nous nous contentons d'exploiter les outils

5. La divergence à surmonter peut être plus profonde comme mettre en correspondance des représentations fondées sur des paradigmes linguistiques distincts mais néanmoins compatibles (par exemple, transformer un arbre syntaxique à base de constituants en arbre de dépendance).

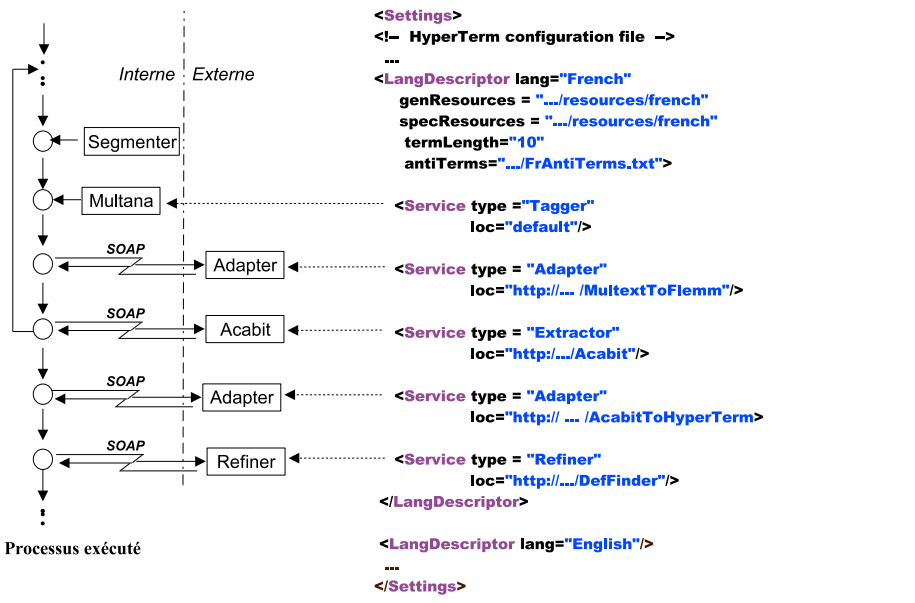

Figure 2. *Une chaîne de traitement construite autour d'*ACABIT*, mêlant des services internes à HyperTerm et des services externes (dont* ACABIT*)*

de mise au point fournis dans les infrastructures génériques de services, notamment par Apache-Axis[6].

Le principe de ce modèle de services et son usage peuvent être récapitulés en quelques mots : concevoir une application dans cette infrastructure consiste à saturer un processus-type composé de services fixes et d'un nombre arbitraire de services instanciables (dont un étiqueteur et un extracteur). Ce processus est conçu pour s'appliquer sur un corpus réparti et la saturation du processus est obtenue en attribuant une implémentation interne ou externe à chacun des services instanciables.

La figure 2 donne un exemple concret de définition d'une application selon ce modèle. Seules les descriptions des services instanciables sont spécifiées dans le fichier de configuration définissant l'application. Nous reviendrons plus longuement sur cette application en § 6.

6. http ://ws.apache.org/axis/

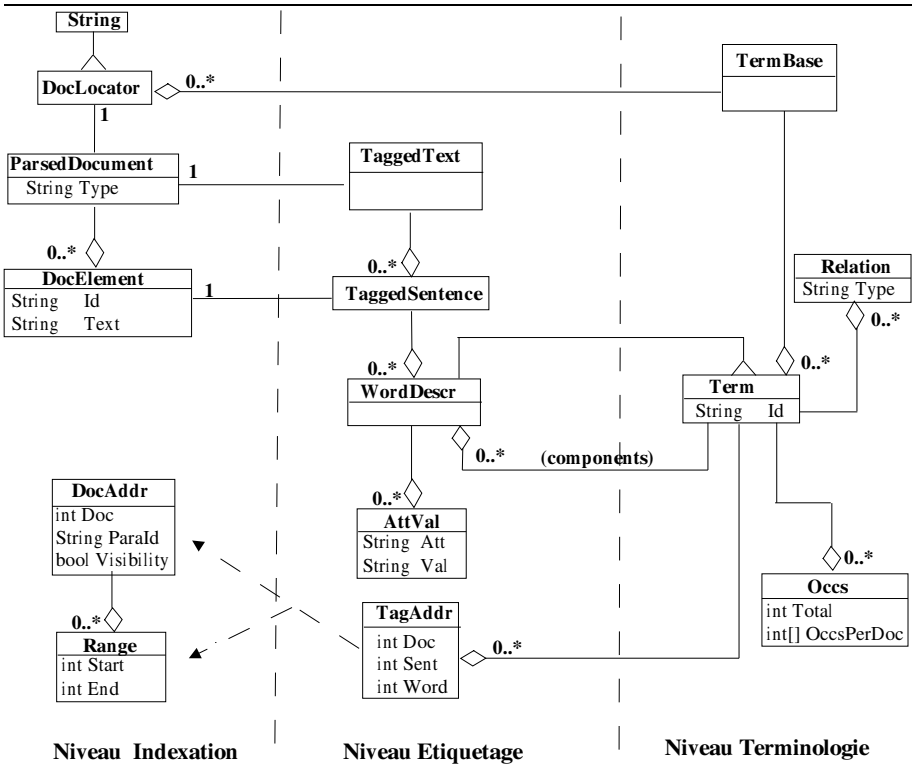

Figure 3. *Le modèle de données de l'infrastructure d'intégration. La mise en correspondance entre les niveaux du modèle est assurée par cette infrastructure*

5.2. *Modèle de données et support logiciel*

L'objectif de ce travail ne se limite pas à proposer un cadre théorique pour la définition d'applications composites. Sur le plan pratique, nous cherchons aussi à minimiser l'effort de développement et d'intégration des services externes. Deux aspects de l'infrastructure y contribuent :

– un modèle de données stratifié pour supporter une conception modulaire des services ;

– un ensemble de composants prédéfinis dans les langages les plus utilisés en TAL pour faciliter le développement de nouveaux services.

Lors de la définition d'un nouveau service, l'utilisateur de l'infrastructure ne doit avoir à manipuler que des entités sémantiquement pertinentes pour l'étape de traitement couverte par le service. C'est ce que permet le modèle de données schématisé en figure 3, dont les entités sont réparties sur trois niveaux. Ainsi, avec ce modèle stra-

tifié[7], la logique d'un extracteur s'exprimera en des termes relevant des niveaux *terminologie* et *étiquetage*. Cependant, pour les besoins de l'indexation, il est nécessaire d'associer à chaque nouvelle occurrence de termes des informations pour la localiser avec précision dans le corpus, en particulier pour assurer efficacement les fonctions de recherche et de navigation. Les passerelles mises en place entre les niveaux du modèle garantissent un repérage précis tout en évitant que le développeur des services linguistiques soit confronté à toute la diversité des formats documentaires. Ainsi, à partir de simples « adresses » (objets de la classe `TagAddr` du niveau étiquetage)[8] fournies par l'extracteur, le système est en mesure de calculer les informations de niveau indexation (objets des classes `DocAddr` et `Range` de niveau indexation) exploitées par l'indexeur pour localiser précisément et visualiser le fragment de textes (i.e. la séquence de caractères) concerné dans le document, quel que soit le format source.

Les descriptions d'interfaces WSDL de services instanciables intégrées dans l'infrastructure fournissent des spécifications techniques suffisantes pour permettre la réalisation de services externes dans tout langage supportant les web services. Pour les langages très utilisés en TAL (Java, C++ et Perl), nous proposons aussi un ensemble de composants prédéfinis (des implémentations minimales de services) sur lesquels le développeur peut prendre appui pour dériver de nouveaux services par spécialisation. Ces composants prédéfinis implémentent toutes les fonctionnalités requises par la communication inter-services (initialisation des sessions, sérialisation/désérialisation des données), de sorte que le contenu des services qui en sont dérivés est exempt de toutes considérations relatives à la communication distribuée.

6. Un cas-test dimensionnant : le couplage HyperTerm-ACABIT

Cette nouvelle plateforme a été rodée sur plusieurs applications dont ont été extraits certains services externes pour constituer une première bibliothèque de services réutilisables. Ils sont exploitables dans HyperTerm par simple modification d'un fichier de configuration. On peut citer parmi les éléments les plus significatifs de cette boîte à outils : un service de reconnaissance de termes[9], une implémentation de l'algorithme minimal d'extraction TERMS (Justeson et Katz, 1995) pour l'anglais, un service dédié à l'analyse d'exigences textuelles en ingénierie des systèmes et une version orientée service d'ACABIT. Il faut préciser que, si ces services ont été conçus à

7. Les trois niveaux pourraient être placés à l'horizontal, et chaque niveau s'appuie sur le niveau inférieur pour ajouter des propriétés enrichissant la séquence textuelle.

8. Une adresse dans une structure étiquetée est un triplet d'indices localisant la description morphologique associé à un mot dans une structure à trois niveaux d'imbrication (texte, phrase, mot) représentant l'ensemble du corpus étiqueté.

9. Contrairement à l'extraction, qui vise à identifier de nouveaux termes, la reconnaissance se contente de repérer des occurrences de termes prédéfinis. Ce service a été défini pour réaliser sur la base d'HyperTerm une application de surveillance de thèmes en veille technologique. Dans cette application, les termes prédéfinis sont les descripteurs linguistiques des thématiques à surveiller.

la base pour être invoqués par HyperTerm, ils ne sont en aucune façon dépendants de cette plateforme et sont de fait directement exploitables comme services autonomes.

Le couplage HyperTerm-ACABIT constitue une illustration représentative des facilités d'intégration offertes par cette plateforme. Ce couplage présente un réel intérêt sur le plan applicatif. Comme indiquée en § 3.2, l'une des grandes lacunes observée de manière récurrente dans les utilisations d'HyperTerm est l'absence d'un traitement efficace des variations morphologiques et syntaxiques, rares dans les corpus exploités lors de la mise au point initiale de la plateforme, mais très fréquentes dans de nombreux corpus spécialisés.

De plus, nous pensons que, pour des utilisations d'HyperTerm orientées recherche d'information, ACABIT présente des caractéristiques qui en font l'un des candidats les plus aptes à assurer la fonction d'extraction spécifiée dans le processus type supporté par HyperTerm (bon compromis pour la longueur des termes, diversité des variantes identifiées, capacité à traiter des corpus volumineux). Des études antérieures (Daille *et al.*, 2001; Jacquemin *et al.*, 2002) ont confirmé l'apport potentiel pour la recherche d'information des ressources terminologiques extraites par ACABIT et ont conduit à son intégration dans la station d'analyse de l'information STANALYST de l'INIST-CNRS.

La chaîne de traitement pour ce couplage est illustrée en figure 2. Le principe de la solution a consisté à réutiliser aussi bien les services fixes de prétraitement textuel, d'indexation et de visualisation que l'étiqueteur morphosyntaxique natif, et à insérer les adaptateurs requis pour assurer l'interopérabilité entre les services.

Plus concrètement, la mise en œuvre de cette configuration a nécessité les aménagements suivants, somme toute minimes au vu du niveau d'intégration atteint :

– Modifications apportées au cœur d'ACABIT

On touche ici à une dimension d'évaluation essentielle du cadre d'intégration proposé. L'une des raisons d'être de cette infrastructure est de permettre une réutilisation quasi-directe de services qui s'inscrivent naturellement dans un schéma fonctionnel recensé dans le modèle de services. ACABIT, en tant que composant d'extraction, entre dans cette catégorie de services aisément intégrables dans les processus HyperTerm.

L'extension apportée à ACABIT s'est limitée à rendre explicite les informations de repérage des termes requises par les services d'indexation. Ainsi, la version modifiée d'ACABIT fournit en sortie, outre les termes extraits et leurs variantes, les adresses en corpus des termes conformément au modèle de données décrit en § 5.2. Cette modification est mineure au sens où elle est réduite à l'ajout de quelques ordres d'exportation de données déjà produites par les algorithmes d'extraction d'ACABIT.

– Mise en œuvre du service « ACABIT »

L'essentiel de l'effort se situe donc aux interfaces. Les fonctions de haut niveau d'ACABIT sont exploitées pour associer des implémentations aux trois méthodes définies dans l'interface WSDL du service instanciable d'extraction (cf. tableau 1). ACABIT étant écrit en Perl, cette nouvelle instance de service s'appuie sur la librairie *SOA*-

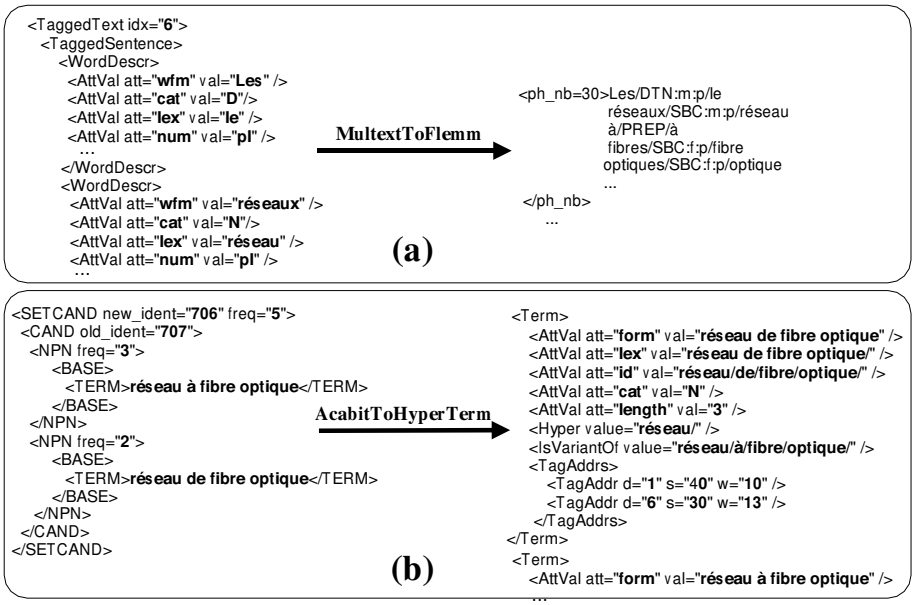

Figure 4. *Exemples de transformations de données opérées par les adaptateurs* `MultextToFlemm` *et* `AcabitToHyperTerm` *mis en place pour le couplage HyperTerm-*ACABIT

PLite[10] dédiée au développement de services en Perl.

Il convient de rappeler que le service résultant reste indépendant d'HyperTerm et de cette nouvelle infrastructure d'intégration. Ce service peut être directement intégré dans d'autres applications à architecture distribuée. En particulier, ce service se contente de manipuler des données dans les formats propres à ACABIT, et il n'a pas pour rôle de produire des termes sous une forme directement interprétable par HyperTerm. Dans la perspective d'intégration dans cette plateforme spécifique, c'est une tâche qui est dévolue aux procédés d'adaptation décrits ci-après.

– Les adaptateurs

Dans la chaîne de traitement mise en place, le service ACABIT est placé entre deux adaptateurs (cf. figure 2). ACABIT est conçu pour traiter des corpus étiquetés selon le modèle morphologique de l'analyseur Flemm (Namer, 2000), alors que l'étiqueteur interne, MultAna, utilisé dans cette chaîne, produit des données conformes au modèle Multext (Veronis et Khouri, 1995). Le premier adaptateur, nommé MultextToFlemm, assure la correspondance entre ces deux modèles morphologiques aux jeux d'étiquettes distincts mais compatibles sur le plan conceptuel. Ainsi, dans le cycle de traitement d'un corpus, chaque document analysé par MultAna est converti au format

10. http ://soaplite.com

Flemm par cet adaptateur avant d'être soumis à ACABIT[11]. Un exemple de conversion morphologique est donné en figure 4a.

Le second adaptateur, AcabitToHyperTerm, invoqué une seule fois en fin d'extraction, a pour rôle de produire à partir des sorties d'ACABIT un réseau de termes assimilable par HyperTerm. Outre l'identification de termes potentiellement pertinents, ACABIT impose sa propre structuration des termes, en tissant de multiples relations de dérivation et de variation. L'adaptateur doit interpréter ces relations pour construire un réseau de termes conforme au modèle d'HyperTerm. Dans le processus de conversion implémenté, toute relation de dérivation obtenue par insertion ou coordination dans ACABIT est directement interprétée comme une relation hiérarchique dans HyperTerm. D'autres relations impliquent des règles d'interprétation plus subtiles. En particulier, la variation est dans ACABIT une relation non orientée au sens où aucune forme-vedette n'est distinguée parmi les termes liés. C'est une information requise par le modèle d'HyperTerm. Le procédé de conversion implémenté dans l'adaptateur désigne comme vedette la forme la plus fréquente. Dans l'exemple en figure 4b, la forme « *réseau à fibre optique* », plus fréquente, est désignée comme terme vedette avec pour variante le terme « *réseau de fibre optique* ».

Ce couplage HyperTerm-ACABIT met bien en perspective les apports de l'infrastructure proposée en matière d'interopérabilité dans un contexte fortement hétérogène. Dans cette configuration de traitement, nous parvenons sans difficulté majeure à mêler des composants écrits dans différents langages : ACABIT et le service associé sont écrits en Perl, le noyau d'HyperTerm est en C++, et les adaptateurs et affineurs complémentaires en Java. Malgré cette diversité de langages, toutes les communications inter-services reposent sur le même modèle.

7. Conclusion et perspectives

Ce travail nous a permis de valider la pertinence de l'approche orientée services pour la mise en œuvre d'applications d'acquisition terminologique. Pour ce type d'applications impliquant des traitements intensifs de grands volumes de données textuelles et structurées, le passage à l'échelle contraint souvent à sacrifier à l'« esthétique » des modèles conceptuels initiaux, et le modèle d'intégration qui définit l'ossature de l'application est rarement épargné. La version réarchitecturée d'HyperTerm est une implémentation fidèle du modèle proposé, et elle a été mise à l'épreuve sur des applications en vraie grandeur. Les performances sont acceptables, même s'il conviendrait d'optimiser les opérations d'encodage et de décodage sollicitées lors de l'invocation de services externes.

11. Notons que la dernière version de Flemm adopte la représentation morphologique compacte recommandée par le consortium Multext. Si nous adoptons aussi ce format d'entrée pour ACABIT, la tâche se limiterait alors à assurer le passage du format « long » de Multext à son format compact.

D'autres applications du TAL sont à l'évidence concernées par ces questions de mutualisation et d'adaptation des services. La flexibilité apportée par cette nouvelle architecture nous a permis de viser d'autres applications impliquant des traitements semblables à l'acquisition terminologique mais de finalité différente, comme la surveillance de thèmes en veille technologique (cf. § 6). Un prolongement logique consisterait à confronter un modèle étendu de cette architecture de référence à d'autres types de systèmes représentatifs du TAL, tels que les systèmes d'extraction d'information et les systèmes de question/réponse.

Nous envisageons également de tirer profit des initiatives récentes en matière de normalisation de données linguistiques et terminologiques menées dans le cadre de l'ISO (dont la norme ISO 16642 TMF pour les ressources terminologiques). Un gain en interopérabilité pourrait résulter de l'utilisation d'un modèle de données pivot conforme à ces standards dans les descriptions d'interfaces de services. Il va de soi qu'une banalisation de ces standards contribuerait à réduire de manière significative le nombre d'adaptateurs requis dans les applications composites.

Remerciements

Nous tenons à remercier les trois relecteurs anonymes pour la précision de leurs remarques.

8. Bibliographie

Aussenac-Gilles N., Séguéla P., « Les relations sémantiques : du linguistique au formel », *Cahiers de Grammaire*, n° 25, p. 175-198, 2000.

Basili R., Di Nanni M., Pazienza M. T., « Engineering of IE Systems : An Object-Oriented Approach », *in* M. T. Pazienza (ed.), *Information Extraction : towards scalable, adaptable systems*, SpringerVerlag, Berlin Heidelberg, 1999.

Biemann C., Bordag S., Quasthoff U., Wolff C., « Web Services for Language Resources and Language Technology Applications », *Proceedings of LREC 2004*, Lisbon, p. 1209-1212, 2004.

Bourigault D., Jacquemin C., « Construction de ressources terminologiques », *in* J.-M. Pierrel (ed.), *Ingénierie des langues*, Hermès, France, p. 215-234, 2000.

Bourigault D., Jacquemin C., L'Homme M.-C. (eds), *Recent Advances in Computational Terminology*, John Benjamins, 2001.

Cabré M. T., Bagot R. E., Platresi J. V., « Automatic term detection : A review of current systems », *in* D. Bourigault, C. Jacquemin, M.-C. L'Homme (eds), *Recent Advances in Computational Terminology*, vol. 2, John Benjamins, p. 53-88, 2001.

Cerbah F., « Exogeneous and Endogeneous Approaches to Semantic Categorization of Unknown Technical Terms », *Proceedings of 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrucken, p. 145-151, 2000.

Condamines A., « Corpus Analysis and Conceptual Relation Patterns », *Terminology*, vol. 8, n° 1, p. 141-162, 2002.

Cunningham H., Software Architecture for Language Engineering, PhD thesis, Department of Computer Science, University of Shieffeld, UK, 2000.

Cunningham H., Maynard D., Bontcheva K., Tablan V., « GATE : A Framework and Graphical Development Environment for Robust NLP tools and applications », *Proceedings of 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, 2002.

Cunningham H., Wilks Y., Gaizauskas R., « GATE - A General Architecture for Text Engineering », *Proc. of 16th International Conference on Computational Linguistics (COLING'96)*, Copenhagen, 1996.

Daille B., « Conceptual structuring through term variations », *in* F. Bond, A. Korhonen, D. MacCarthy, A. Villacicencio (eds), *Proceedings ACL 2003 Workshop on Multiword Expressions : Analysis, Acquisition and Treatment*, ACL, p. 9-16, 2003.

Daille B., « Variations and application-oriented terminology engineering », *Terminology*, vol. 11, n° 1, p. 181-197, 2005.

Daille B., Royauté J., Palanco X., « Évaluation d'une plate-forme d'indexation de termes complexes », *TAL*, vol. 41, n° 2, p. 395-422, 2001.

Dalli A., Tablan V., Bontcheva K., Wilks Y., Broeder D., Brugman H., Wittenburg P., « Web Services Architecture for Language Resources », *Proc. of LREC 2004*, Lisbon, p. 365-368, 2004.

Drouin P., « Term extraction using non-technical corpora as a point of leverage », *Terminology*, vol. 9, n° 1, p. 99-117, 2003.

Ferrucci D., Lally A., « UIMA : an architectural approach to unstructured information processing in the corporate research environment », *Natural Language Engineering*, vol. 10, n° 3-4, p. 327-348, 2004.

Grishman R., Architecture Committee, TIPSTER Text Phase II Architecture Concept, version 1.1.2, Technical report, New York University, 1996.

Hamon T., Nazarenko A., « Detection of synonymy links between terms : experiment and results », *Recent Advances in Computational Terminology*, John Benjamins, p. 185-208, 2001.

Ide N., Romary L., « Standards for Language Resources », *Proceedings of the IRCS Workshop on Linguistic Databases*, Univ. of Pennsylvania, Philadelphia, p. 141-149, 11-13, 2001.

Jacquemin C., *Spotting and Discovering Terms through NLP*, MIT Press, Cambridge MA, 2001.

Jacquemin C., Daille B., Royauté J., Polanco X., « In vitro evaluation of a program for machine-aided indexing », *Information Processing and Management (IPM), Elsevier Science Ltd*, n° 38, p. 765-792, 2002.

Justeson J., Katz S., « Technical terminology : some linguistic properties and an algorithm for identification in text », *Natural Language Engineering*, vol. 1, n° 1, p. 9-27, 1995.

Kageura K., Daille B., Nakagawa H., Chien L.-F., « Recent Trends in Computional Terminology », *Terminology*, vol. 10, n° 2, p. 1-21, 2004.

Kim J.-D., Ohta T., Tsuruoka Y., Tateisi Y., Collier N., « Introduction to the Bio-Entity Recognition Task at JNLPBA », *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, Genève, 2004.

L'Homme M., « A Lexico-semantic Approach to the Structuring of Terminology », *3rd International Workshop on Computational Terminology (Computerm 2004)*, Genève, p. 7-14, 2004.

Mikheev A., Grover C., Moens M., « XML Tools and Architecture for Named Entity Recognition », *Markup Languages*, vol. 1, n° 3, p. 89-113, 1999.

Namer F., « Flemm : Un analyseur Flexionnel du Français à base de règles" », *TAL*, vol. 41, n° 2, p. 523-547, 2000.

Namer F., « Morphosémantique pour l'appariement de termes dans le vocabulaire médical : approche multilingue », *Actes TALN 2005*, 2005.

Namer F., Zweigenbaum P., « Acquiring meaning for French medical terminology : contribution of morphosemantics », *in* M. Fieschi, E. Coiera, Y. J. Li (eds), *Proc. of 10th Congress on Medical Informatics*, San Francisco, 2004.

Nazarenko A., Aït El Mekki T., « Building back-of-the-book indexes », *Terminology*, vol. 11, n° 1, p. 199-224, F. Ibekwe-SanJuan, A. Condamines and M. T. Cabré (eds), 2005.

Nazarenko A., Hamon T. (eds), *Structuration de Terminologie*, vol. 43 n 1 of *TAL*, Hermès, Paris, 2002.

Petitpierre D., Russell G., MMORPH – The Multext Morphology Program, Technical report, Multext Deliverable 2.3.1, 1995.

Reithinger N., Fedeler D., Kumar A., Lauer C., Pecourt E., Romary L., « MIAMM - A Multimodal Dialogue System Using Haptics », *in* J. van Kuppevelt, L. Dybkjaer, N. O. Bersen (eds), *Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Kluwer Academic Publisher, 2003.

Salem A., *Pratique des segments répétés. Essai de statistique textuelle*, Klincksieck, 1997.

Tartier A., Analyse automatique de l'évolution terminologique : variations et distances, Thèse en informatique, Université de Nantes, October, 2004.

Veronis J., Khouri L., « Etiquetage grammatical multilingue : le projet MULTEXT », *TAL*, vol. 39, n° 1-2, p. 233-248, 1995.

Widlöcher A., « Analyse par contraintes de l'organisation du discours », *Actes de TALN 2006*, Presses Universitaires de Louvain, Leuven, Belgique, p. 367-376, 2006.

Widlöcher A., Bilhaut F., « La plate-forme LinguaStream : un outil d'exploration linguistique sur corpus », *Actes de TALN 2005*, Dourdan, France, p. 517-522, 06, 2005.

Zajac R., Casper M., Sharples N., « An Open Distributed Architecture for Reuse and Integration of Heteregeneous NLP components », *Proceedings of ANLP'97*, 1997.