Identification des rôles sémantiques par la classification

Luc Bélanger et Guy Lapalme

Laboratoire RALI
Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal (Québec) Canada H3C 3J7
{*belanglu,lapalme*}*@iro.umontreal.ca*

RÉSUMÉ. *Dans cet article, nous montrons qu'il est possible de développer un analyseur sémantique à partir d'un corpus annoté de rôles sémantiques. L'identification d'attributs sur les noeuds d'un arbre de dérivation syntaxique nous permet de considérer le problème comme un problème de classification. L'utilisation d'algorithmes de classification à base de vecteur de support permet d'obtenir un analyseur performant qui peut être utilisé dans plusieurs applications du traitement des langues naturelles.*

MOTS-CLÉS : *structures prédicat-arguments, machine à vecteur de support, rôle sémantique, analyse sémantique*

1. Introduction

L'étiquetage des rôles sémantiques est une tâche qui suscite un intérêt croissant dans le domaine du traitement automatisé des langues naturelles. La disponibilité de corpus annotés sémantiquement tels que le PropBank [KIN 02] et FrameNet [BAK 98] a largement contribué au développement d'analyseurs sémantiques. Nous nous sommes intéressé à ce type d'analyseur sémantique parce qu'ils peuvent être précis tout en ayant une couverture indépendante du domaine à traiter.

Dans cet article, nous allons montrer comment nous avons développé un analyseur sémantique à partir d'un corpus annoté sémantiquement. Le développement de notre identificateur de rôles sémantiques s'inspire grandement des travaux de Gildea et Jurafsky [GIL 02a] et de Pradhan et al. [PRA 04]. L'algorithme de classification utilisé pour l'analyseur sémantique étant une machine à vecteur de support (SVM), nous mettrons l'accent sur la modélisation du problème en un problème de classification.

2. Description de la tâche

Le problème qui nous intéresse est l'identification des structures prédicat-arguments présentes dans une phrase. Une structure prédicat-arguments est un prédicat avec ses arguments étiquetés selon le rôle qu'ils jouent dans la réalisation du prédicat. Ces structures permettent de donner une interprétation sémantique des phrases en identifiant qui a fait quoi à qui, où, quand, comment et pourquoi.

Les données que nous utilisons proviennent du corpus PropBank et du Penn TreeBank, le PropBank étant une couche sémantique ajoutée au Penn TreeBank. Le corpus est composé de deux parties, l'ensemble des frames donnant un sens aux arguments des prédicats et l'annotation des rôles sémantiques sur les arbres de dérivation syntaxique du Penn TreeBank. Chaque frame est composé d'un ensemble de rôles déterminés par les sens que peut avoir le prédicat. Le tableau 1 contient deux exemples de frame.

L'annotation d'une structure prédicat-argument se fait par l'assignation d'une étiquette préfixée par ARG, suivie d'un chiffre entre 0 et 5 ou de la lettre M suivie d'un suffixe dénotant un argument d'adjonction (12 étiquettes secondaires possibles). De façon générale l'argument ARG0 a le rôle du sujet et l'argument ARG1 celui d'objet direct. Le rôle des autres arguments variant d'un verbe à l'autre, il est impossible d'en spécifier le rôle sans utiliser le frame du prédicat. L'interprétation des rôles doit toujours être réalisée par rapport au sens que le prédicat a dans son frame. Le tableau 1 donne la correspondance entre les arguments des prédicats *purchase* et *issue* et leur rôle sémantique tel que défini dans les frames du PropBank.

Rôles	purchase	issue
ARG0	purchaser	issued
ARG1	thing purchased	thing issued
ARG2	seller	issued to
ARG3	price paid	attribute, issued as or at
ARG4	benefactive	-

TAB. 1. *Sémantique des arguments des prédicats* purchase *et* issue*, extraient du PropBank*

L'identification des arguments peut être réalisée de deux manières : en utilisant l'arbre de dérivation syntaxique de la phrase ou en analysant les caractéristiques de surface de la phrase sans utiliser la dérivation syntaxique [CAR 04]. Nous utilisons la dérivation syntaxique car ceci facilite l'extraction des attributs que nous utiliserons pour la classification et donne de meilleurs résultats, en autant que la dérivation syntaxique soit exacte [GIL 02b].

L'identification des arguments à partir de l'arbre de dérivation syntaxique débute par l'identification des prédicats. Nous identifions les prédicats à partir de l'arbre de dérivation syntaxique, un noeud dont l'étiquette est un verbe est identifié comme un prédicat. Pour chaque prédicat nous prenons ensuite tous les noeuds de l'arbre de dérivation syntaxique et nous affectons une étiquette ARG[0-5] ou ARGM aux noeuds qui sont des arguments au prédicat considéré à ce moment.

Nous considérons la détermination des étiquettes comme un problème de classification multi-classes des noeuds de l'arbre d'une dérivation. Puisque plus de 80% des noeuds de la dérivation syntaxique d'une phrase ne sont pas des arguments, l'entraînement d'un seul classificateur multi-classes sur tous les noeuds est inefficace pour deux raisons : la disproportion entre les classes amène un problème de surentraînement et les noeuds non-arguments n'ont pas à être classifiés dans les classes d'arguments.

Le problème d'efficacité lié aux noeuds non-arguments peut être contourné en éliminant ces noeuds par des heuristiques et par classificateur binaire séparant les arguments des non-arguments. Cette approche nous permet donc de diviser le problème en trois étapes :

Étape 1 : Rejet des candidats qui ne sont assurément pas des arguments et extraction des attributs pour chaque candidat restant ;

Étape 2 : Classification des candidats selon qu'ils sont ou non des arguments ;

Étape 3 : Classification multi-classes des arguments selon leur rôle.

3. Modélisation du problème

Pour identifier les arguments, il faut définir un candidat par rapport aux autres, de sorte qu'il puisse être reconnu comme appartenant à une catégorie. Un candidat se définit par un couple $\langle p, a \rangle \in \mathcal{P} \times \mathcal{A}$ où $\mathcal{P}$ est l'ensemble des prédicats et $\mathcal{A}$ est l'ensemble des noeuds d'une dérivation syntaxique. Pour chaque couple $\langle p, a \rangle$, on extrait une représentation $F_{p,a}$ sous la forme d'une liste d'attributs.

Les attributs présentés dans le tableau ci-dessous sont ceux du candidat $\langle purchase, PP \rangle$, couvrant le texte *at 7.0%*, extraits de l'arbre de dérivation syntaxique de la phrase *How can I purchase Bell Canada bonds issued at 7.0% ?*. Ces attributs sont ceux proposés dans la littérature [GIL 02a, PRA 04].

Attributs	$F_{\langle purchase, PP \rangle}$
Prédicat	*purchase*
Type de la phrase	PP
Chemin dans l'arbre du noeud au prédicat	$PP \uparrow VP \uparrow NP \uparrow VP \downarrow VB$
Position du noeud relativement au verbe (avant ou après)	*après*
Voix, le verbe est-il actif ou passif	*actif*
Mot de tête (Head Word) avec la case et la morphologie	*at*
Catégorie gouvernante, seulement si le noeud est un NP	*null*

L'algorithme de classification utilisé pour réaliser ces expériences est C-SVC, implémenté dans les logiciels LIBSVM [CHA 01] et SVM^{light} [JOA 99]. Nous avons utilisé des fonctions de noyau de type RBF et polynomial. Les données utilisées pour l'entraînement des classificateurs proviennent toutes du corpus PropBank. Le corpus est composé de 112 917 annotations couvrant 3 323 verbes et il se divise en 25 parties (wsj-00 à wsj-24). L'extraction des attributs pour l'ensemble du corpus produit 754 000 attributs en réalisant un encodage binaire de ceux-ci.

4. Résultats

4.1. *Évaluation de classificateurs pour l'identification des arguments*

Les expériences réalisées dans le but d'optimiser la première étape de classification des candidats en arguments *vs* non-arguments nécessitent énormément de temps de calcul. Nous avons fait une vingtaine d'expériences d'entraînement de classificateurs avec plusieurs logiciels et paramètres différents, tout en variant les sections du corpus utilisées.

L'entraînement d'un classificateur sur un ensemble restreint de 4 sections du PropBank, wsj-[01-04] (91 122 annotations produisant 7 255 767 candidats) à nécessité 6 jours de calcul. Selon les caractéristiques souhaitées du classificateur nous sommes capables d'obtenir des résultats similaires en n'entraînant que sur une seule section du corpus.

Puisque nous voulons utiliser la première étape de classification comme un filtre, nous voulons un taux de rappel élevé sur les arguments. Une façon d'obtenir un classificateur avec un taux de rappel élevé est de donner un poids 5 fois supérieur aux erreurs commises sur les exemples positifs avec une fonction de noyau polynomiale de degré 2. L'entraînement d'un tel classificateur, réalisé sur le corpus wsj-01 donne avec les corpus wsj-[01-04], wsj-05 et wsj-06, une moyenne de précision de 60%, de rappel de 92% et d'*accuracy* de $96,4\%$ ($|$candidats bien classifiés$|/|$candidats$|$) Pour notre problème ces résultats sont acceptables car nous récupérons plus de 90% des arguments, les 2 candidats sur 5 qui ne sont pas des arguments pourront être éliminés lors de l'étape suivante.

Nos expériences nous ont permis de constater que le problème nécessite un très grand temps de calcul pour entraîner les classificateurs, principalement à cause de la disproportion entre les candidats arguments et non-arguments. Les méthodes de classification à base de fonction de noyau ont de la difficulté à traiter les ensembles de données débalancés.

4.2. *Évaluation de classificateurs pour la classification des arguments*

Lors de la deuxième étape, nous avons entraîné 5 classificateurs binaires sur le corpus d'entraînement wsj-[02-21] ne contenant que des arguments, de cette façon nous partons de l'hypothèse que l'étape précédente classifie les arguments et non-arguments parfaitement. Les classificateurs sont entraînés pour séparer les classes d'arguments les unes des autres, ainsi le premier classificateur entraîné sépare la classe des arguments ARG0 de tous les autres arguments. Les classificateurs ont ensuite été utilisés dans une configuration un contre tous (OVA). Il n'y a pas de classificateur pour ARG4 et ARG5, leur nombre d'occurences étant trop petit. La combinaison de ces classificateurs

appliquée au corpus wsj-23 donne les résultats rapportés dans le tableau ci-dessous. Chaque ligne du tableau donne la distribution de la classification des arguments réalisés par les classificateurs. Par exemple la ligne de ARG0 est le nombre d'arguments de type ARG0 qui ont été classifiés dans chacune des catégories. La lecture horizontale du tableau donne le taux de rappel et la lecture verticale la précision du classificateur pour chacune des catégories.

	ARG0	ARG1	ARG2	ARG3	ARG4	ARG5	ARGM	null	Total	Rappel (%)
ARG0	6653	98	6	0	0	0	19	36	6812	97.67
ARG1	221	5285	36	7	0	0	60	149	5758	91.79
ARG2	18	149	1060	0	0	0	144	218	1589	66.71
ARG3	0	15	9	108	0	0	33	69	234	46.15
ARG4	0	1	14	0	0	0	11	126	152	-
ARG5	0	0	0	0	0	0	4	13	17	-
ARGM	17	44	68	3	0	0	7640	186	7958	96.00
Total	6909	5592	1193	118	0	0	7911	797	22520	
Précision (%)	96.29	94.51	88.85	91.53	-	-	96.57	-		92.12

Les résultats de cette classification sont satisfaisants dans la mesure où la classification des arguments représentant le sujet ARG0 et l'objet direct ARG1 contient peu d'erreurs.

5. Conclusion

Dans cet article, nous avons montré qu'il est possible de développer un identificateur de rôles sémantiques à partir d'un corpus annoté et d'un algorithme de classification. L'ajout de nouveaux attributs et une meilleure sélection de ceux-ci permettraient d'améliorer nos résultats. L'utilisation d'algorithmes de classification dont le temps d'entraînement est moins prohibitif est aussi à considérer pour développer ces nouveaux attributs. La méthode décrite dans cet article permet de créer un analyseur sémantique pour analyser des courriels. L'analyseur servira à extraire les arguments liés aux prédicats contenus dans le courriel dans le but d'identifier la requête pour traiter automatiquement les courriels.

6. Bibliographie

[BAK 98] BAKER C. F., FILLMORE C. J., LOWE J. B., The Berkeley FrameNet project, *Proceedings of the COLING-ACL*, Montreal, Canada, 1998.

[CAR 04] CARRERAS X., MÀRQUEZ L., Introduction to the CoNLL-2004 Shared Task : Semantic Role Labeling., *Proceedings of CoNLL 2004*, 2004.

[CHA 01] CHANG C.-C., LIN C.-J., LIBSVM : a library for support vector machines, 2001, Software available at `http ://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[GIL 02a] GILDEA D., JURAFSKY D., Automatic labeling of semantic roles, *Computational Linguistics*, vol. 28, n^o 3, 2002, p. 245–288.

[GIL 02b] GILDEA D., PALMER M., The necessity of syntactic parsing for predicate argument recognition, *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA, 2002.

[JOA 99] JOACHIMS T., *Advances in Kernel Methods - Support Vector Learning*, Chapitre 11 Making Large-Scale SVM Learning Practical, p. 41-56, MIT-Press, 1999.

[KIN 02] KINGSBURY P., PALMER M., From Treebank to PropBank, *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC2002)*, Las Palmas, Spain, 2002.

[PRA 04] PRADHAN S., WARD W., HACIOGLU K., MARTIN J. H., JURAFSKY D., Shallow Semantic parsing using support vector machines, *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistic annual meeting*, Boston, MA, May 2004, Association for Computational Linguistics.