Approche générique pour l'extraction de relations à partir de textes

Seif Eddine Kramdi, Ollivier Haemmerlé et Nathalie Hernandez

IRIT – Université de Toulouse, UTM, 5 allées Antonio Machado, F-31058 Toulouse Cedex 9
{kramdi, haemmer, hernande}@irit.fr

Résumé : Cet article s'intéresse à l'extraction de relations dans le contexte du web sémantique, en vue de procéder à de la construction d'ontologies aussi bien qu'à de l'annotation automatique de documents. Notre approche permet l'extraction de relations entre entités à partir de textes. Elle ne fait pas d'hypothèse sur les entités, de manière à la rendre aussi générique que possible, et à autoriser par exemple l'extraction de relations entre concepts aussi bien que l'extraction de relations entre instances de concepts. Pour atteindre cet objectif, nous nous fondons sur l'algorithme LP^2. Afin d'adapter cet algorithme à l'extraction de relations, nous proposons une nouvelle notion de contexte reposant sur un graphe de dépendances, généré par un analyseur syntaxique. Un tel graphe de dépendances est bien adapté à la représentation de relations, puisqu'il permet, notamment, de repérer aisément les différents arguments d'un verbe dans une phrase. Nous présentons l'implémentation réalisée suivie d'une première phase d'expérimentations.

Mots-clés : Web sémantique, Extraction de connaissances, Apprentissage.

1 Introduction

La masse d'information disponible sur le Web dans des documents essentiellement textuels n'est quasiment plus exploitable sans avoir recours à des aides automatisées. L'extraction de connaissances est centrale dans ce processus d'automatisation. Des travaux visent à analyser des documents dans l'objectif d'extraire des connaissances, et ceci dans un format permettant leur exploitation par des humains ou des machines. L'extraction de connaissances peut servir à construire des ontologies en extrayant des concepts et des relations. La construction et la mise à jour d'ontologies sont des processus essentiels en vue d'une exploitation pérenne du Web. L'approche proposée dans le projet SmartWeb (Schutz & Buitelaar, 2005) pour l'extraction de relations entre concepts s'inscrit dans ce cadre. L'extraction de connaissances peut également être utilisée pour le peuplement (ou instanciation) d'ontologies. Le peuplement d'ontologies à l'aide de méthodes d'extraction de connaissances revient à trouver et associer aux concepts de ces ontologies, des instances décrites dans des textes (Cimiano, 2006). Les différentes ontologies peuplées peuvent alors être utilisées pour annoter les textes qui ont servi au peuplement.

Ces deux utilisations de l'extraction de connaissances – pour construire des ontologies d'une part, pour les peupler en vue d'annoter d'autre part – sont considérées comme faisant partie du Web Sémantique, cadre dans lequel nous plaçons nos travaux.

La particularité des extracteurs de connaissances destinés à une utilisation dans un environnement Web est qu'ils doivent tenir compte de la nature hétérogène des documents mais également, comme la plupart des applications relatives au Web, être en mesure de supporter le passage à l'échelle. L'étude des différentes méthodologies d'extraction actuelles conduit à constater la quasi-absence de démarches indépendantes du domaine d'utilisation présentant de bonnes performances, ou même, à tout le moins, de démarches ne nécessitant pas un gros effort pour leur adaptation à d'autres cas d'utilisation. Ceci est dû à la nécessité d'une grande fiabilité dans le processus d'extraction. Les systèmes développés – souvent des systèmes propriétaires – le sont pour une utilisation donnée ; ils se fondent sur des particularités du type de textes analysés, en élaborant par exemple des règles linguistiques très spécifiques (Aussenac-Gilles & Jacques, 2008). De telles règles sont très liées à la manière dont les informations sont exprimées dans le domaine d'utilisation (Fundel et al., 2007).

L'approche que nous présentons dans cet article a pour but de définir une méthode d'extraction de relations dans des textes, pouvant servir à la fois au peuplement d'ontologies, à la pose d'annotations dans des textes et à la construction d'ontologies. Elle se veut la plus généraliste possible dans le sens où elle doit permettre d'extraire des entités liées sémantiquement dans les textes. Ces entités peuvent être des instances ou des concepts, la seule hypothèse faite étant que les documents sont préalablement annotés par ces entités. Notre méthode d'extraction doit identifier la relation qui lie les entités en question. Notons que, contrairement à l'extraction d'instances de concepts – où les performances des systèmes commencent à être intéressantes, atteignant une précision d'extraction de plus de 90% pour certaines applications – l'extraction d'instances de relations est loin d'atteindre de tels résultats. Pourtant, elle représente un enjeu capital pour l'annotation automatique. Certaines démarches sont néanmoins intéressantes, comme par exemple RelExt (Schutz & Buitelaar, 2005) qui fonde son extraction sur une théorie linguistique centrée sur l'expression des relations à travers la sémantique des verbes et sur une étude statistique sur les occurrences de ces verbes. Cette approche n'est cependant proposée que pour l'enrichissement d'ontologies : les relations extraites se situent uniquement au niveau des concepts. Contrairement à certaines approches (Velardi & al., 2007), nous choisissons de ne pas nous focaliser sur un type de relation donné.

Afin de tenir compte de notre contexte d'utilisation, nous souhaitons que notre extracteur soit :

- adaptatif : l'algorithme devra pouvoir générer de manière automatique des règles d'extraction « adaptées » aux types de textes étudiés et à la nature de la relation à extraire. Il devra pouvoir repérer par quel moyen

la relation s'exprime dans les textes analysés, et ceci à chaque nouvelle utilisation ;

- autonome : l'intervention d'experts dans le processus de génération des règles doit être limitée. L'utilisation de démarches d'apprentissage semble donc opportune ;
- robuste : l'algorithme doit être en mesure de faire face au volume important de ressources à traiter sur la toile.

Notre étude se positionne dans le cadre du projet ANR WebContent qui vise à concevoir une plate-forme de développement d'applications fondées sur les technologies du web sémantique. Le but de notre étude est principalement de fournir à cette plate-forme un extracteur de relations utilisables par les diverses applications construites sur la plate-forme sous forme d'un service Web. Il pourra être utilisé, en le combinant à d'autres services Web, à l'élaboration de tâches plus complexes : annotation de documents, peuplement d'ontologies, construction d'ontologies, etc.

Ce travail est soutenu par l'ANR dans le cadre du projet plate-forme WebContent.

2 L'algorithme LP^2

LP² (Ciravegna, 2001) – pour Learning pattern by language process – est un algorithme adaptatif d'annotation : il génère des règles d'annotation adaptées aux textes traités. Il a été défini pour annoter des documents textuels à partir d'instances de concept. L'algorithme que nous proposons dans cet article est une adaptation de LP^2. Le choix de cet algorithme comme base de notre travail a été motivé par plusieurs de ses caractéristiques :

- son caractère adaptatif : en effet LP² génère des règles d'extraction plus ou moins complexes suivant la richesse du langage utilisé pour exprimer la relation dans le texte ;
- l'approche inductive de l'algorithme, qui permet de générer des règles d'extraction, à l'inverse des démarches transductives peu adaptées à l'extraction de connaissances sur de gros volumes de données ;
- ses performances qui ont été évaluées dans le cadre de différents types d'applications, comme Amilcare (Ciravegna & Wilks, 2003).

2.1 Principe de LP^2

L'algorithme LP^2 effectue sa phase d'apprentissage des règles d'extraction de connaissances à l'aide d'un ensemble d'apprentissage décomposé en deux sous-ensembles : le premier est utilisé pour la génération des règles, le deuxième servant pour sa part à la sélection des règles les plus efficaces parmi les règles générées.

Une règle d'extraction se compose de deux parties : une partie condition, où l'on vérifie l'existence d'un contexte donné – dans l'exemple suivant, nous vérifions l'existence d'un enchaînement de 5 mots – et d'une partie action qui, dans ce cas,

insère un tag de type stime – starting time – entre le 3ème et le 4ème mot, signifiant en cela qu'une instance du concept stime (heure de début) a été identifiée.

Condition	Action
Word=?	Insert Tag
the	
seminar	
at	**stime**
4	
pm	

Fig. 1 – Exemple de règle d'extraction générée par LP²

LP² est un algorithme inductif. Cela veut dire que les règles qu'il utilise sont induites à partir d'exemples, de la manière suivante :

- une fenêtre de mots appartenant à une phrase de l'ensemble destiné à la génération est considérée ; une première règle fondée sur cette fenêtre est générée (la figure 1 est l'illustration d'une telle règle) ;
- des propriétés – telles que la catégorie grammaticale du mot, le lemme, la casse, la catégorie sémantique identifiée à partir d'une ontologie, etc. – sont ajoutées pour chacun des mots de cette fenêtre (la règle présentée dans la partie gauche de la figure 2 est l'illustration d'une telle règle) ;
- La règle est ensuite généralisée de différentes manières, en relâchant les contraintes portant sur les propriétés ajoutées à l'étape précédente. Les règles généralisées ainsi produites sont testées sur l'ensemble de filtrage, de manière à sélectionner les règles présentant les meilleurs taux de rappel et de précision (la partie droite de la figure 2 présente une règle généralisée induite).

word index	Condition	Additional Knowledge				Action
	Word	Lemma	LexCat	case	SemCat	Tag
1	The	the	Art	low		
2	Seminar	Seminar	Noun	low		
3	at	at	Prep	low		**stime**
4	4	4	Digit	low		
5	pm	pm	Other	low	timeid	
6	will	will	Verb	low		

Word index	Condition					Action
	Word	Lemma	LexCat	Case	SemCat	Tag
3		at				**Stime**
4			Digit			
5					timeid	

Fig. 2 – A gauche exemple d'une règle étendue, à droite exemple d'une règle induite à partir de cette règle étendue

2.2 Limitation de LP^2 pour l'extraction de relations entre entités

Etant un algorithme destiné à la détection d'instances de concepts, LP² nécessite une adaptation afin d'autoriser l'extraction de relations entre deux entités. Une adaptation naïve pourrait consister à augmenter la fenêtre prise en compte dans la partie condition de la règle, en la faisant porter sur les contextes des occurrences des deux entités liées par la relation. La taille du contexte serait donc multipliée par deux – mots voisins des deux arguments de la relation. L'induction des propriétés d'extraction paraît alors peu adaptée, et ceci en raison :

- de l'augmentation importante du temps de calcul nécessaire à la généralisation d'un contexte trop grand ;
- de l'absence de certitude que les éléments à travers lesquels la relation est exprimée soient aux alentours des entités – un verbe éloigné, une référence qui se situe en dehors du contexte étudié, etc.

Nous proposons donc de modifier LP² en définissant une nouvelle notion de contexte, mieux adaptée à l'extraction de relations.

3 Approche proposée

3.1 Modifications de l'algorithme LP^2

Afin d'extraire des relations, nous proposons de modifier l'algorithme LP^2 en redéfinissant la notion de contexte utilisée pour l'expression des conditions des règles d'extraction. Nous proposons de repérer les éléments exprimant la relation en nous fondant sur la structure grammaticale de la phrase, grâce à une représentation sous forme de graphes de dépendances. En effet, de tels graphes permettent de repérer des relations entre des entités assez éloignées dans la phrase d'origine (Fundel et al., 2007), alors qu'elles sont proches dans le graphe. D'autre part, nous avons à notre disposition des outils d'analyse de texte permettant de générer de tels graphes automatiquement, comme par exemple (Stanford, 2008). Un graphe de dépendances a une structure d'arbre. Les nœuds fils d'un nœud donné sont appelés « expansion » du nœud, le nœud père étant appelé la « tête ». Il existe une relation de dépendance syntaxique entre une tête et ses expansions. Considérons par exemple la phrase « l'ingénieur d'Airbus a réparé l'A380 en panne », et le graphe de dépendances lui correspondant, représenté dans la figure 3.

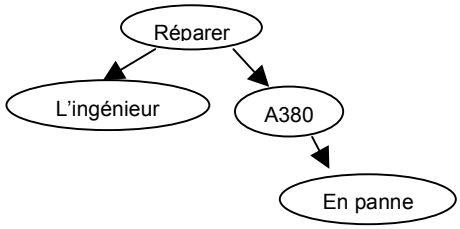

Fig. 3 – Exemple d'un graphe de dépendances

On constate sur cet exemple que la tête d'un nœud permet de décrire la structure narrative dans laquelle il intervient – l'ingénieur intervient dans une réparation, la panne est dans l'A380, etc. De même, si l'on se place du point de vue des expansions, ces dernières peuvent être vues comme des modalités modifiant le sens de leurs pères – l'A380 est qualifié comme étant en panne, etc. De plus, dans plusieurs théories linguistiques (Schutz & Buitelaar, 2005) et (Fundel et al., 2007), le verbe central – le nœud racine de l'arbre – peut être perçu comme le représentant de l'action ou de la situation décrites par la phrase – ici, le nœud central « réparer » précise qu'il s'agit d'une réparation.

Etant donné un graphe de dépendances dans lequel nous définissons deux nœuds comme étant les arguments potentiels d'une relation (arg1 et arg2), le contexte étudié pour savoir si une relation est exprimée ou non entre les arguments sera l'ensemble des nœuds têtes et expansions reliés aux deux arguments et le nœud racine de l'arbre. La figure 4 schématise le graphe de dépendances considéré.

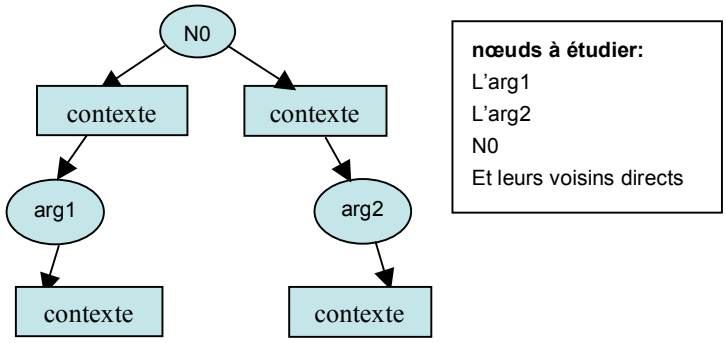

Fig. 4 – Contexte à étudier

L'essentiel des modifications que nous apportons à l'algorithme LP² peut se résumer comme suit :

- l'induction de règles d'extraction se fait maintenant sur les contextes des arguments dans un graphe de dépendances ;
- l'ajout de nouvelles propriétés sur lesquelles se fait l'induction, telle que la position du mot par rapport aux arguments (avant, après ou entre les arguments).

L'exemple suivant illustre la forme des règles que l'algorithme d'induction modifié renvoie. Considérons la phrase suivante de l'ensemble d'apprentissage, préalablement annotée : « <arg1>The 50 planes </arg1> will be <relation deliver>delivered </relation deliver> to <arg2>the airline</arg2> from 2009 to 2010. » La figure 5 représente le graphe de dépendances généré automatiquement, sur lequel nous allons fonder la partie condition de la règle induite.

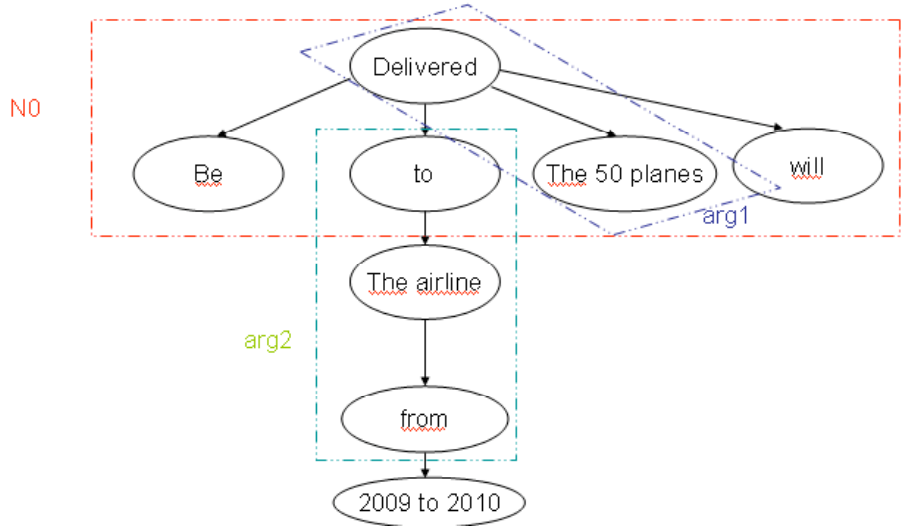

Fig. 5 – Graphe de dépendances généré à partir de la phrase extraite de l'ensemble d'apprentissage

A partir du graphe de dépendances, nous générons la règle suivante, signifiant que si la racine de l'arbre correspond à un verbe identifié comme appartenant à la catégorie sémantique « relation deliver », que son premier argument appartient à la catégorie sémantique « airline » et que son deuxième argument appartient à la catégorie sémantique « plane », alors il existe une relation « deliver » entre ces deux arguments.

Si (N0 semC "relation deliver") ➔(label:"to) ➔(arg1 Csem "airline")
➔arg2 (Csem "plane")

Alors (arg1, arg2) appartient à la relation deliver

3.2 Implémentation

Pour que notre implémentation soit aussi générale que possible, et permette la prise en compte de propriétés variées sur les nœuds du graphe de dépendances, nous

avons utilisé, pour la représentation du contexte, une structure en liste regroupant des éléments de type (nœud, propriété, valeur) sans pour autant spécifier la nature de la propriété utilisée ou les valeurs qu'elle pourrait prendre. Cette façon de procéder nous permet de laisser une grande liberté, en fonction de l'utilisation que l'on souhaite faire de l'algorithme. Par exemple, si l'on souhaite extraire des relations entre concepts plutôt qu'entre instances, les propriétés considérées pourront concerner la définition des concepts dans l'ontologie.

Le processus de génération des règles (qui seront utilisées dans le processus d'extraction) est sensiblement le même que celui adopté par Ciravegna dans (Ciravegna, 2001), mis à part le fait que la génération de règles se fait à présent sur la notion de contexte que nous proposons et qui repose sur les graphes de dépendances. La première étape du processus de génération consiste à créer une règle élémentaire pour chacun des nœuds de l'arbre enrichi par ses propriétés (catégorie sémantique obtenue à partir de l'ontologie, lemme, position dans le graphe, etc.). Ces règles élémentaires sont ensuite combinées, et les différentes combinaisons sont évaluées sur l'ensemble de filtrage des règles générées.

L'implémentation a été réalisée en java 1.5, en utilisant l'environnement de développement Eclipse. Nous avons développé environ 1000 lignes de code réparties sur 9 classes.

4 Expérimentations

Le corpus que nous avons utilisé pour l'évaluation de l'algorithme est celui proposé dans la campagne LLL05 challenge – learning language in logic 2005 (LLL, 2005). Le but de la campagne est principalement d'évaluer les performances d'algorithmes d'apprentissage pour l'extraction de règles décrivant des interactions entre des protéines et des gènes dans des résumés d'articles du domaine biomédical.
Nous considérons pour l'évaluation :

- **un corpus** composé de 55 phrases offrant 103 exemples positifs (interactions exprimées) et 27 exemples négatifs ;
- **un dictionnaire d'entités nommées** incluant les variantes et les synonymes des différents gènes et protéines rencontrés dans les corpus.

Démarche d'évaluation

Pour évaluer notre algorithme, nous avons découpé *aléatoirement* le corpus en trois ensembles : 2 ensembles pour l'apprentissage et 1 ensemble pour l'évaluation de notre approche.

Pour l'apprentissage, nous considérons :

- **un ensemble pour la production des règles** représentant l'ensemble des exemples positifs sur lesquels se fera l'induction des règles d'extraction ;

- **un ensemble de filtrage pour la sélection des règles produites** représentant l'ensemble des exemples positifs et négatifs, sur lesquels nous nous appuierons pour décider des meilleures règles induites à conserver.

Pour l'évaluation de notre approche, nous définissons :

- **un ensemble de test** constitué des exemples positifs et négatifs qui restent, et sur lesquels seront appliquées les règles générées sur l'ensemble d'apprentissage.

L'objectif de l'évaluation est d'analyser les performances de notre algorithme en faisant varier les données prises en compte pour l'apprentissage. Nous souhaitons ainsi évaluer la robustesse de notre algorithme et analyser sa pertinence pour l'extraction de relations dans le cadre du web sémantique.

Notre évaluation s'appuie sur l'analyse des trois mesures suivantes : précision (P), rappel (R) et F-Mesure (F).

Test 1 : test de stabilité : influence de la taille de l'ensemble d'apprentissage sur les résultats

Ce test fait varier la taille de l'ensemble d'apprentissage de façon à évaluer si les performances de notre algorithme sont fortement liées à la quantité d'exemples utilisés pour l'induction et le filtrage des règles.

La taille de l'ensemble d'apprentissage varie de 110 à 80 exemples. La figure 6 représente les résultats obtenus sur cet ensemble de test.

Nombre d'exemples considérés pour l'apprentissage	110	100	90	80
précision	0,53	0,56	0,51	0,61
rappel	0,30	0,29	0,33	0,33
F-mesure	0,36	0,37	0,37	0,39

Fig. 6 – Résultats du test 1

Cette évaluation montre que notre approche ne nécessite pas forcément un ensemble d'apprentissage important. La F-Mesure est plus élevée lorsque seulement 80 exemples sont utilisés pour générer les règles. Les tests suivants visent à analyser les critères permettant de sélectionner les règles capturant le plus d'exemple tout en générant le moins de bruit.

Test 2 : variations sur l'ensemble d'exemples à généraliser : influence du nombre d'exemples considérés pour la génération des règles

Pour ce test, nous faisons varier le nombre d'exemples considérés pour l'induction des règles d'extraction d'une relation. Nous considérons, pour chaque relation, 1 à 4 exemples choisis aléatoirement pour induire les règles. Nous analysons également les performances obtenues pour les relations pour lesquelles 11 exemples sont disponibles, ce qui correspond au maximum disponible dans notre corpus de test.

nombre d'exemples utilisés pour induire les règles de chaque relation	1	2	3	4	11
P	0,55	0,55	0,54	0,49	0,54
R	0,22	0,32	0,37	0,35	0,64
F	0,29	0,39	0,43	0,39	0,58

Fig. 7 – Résultats du test 2

La diversification des règles induites par l'augmentation du nombre d'exemples que l'on généralise influence directement le rappel. Ceci est essentiellement dû au fait que diversifier les règles d'extraction permet d'englober un panel plus large d'exemples positifs. Les taux de rappel et de précision sont les plus importants pour les relations pour lesquelles 11 exemples sont disponibles. Ces 11 exemples permettent de capturer un ensemble représentatif des différentes manières selon lesquelles une relation est exprimée. Cependant, autant d'exemples sont rarement disponibles.

Test 3 : variations sur l'ensemble de filtrage des règles : sélection des règles à conserver en fonction du nombre d'exemples qu'elles capturent

Ce test vise à évaluer un critère de sélection des règles générées sur l'ensemble de production des règles. Nous proposons ici de sélectionner les règles en fonction du nombre d'exemples qu'elles capturent sur l'ensemble de filtrage. Les performances de ces règles sur l'ensemble de test sont représentées dans la figure 8.

Nombre minimum d'exemples de l'ensemble de filtrage capturés par les règles	2	3	4	5
P	0,48	0,54	0,59	0,5
R	0,4	0,39	0,41	0,47
F	0,4	0,42	0,48	0,46

Fig. 8 – Résultats du test 3

En faisant varier le minimum de 2 à 5 exemples capturés, nous pouvons constater une hausse des performances du système – avec une F-mesure variant de 0,40 à 0,46. Les règles englobant plus d'exemples sont en effet moins contraignantes, ce qui permet d'augmenter le rappel. Ce test montre que ce critère de sélection garantit une certaine qualité des règles.

Test 4 : variations sur l'ensemble de filtrage des règles : sélection des n « meilleures » règles

Dans ce test, nous évaluons l'impact du nombre de règles sélectionnées sur les performances du système. Les règles induites sont classées par rapport au nombre d'exemples capturés sur l'ensemble de filtrage et nous en sélectionnons les n premières (n variant de 5 à 20). Les performances obtenues sur l'ensemble de test sont présentées dans la figure 9.

Nombre de règles sélectionnées	5	10	15	20
P	0,55	0,53	0,48	0,52
R	0,24	0,45	0,44	0,52
F	0,3	0,47	0,48	0,51

Fig. 9 – Résultats du test 4

Nous remarquons que l'augmentation du nombre de règles diminue la précision – plus de règles impliquent plus d'exemples négatifs capturés – mais de son côté, le rappel ne cesse d'augmenter – de 0,23 pour 5 règles à 0,52 pour 20 règles.

En résumé, les performances de notre algorithme sont intéressantes car il permet en moyenne d'identifier des relations avec une F-mesure proche de 0,50 lorsque pour chaque relation, au moins 4 exemples sont disponibles pour générer les règles et 6 exemples sont disponibles pour filtrer les règles générées qui sont en moyenne au nombre de 20. Ceci permet d'envisager l'utilisation de notre approche pour l'extraction de relations entre instances ou de relations entre concepts. En effet pour ce genre d'application, il est raisonnable d'envisager l'existence d'un corpus de cette taille pour l'apprentissage.

5 Conclusion

Nous avons présenté dans cet article une approche permettant l'extraction de relations entre entités à partir de textes. Cette approche ne fait pas d'hypothèse sur les entités, de manière à la rendre aussi générique que possible, et d'autoriser par exemple l'extraction de relations entre concepts aussi bien que l'extraction de relations entre instances de concepts. Pour atteindre cet objectif, nous nous sommes fondés sur l'algorithme LP^2, qui a déjà fait ses preuves dans le cadre de l'extraction d'instances de concepts. Cet algorithme fonctionne en analysant le « contexte » dans lequel les instances de concepts sont recherchées. Afin d'adapter cet algorithme à l'extraction de relations, nous avons modifié cette notion de contexte, qui ne repose plus sur une fenêtre de mots mais sur un graphe de dépendances, généré par un analyseur syntaxique. Un tel graphe de dépendances est bien adapté à la représentation de relations, puisqu'il permet, notamment, de repérer aisément les différents arguments d'un verbe dans une phrase.

Nous avons mené une première phase d'expérimentations, en utilisant un corpus de taille modeste. Les premiers résultats nous paraissent encourageants et semblent valider notre approche. Il reste néanmoins à passer à l'échelle du Web. Pour cela, nous travaillons actuellement à la transformation de notre première implémentation en un service Web, qui sera intégré à la plate-forme nationale WebContent. Dès que cette intégration aura eu lieu, nous pourrons tester notre approche dans un contexte applicatif réel, tant pour la construction d'ontologie que pour l'annotation

automatique de documents, et ce sur un ou plusieurs des domaines d'applications retenus dans le projet WebContent.

Plusieurs pistes de recherche s'offrent à nous dans la suite de ce travail :

- nous devons procéder à une meilleure évaluation des performances de notre algorithme, et le comparer à d'autres systèmes d'extraction prenant en compte des textes hétérogènes, ce qui pourrait déterminer avec plus de précision la relation qui existe entre les performances de notre approche et la nature des textes analysés et des relations extraites ;
- l'amélioration de la notion de contexte extrait à partir de graphes de dépendances, par exemple en prenant en compte les chemins, en étudiant plusieurs niveaux de voisins, etc.
- l'étude des performances du système suivant l'ensemble d'exemples choisi pour la génération des règles d'extraction. En utilisant des exemples qui exprimeraient la relation à l'aide de différents procédés narratifs, il serait vraisemblablement possible de générer des règles couvrant un maximum d'exemples.

Références

Aussenac-Gilles N. & Jacques M.-P. (2008). Designing and evaluating patterns for relation acquisition from texts with cam_el_eon. *Terminology, special issue on Pattern-Based approaches to Semantic Relations*. 14(1):45-73.

Cimiano P. (2006). Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer-Verlag.

Ciravegna F. (2001). (LP)2, an adaptive algorithm for information extraction from web-related texts. *Proceedings of IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence*.

Ciravegna F. & Wilks Y. (2003). Designing Adaptive Information Extraction for the Semantic Web in Amilcare. In Handschuh, S. and Staab S., Eds. *Annotation for the Semantic Web*. IOS Press.

Fundel K., Zimmer R. & Küffner R. (2007). RelEx---Relation extraction using dependency parse trees. *Bioinformatics*, 23(3). pp. 365-371. Oxford University Press.

Iria J. (2005). T-Rex: A Flexible Relation Extraction Framework. *8th Annual CLUK Research Colloquium*. Manchester, UK.

LLL (2005). www.cs.york.ac.uk/aig/lll/lll05/. Proceedings of the Workshop Learning Language in Logic (LLL05). Bonn, Germany.

Schutz A. & Buitelaar P. (2005). RelExt: A Tool for Relation Extraction in Ontology Extension. *Proceedings of the 4th International Semantic Web Conference*. Galway, Ireland. LNCS #3729, pp. 593-606, Springer.

Stanford (2008). nlp.stanford.edu/software/lex-parser.shtml. The Stanford Parser: A statistical parser. *The Stanford Natural Language Processing Group*.

Velardi P., Cucchiarelli A., Petit M. (2007). A taxonomy learning method and its application to characterize a scientic web community. *IEEE Transactions on Knowledge and Data Engineering*. 19(2):180-191.