Résolution d'anaphores nominales avec les séparateurs à vastes marges sur les arbres syntaxiques

Dimedrik Feudjieu, Paulin Melatagia Yonta

Résolution d'anaphores nominales avec les séparateurs à vastes marges sur les arbres syntaxiques

Dimedrik Feudjieu*- Paulin Melatagia Yonta*,**

*Département d'informatique, Université de Yaoundé I, Cameroun
**Sorbonne Université, IRD, UMMISCO, F-93143, Bondy, France
feudjieuvanil@gmail.com , paulinyonta@gmail.com

RÉSUMÉ. La résolution de coréférence est la tâche qui consiste à trouver toutes les expressions qui réfèrent à la même entité dans un texte. Elle est importante pour un grand nombre d'applications du TALN qui impliquent une compréhension du langage naturel. Cependant, la connaissance syntaxique est très importante pour la résolution des groupes nominaux et est traditionnellement capturée en termes de vecteurs de caractéristiques sélectionnées et définies heuristiquement. Dans cet article nous proposons un modèle basé sur les séparateurs à vastes marges et les méthodes à noyaux appliqués sur des arbres syntaxiques (plus faciles à construire et plus riches en informations nécessaires à la résolution d'anaphores nominales). De cette façon, nous évitons les efforts liés au décodage de ces arbres en vecteurs de caractéristiques. Le modèle obtenu après entraînement a été testé sur un sous-ensemble du corpus semEval task 1, les F-mesures sur les différentes métriques d'évaluation des systèmes de résolution de coréférence sont respectivement 48.36% pour le MUC, 49.53% pour le BLANC, 86.99% pour le BCUB et 78.96% pour le CEAF.

ABSTRACT. Coreference resolution is the task of finding all expressions that refer to the same entity in a text. It is an important step for a lot of NLP applications that involve natural language understanding. However syntactic knowledge is important for noun phrase resolution and is traditionally represented in terms of features vector selected and defined heuristically. In this paper we propose a model based on support vectors machine and kernel methods, applied to syntax trees (easier to construct and richer in information needed to resolve nominal anaphors). In this way, we avoid the effort involved in decoding these trees into feature vectors. The model obtained after training has been tested on a subset of the semEval task 1 corpus. The F-measures on the different evaluation metrics for the coreference resolution systems are respectively 48.36% for the MUC, 49.53% for the BLANC, 86.99% for the BCUB and 78.96% for the CEAF.

MOTS-CLÉS : Résolution de coréférence, Méthodes à noyaux, Séparateur à vastes marges, Arbre syntaxique

KEYWORDS : Coreference resolution, Kernels methods, Support vector machine, Syntaxic tree

1. Introduction

La résolution des groupes nominaux est la tâche qui consiste à trouver l'antécédent correct d'une anaphore nominale donnée dans un document. Cette tâche s'est avérée très utile dans un grand nombre d'applications du TAL, notamment le résumé automatique de texte (Saliha et al.[3]), la traduction automatique (Lesly Miculicich et al.[9]) et l'extraction de l'information (Zelenko et al.[18]). Les premières tentatives de résolution des groupes nominaux portent le nom d'approches basées sur les règles, où l'idée générale était d'incorporer une source de connaissance (un ensemble de règles) pour élaguer les candidats antécédents peu probables, jusqu'à ce qu'un petit ensemble soit obtenu, et ensuite choisir le meilleur candidat en fonction du centre d'attention actuel ou du centre préféré. Bien que précises, ces approches ont vite trouvé limite dans la difficulté à mettre en place les règles. Les deuxièmes approches sont celles basées sur l'apprentissage automatique qui exploitent un corpus annoté avec les chaines de coréférence pour mettre sur pied un modèle qui permettra à la machine plus tard la construction des chaines de coréférence pour des corpus non annotés. Ces dernières approches utilisent généralement un vecteur de caractéristiques pour capturer les informations nécessaires à la résolution nominale. Malheureusement, ces informations peuvent être incomplètes et difficiles à obtenir, ce qui conduit à l'utilisation des structures plus faciles à construire et contenant plus d'informations. Il s'agit des arbres syntaxiques.

Dans cet article, nous proposons d'utiliser les arbres syntaxiques pour effectuer la tâche de résolution nominale. Mais puisque cette tâche nécessite des informations de plusieurs niveaux de traitement du langage, nous allons précisément les enrichir avec des informations venant d'autres niveaux de traitement (lexical, morphologique et sémantique), ensuite nous utiliserons des fonctions noyau pour calculer la similarité (entre pair d'arbres) qui sera intégrée dans un séparateur à vastes marges.

Le reste du papier est organisé comme suite : la Section 2 présente quelques travaux antérieurs sur la résolution d'anaphores, la Section 3 présente les modèles à base de fonctions noyau existants pour la résolution nominale et notre proposition, la Section 4 présente nos expérimentations et discussions sur les résultats obtenus, la Section 5 conclut le papier.

2. Résolution d'anaphores

De multiples approches pour le problème de résolution de coréférence ont été mises sur pied, la section actuelle présente quelques-unes de ces approches, allant des méthodes basées sur des règles, jusqu'à celles basées sur l'apprentissage artificiel.

Jerry Hobbs[10] a élaboré un algorithme sur la base de contraintes imposées par la syntaxe de l'anglais qui prend en entrée un arbre syntaxique complet et correct, le parcourt à la recherche d'antécédents en lui appliquant diverses contraintes syntaxiques et morphologiques (traits de genre, de nombre, préférence pour les groupes nominaux). Les anaphores traitées par cet algorithme présentent un taux de réussite global assez élevé (le taux est le nombre d'anaphores trouvées sur le nombre total des anaphores), il est de 82% sur un corpus de manuels informatiques. Cependant, cette méthode ne permet pas de résoudre les pronoms dans certaines constructions car elle repose essentiellement sur l'aspect syntaxique. Shalom Lappin et Herbert J. Leass[6] résolvent ce problème en proposant un algorithme basé sur l'utilisation d'informations de nature syntaxique et

morphologique. Les tests ont donné un taux de réussite de 86%, soit 4% de plus que l'algorithme de Hobbs[10] sur le même corpus. Cette famille d'approches donne d'assez bons résultats mais a pour inconvénients de nécessiter des données provenant de divers niveaux de traitement du langage, ce qui est coûteux en temps et en ressource humaines. Les approches reposant sur l'apprentissage automatique résolvent ces problèmes comme le montre Soon et al.[12] qui utilisent un arbre de décision pour réaliser la tâche de résolution, qu'ils réduisent par ailleurs à un problème de classification. La question étant de savoir si deux mentions coréfèrent ou non. Leur modèle a été testé sur les corpus du Message Understanding Conference, MUC-6 et MUC-7 , et a obtenu une F-mesure de 62.6% respectivement 60.4%.
De nombreux algorithmes d'apprentissage automatique sont basés sur une extraction de caractéristiques c'est-à-dire qu'ils représentent chaque instance d'entrainement comme une séquence $(f_1, f_2, ..., f_m)$ de caractéristiques, communément appelée vecteur de caractéristiques, et utilisent comme mesure de similarité un produit scalaire. Le problème avec ces algorithmes est le fait que les données peuvent ne pas être facilement représentables par des vecteurs de caractéristiques explicites (par exemple, dans le langage naturel les phrases sont mieux décrites au moyen d'arbres ou même de graphes). Dans ce cas, l'extraction de caractéristiques devient d'une part très complexe, incomplète, et d'autre part peut conduire à l'obtention de vecteurs de caractéristiques de taille très élevée rendant ainsi le calcul du produit scalaire couteux. Les méthodes à noyaux sont une alternative intéressante aux méthodes basées sur les vecteurs de caractéristiques. Nous introduisons dans la suite la classe des méthodes d'apprentissage basée sur les méthodes à noyaux.

3. Les méthodes à noyaux pour la résolution nominale

3.1. Les méthodes à noyaux

Les méthodes à noyaux permettent de trouver des fonctions de décision non linéaires, tout en s'appuyant fondamentalement sur des méthodes linéaires. Ces méthodes reposent sur le concept de fonction noyau qui est une mesure de similarité satisfaisant certaines propriétés. Plus précisément, une fonction noyau K sur l'espace X est une fonction K : X $\times$ X $\rightarrow$ [0 , ∞], qui associe à chaque pair d'objets x,y $\in$ X un score K(x,y) représentant la similarité entre eux. Une fonction noyau doit être symétrique et semi-définie positive. Il est démontré que toute fonction noyau calcule implicitement le produit scalaire des données initiales dans un espace de redescription et de plus grande dimension c'est-à-dire K(x,y) = $\langle \Phi(x), \Phi(y) \rangle$. Avec Φ comme étant une fonction de X vers un espace de redescription F doté d'un produit scalaire. Les fonctions noyau peuvent êtres utilisées sur des structures discrètes (arbres, graphes, chaines de caractères) afin d'évaluer la similarité. Un exemple très répandu est la fonction subset-tree kernel proposée par Collins et Duffy[4] qui calcule la similarité entre deux arbres syntaxiques en termes du nombre de leurs sous arbres communs et pondérés de manière appropriée. C'est d'ailleurs cette fonction que nous utiliserons dans la suite de ce papier.

Pour exploiter les avantages fournis par les fonctions noyau, il faudrait les intégrer dans un algorithme d'apprentissage qui utilise un produit scalaire comme mesure de similarité (Perceptron, SVM,ACP ,etc..). Les séparateurs à vastes marges (SVM)[14] ont été choisis dans ce travail car ils permettent l'intégration de fonctions noyau (les données pourront être projetées dans un espace de très grande dimension) et sont capables de généraliser et d'éviter le surapprentissage dans les espaces de dimension extrêmement

élevée (et même infinie), en plus ils possèdent des propriétés prouvées théoriquement et pragmatiquement.

3.2. L'approche de Yang et al.

Compte tenu du succès des méthodes à noyaux dans les tâches telles que l'analyse syntaxique (Collins et Duffy[4]), l'extraction des relations (Zelenko et al.[17]), Yang et al.[16] les utilisent pour la résolution des pronoms. Selon ces auteurs, l'arbre syntaxique qui couvre une paire pronom-antécédent, serait utile dans le processus de résolution. En effet la connaissance syntaxique couramment utilisée pour la résolution des pronoms, tels que les rôles grammaticaux ou les relations de gouvernance, peut être directement décrite dans l'arbre. D'autres connaissances syntaxiques non connues, et potentiellement utiles pour la résolution pourraient aussi être implicitement représentées dans l'arbre. Ainsi en comparant les sous-structures communes entre deux arbres, il est possible de découvrir à quel degré ils contiennent des informations syntaxiques similaires. Ceci peut être fait à l'aide d'une fonction noyau de convolution, notamment la fonction subset-tree kernel de Collins et Duffy[4]. Yang et al.[16] appellent ces arbres syntaxiques des caractéristiques structurées. Mais sachant que la résolution d'anaphores est une tâche nécessitant plusieurs niveaux de traitement du langage, ils rajoutent, d'autres caractéristiques dites plates (aussi appelées vecteurs de caractéristiques) afin de capturer différents aspects comme lexical, sémantique et morphologique. Les auteurs constatent également que plus un arbre a de sous-structures, plus il est porteur d'informations, mais aussi plus il pourrait contenir des erreurs liées à l'analyse syntaxique. Pour palier ce problème, la fonction noyau subset-tree kernel est appliquée sur trois variantes de l'arbre syntaxique. La première s'appelle Min-Expansion, qui n'inclut que les nœuds se trouvant sur le chemin le plus court reliant le pronom et l'antécédent candidat. La seconde est appelée Simple-Expansion, qui inclut également les enfants de premier niveau des nœuds dans le Min-Expansion. La troisième est Full-Expansion, qui inclut tous les nœuds qui couvrent les mots entre l'antécédent candidat et le pronom.
Exemple : Soit la phrase, *[The man] in the room saw [him].*
En construisant les variantes d'arbre syntaxique précédemment présentées, on obtient les résultats ci-dessous.

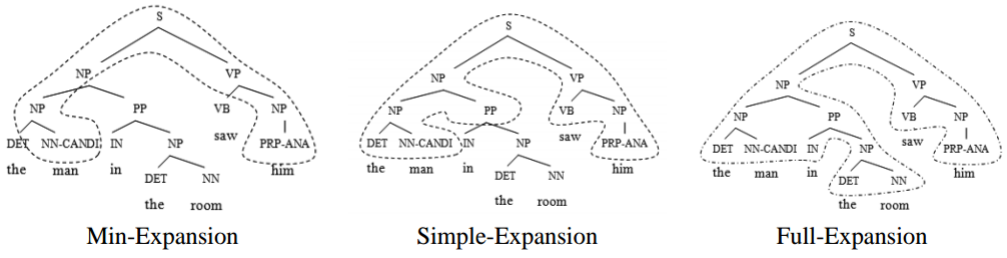


Figure 1 – Caractéristiques structurées pour l'instance {"him", "the man"}

Les caractéristiques structurées sont combinées avec des caractéristiques plates à l'aide de l'équation suivante :

$$K_c(x_1, x_2) = \frac{K_t(x_1, x_2)}{\sqrt{K_t(x_1, x_1) * K_t(x_2, x_2)}} * \frac{K_n(x_1, x_2)}{\sqrt{K_n(x_1, x_1) * K_n(x_2, x_2)}} \quad [1]$$

Où K_t est la fonction noyau sur les arbres, et K_n celle appliquée aux caractéristiques plates. K_c la fonction noyau résultante dans un SVM (Séparateurs à Vastes Marges) lors de l'entrainement. Les instances d'entrainement sont générées en utilisant l'approche proposée par Soon et al.[12].
Les modèles résultants ont été testés sur le corpus ACE-2 V1.0 de l'Automatic Content Extraction, les résultats obtenus donnent une F-mesure de 81.5% a 82.3% sur la métrique *success* (le rapport du nombre d'anaphores correctement résolus sur le nombre total des anaphores).

3.3. Arbres syntaxiques enrichis

Notre tâche est de détecter les différentes mentions nominales qui sont coreferentes. Les travaux précédents, notamment ceux de Yang et al.[16], Kong et al.[5] ont réduis ce probléme à un problème de classification binaire consistant à savoir si deux mentions données sont coréférentes. Cette tâche est délicate dans la mesure ou elle nécessite des informations de plusieurs niveaux du traitement du langage pour être correctement effectuée. Et l'idée apporter par les approches antérieures, était d'appliquer des fonctions noyaux sur des arbres syntaxiques d'une part et sur un ensemble de caracteristiques dites plates (incluant le niveau syntaxique, lexicale, semantique, etc..) d'autre part. Dans le cas de Yang et al.[16], la fonction Subset Tree Kernel (STK) de Collins et Duffy[4] est appliquée aux arbres syntaxiques en faisant un calcul récurssif entre les sous fragments de chaque paire d'arbres pour déterminer la similarité. Ensuite une seconde fonction noyaux est appliquée aux caractéristiques plates, et l'ensemble des fonctions noyaux sont combinées en utilisant un produit, afin d'êtres utilisées dans un SVM.

Cependant la fonction STK calcule la similarité de maniere récurssive entre les sous fragments de deux arbres et que cette similartité a un impact direct sur la résolution. Il est important de noter que ce calcul n'implique que le niveau syntaxique, en d'autres terme, le niveau syntaxique (apporter par les arbres syntaxique) et les autres niveaux (apporter par les caractéristiques plates) sont decouplés lors du calcul de la similarité avant d'être combinés plutard. Nous allons donc vers l'hypothèse selon laquelle la similarité entre sous fragments d'arbres pourrait être mieux calculée si chaque sous fragment incluait en dehors des informations syntaxiques, d'autres informations provenant directement d'autres niveaux du traitement du langage. Afin de répondre à ce besoin, nous avons décidé de nous affranchir de la combinaison entre fonctions noyaux en proposant une représentation syntaxique enrichie, qui permettra d'integrer en une seule structure les informations provenant de divers niveaux de traitement du langage.

Soient m_i et m_j, deux mentions, avec m_i comme antécédent de m_j. L'idée ici est d'enrichir les nœuds de l'arbre syntaxique représentant les mentions m_i et m_j avec des attributs issuent des caracteristiques plates, car celles-ci contiennent des infomations provenant de divers niveaux de traitement du langage. Et ensuite d'utiliser une version modifiée de la fonction noyau de Collins et Duffy[4] pour mésurer la similarité entre deux arbres syntaxiques enrichis. La procédure se fait en deux étapes.

La première étape qui est l'enrichissement des arbres consiste à ajouter au nœud représentant une mention *m* un ensemble d'attributs. L'ensemble des attributs ajouté au

nœud associé à *m* est :

– *Word* : c'est une chaine de caractère représentant la mention en cours de traitement.

– *Gender* : est le genre de la mention *m*. Ses valeurs possibles sont M (Masculin), F (Féminin) et N (Neutre).

– *Number* : cette caractéristique précise si *m* est au singulier (S) ou au pluriel (P).

– *Entity* : il s'agit de la catégorie d'entités nommées à laquelle appartient la mention m.

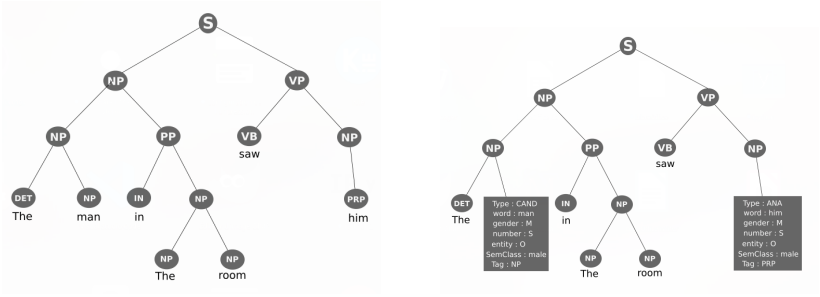
– *SemClass* : cet attribut donne la classe sémantique d'une mention.

– *Type* : désigne si la mention est un antécédent candidat (CAND) ou une anaphore (ANA).

– *Tag* : désigne l'étiquette morphosyntaxique de la mention dans l'arbre.

Exemple : Soit la phrase, *The [man] in the room [saw] him*. La Figure 2(a) représente l'arbre syntaxique associé au couple de mentions *(man ; him)*

Ces caractéristiques sont issues de travaux exploitant d'une part des connaissances lin-



(a) (b)

Figure 2 – Arbre syntaxique en (a) et arbre syntaxique enrichi en (b) de la phrase *The man in the room saw him*

guistiques existantes sur la langue Anglaise (pour que des syntagmes coréférent, ces derniers doivent porter les mêmes traits ϕ, ils doivent être en accord en genre, en nombre, en personne, etc. Adger et al.[1]) et d'autre part de la sélection des meilleurs attributs présentés par Soon et al.[12].

La seconde étape est le calcul de la similarité entre les arbres enrichis. Pour les structures discrètes comme les arbres, le calcul de similarité peut se faire avec des fonctions noyau, plus précisément des fonctions noyau de convolution. Une fonction noyau très célèbre pour les arbres syntaxiques est le subset Tree Kernel définit par Collins et Duffy[4], qui consiste à évaluer la similarité de manière récursive entre les arbres à partir de leur composantes.

Collins et Duffy[4] définissent 3 règles leur permettant d'effectuer le calcul de la fonction, il s'agit de :

– Si les productions à n_1 et n_2 sont différentes alors C(n_1 , n_2) = 0

– Si les productions à n_1 et n_2 sont identiques alors C(n_1 , n_2) = 1

– Si les productions à n_1 et n_2 sont identiques et n_1 et n_2 ne sont pas des noeuds pré-terminaux alors

$$\mathrm{C}(n_1\ ,\ n_2) = \prod_{n=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j)))$$

où $nc(n_1)$ est le nombre de noeuds fils de n_1 dans l'arbre, le i-ème fils de n_1 est $ch(n_1, i)$

Afin de prendre en compte les caractéristiques au niveau des nœuds mentions, nous avons rajouté une règle supplémentaire à celles précédentes.

– Si les nœuds n_1 et n_2 sont terminaux et représentent des mentions alors

$$\mathrm{C}(n_1\ ,\ n_2) = K_{bow}(n_1.word, n_2.word) + K_{pol}(vect(n_1), vect(n_2))$$

où K_{bow} désigne le bag-of-words kernel, K_{pol} une fonction noyau polynomiale et $vect(n_i)$ la représentation vectorielle des attributs de n_i excepté l'attribut *word*.
On obtient donc une fonction capable de calculer la similarité entre les arbres enrichis. Cette fonction est effectivement une fonction noyau compte tenu du fait que la classe des fonctions noyau est close par addition et multiplication.

Dans la section suivante nous présenterons les résultats des expérimentations effectuées avec cette nouvelle fonction à noyau sur les arbres syntaxiques enrichis.

4. Expérimentations et discussions

Dans notre étude, nous nous sommes concentrés sur la résolution des groupes nominaux en général. Toutes les expérimentations ont été réalisées sur une partie du jeu de donnée semEval 2010 task 1, qui contient 85 documents, soit 1141 phrases et 24206 mots.

Pour l'apprentissage, le classifier linéaire SVM (Séparateurs Vastes Marges, Vapnik et al.[14]), a été utilisé car il permet l'intégration des fonctions noyau afin de capturer les caractéristiques structurées sous forme d'arbre. SVM-Light-TK (Moschitti 2004) est la librairie open source écrite en langage C, qui nous a permis d'entraîner notre SVM, car elle intègre la fonction subset-tree kernel à laquelle nos modifications ont été faites. Les résultats obtenus ont été évalués à l'aide du programme de scoring fournit par SemEval (`http://stel.ub.edu/semeval2010-coref/download#working`) intégrant les 4 métriques définies pour l'évaluation des systèmes de résolution de coréférence : MUC (*Message Understanding Conference*[15]), B^3 (*B-Cubed*[2]), BLANC (*BiLateral Assessment of Noun-phrase Coreference*[8]), CEAF F-mesure (*Constrained Entity-Aligned F-measure*[7]). Les différents modèles entraînés sont : Min-Expansion (M_EXP), Simp-Expansion (S_EXP), Full-Expansion (F_EXP). Notre ensemble de données initial (85 documents) a été divisé conformément à la méthode holdout 60 (70%) documents pour l'entraînement et 25 (30%) documents pour les tests. Pour générer les instances d'entrainement, la méthode décrite par Soon et al.[12] a été appliquée sur les 60 documents, elle consiste à créer une instance positive en associant chaque anaphore nominale rencontrée et son antécédent le plus proche. Les instances négatives sont formées en associant l'anaphore avec chaque mention se trouvant entre elle et sont antécédent le plus proche. Cette méthode de génération nous conduit un jeu de données contenant plus d'instances négatives (93.5%) que positives (6.5%). Pour résoudre le problème de déséquilibre de classes, nous avons d'une part équilibré en termes d'instances positives et négatives l'ensemble d'entrainement (E_{eq}), et d'autres part généré un ensemble avec 2 fois plus (E_{do}) d'instances négatives. E_{or}, désigne le jeu de données d'origine.

Tous les modèles obtenus ont ensuite été testés sur les 25 documents restants du jeu de données original. Les différents résultats sont présentés dans les tableaux 1, 2, 3 et 4. Les

résultats obtenus avec notre proposition d'arbres syntaxiques enrichis (nommée **ExtendedST** pour **Extended Syntaxic Tree**) sont comparés avec ceux des meilleurs systèmes présentés lors de semEval 2010 task 1. Il s'agit de :

– Corry (Olga Uryupina et al.[13]) est un système de résolution de coréférence pour l'anglais, qui s'appuie sur un grand ensemble de caractéristiques (réduits au nombre de 64 pour des raisons d'efficacité). Corry utilise deux classifieurs, l'un pour la coréférence et l'autre pour l'anaphore, ensuite le résultat de chaque classifieur est combiné afin de maximiser une fonction définie par les auteurs. Ces derniers aboutissent à une conclusion selon laquelle, la mise sur pied d'un classifieur est motivé par la métrique (MUC, BLANC, CEAF, BCUB) qu'on souhaite optimiser, d'où la construction du *Corry-c* qui optimise le CEAF, *Corry-b* qui optimise la métrique BLANC et *Corry-m* le MUC.

– Relaxcor (Sapena et al.[11]) : c'est un système de résolution de coréférence basé sur le problème de satisfaction de contraintes (Constraint Satisfaction Problem). Il représente le problème sous la forme d'un graphe reliant n'importe quelle paire de mentions candidates. Il applique ensuite l'algorithme du Relaxation labelling sur le graphe pour décider des nœuds susceptibles de faire partie de la même chaine de coréférence. Ce système présente les meilleures valeurs sur la métrique BCUB.

	ExtendedST									corry-m
MUC	M_EXP			S_EXP			F_EXP			
	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{or}
Précision	84.26	59.55	0.74	79.02	54.68	13.48	85.01	80.14	35.95	56.2
Rappel	28.66	26.72	66.66	27.22	27.49	85.71	28.48	27.72	73.84	62.5
F-mesure	42.77	36.89	1.48	40.49	36.59	23.3	42.66	41.1	***48.36***	**59.2**

Tableau 1 – Résultats des expérimentations pour MUC + Corry-m SemEval 2010 task 1

	ExtendedST									corry-c
CEAF	M_EXP			S_EXP			F_EXP			
	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{or}
Précision	13.13	31.03	71.5	14.08	35.38	65.25	12.07	13.98	77.8	77.7
Rappel	15.17	35.86	71.89	16.27	40.88	75.39	13.95	16.15	80.16	77.7
F-mesure	14.08	33.27	71.69	15.1	37.93	69.96	12.94	14.99	***78.96***	**77.7**

Tableau 2 – Résultats des expérimentations pour CEAF + Corry-c SemEval 2010 task 1

	ExtendedST									relaxcor
BCUB	M_EXP			S_EXP			F_EXP			
	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{or}
Précision	95.67	92.22	71.32	95.67	92.22	72.29	95.67	92.22	76.99	96.7
Rappel	6.21	7.44	99.89	6.21	7.44	95.81	6.21	7.44	100	75.2
F-mesure	11.67	13.77	83.22	11.67	13.77	82.4	11.67	13.77	***86.99***	**84.6**

Tableau 3 – Résultats des expérimentations pour BCUB + relaxcor SemEval 2010 task 1

	ExtendedST									corry-b
BLANC	M_EXP			S_EXP			F_EXP			
	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{eq}	E_{do}	E_{or}	E_{or}
Précision	50.45	56.09	50.08	49.89	58.5	55.06	49.92	50.4	59.19	75.3
Rappel	50.41	50.73	56.17	49.94	51.01	50.8	49.71	51.18	51.76	69.3
F-mesure	6.26	33.6	49.43	9.11	33.83	46.38	3.74	9.38	***49.53***	**71.8**

Tableau 4 – Résultats des expérimentations pour BLANC + Corry-b SemEval 2010 task1

Des résultats des expérimentations effectuées, nous en déduisons que :

– Le modèle que nous proposons permet sur le jeu de données de SemEval 2010 task 1, d'obtenir une F-mesure meilleur pour CEAF et BCUB par rapport aux meilleurs modèles présentés lors de semEval 2010 task 1 (voir Tableau 2 et Tableau 3).

– Nous observons également que le modèle entraîné avec la variante de l'arbre syntaxique Min-expansion produit de meilleurs résultats que les autres quand il s'agit de la métrique MUC et sur le jeu de données E_{eq}. L'utilisation de la variante Simple-expansion majore les autres pour toutes métriques et sur le jeu de données E_{do}, à l'exception du MUC. L'utilisation de la variante Full-expansion produit les meilleurs résultats sur toutes les métriques quand il est entrainé avec E_{or}, ce qui signifie qu'il s'agit de la variante à considérer lorsqu'aucune hypothèse n'est faite sur le jeu de données.

– Il est à noter que, ExtendedST a été entraîné avec 60 documents du jeu de données SemEval 2010 task 1 (qui comporte 229 documents pour l'entraînnement). Ceci permet d'apporter plus de poids aux performances de celui-ci pour les métriques CEAF et BCUB, et d'entrevoir qu'avec tout le jeu de données, l'ensemble des résultats pourraient être améliorés et la différence observée sur MUC (resp. BLANC) par rapport à Corry-m (Corry-b) largement réduite.

5. Conclusion

Il a été question dans cet article de faire de la résolution nominale en exploitant des caractéristiques produites par les arbres syntaxiques qui sont généralement représentées sous la forme d'un vecteur. Ces caractéristiques sont généralement sélectionnées et définies

par des heuristiques, ne capturant pas nécessairement toutes les informations fournies par les arbres syntaxiques. Nous avons fait usage dans cet article, des méthodes à noyaux afin de capturer automatiquement ces informations. Plus précisément, nous avons enrichi l'arbre syntaxique et utilisé une fonction noyau pour calculer la similarité entre paires d'arbres. Les expérimentations sur une partie du jeu de données *SemEval task 1* montrent les modèles construits à partir d'arbres syntaxiques enrichis permettent une amélioration non négligeable de certaines métriques telles que le CEAF (1.26%) et le BCUB (2.39%). Une analyse approfondie de divers facteurs tels que la taille des données d'entraînement, l'ensemble des caractéristiques des nœuds mentions, nous permettra de verifier que l'approche consistant à enrichir l'arbre syntaxique peut être très performante pour la résolution de groupes nominaux.

6. Bibliographie

[1] D. Adger. *Core Syntax: A Minimalist Approach*. Core linguistics. Oxford University Press, 2003, p. 42. ISBN: 9780199243709. URL: https://books.google.cm/books?id=GMJ1QgAACAAJ.

[2] Enrique Amigó et al. "A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints". In: *Inf. Retr.* 12.4 (Aug. 2009), pp. 461–486. ISSN: 1386-4564. DOI: 10.1007/s10791-008-9066-8. URL: http://dx.doi.org/10.1007/s10791-008-9066-8.

[3] Saliha Azzam, Kevin Humphreys, and Robert J. Gaizauskas. "Using Coreference Chains for Text Summarization". In: *COREF@ACL*. 1999.

[4] Michael Collins and Nigel Duffy. "Convolution Kernels for Natural Language". In: *Advances in Neural Information Processing Systems* 14 (Nov. 2002).

[5] Kong Fang and Guodong Zhou. "Improve Tree Kernel-Based Event Pronoun Resolution with Competitive Information." In: Jan. 2011, pp. 1814–1819. DOI: 10.5591/978-1-57735-516-8/IJCAI11-304.

[6] Shalom Lappin and Herbert J. Leass. "An Algorithm for Pronominal Anaphora Resolution". In: *Comput. Linguist.* 20.4 (Dec. 1994), pp. 535–561. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=203987.203989.

[7] Xiaoqiang Luo. "On Coreference Resolution Performance Metrics". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, pp. 25–32. DOI: 10.3115/1220575.1220579. URL: https://doi.org/10.3115/1220575.1220579.

[8] Xiaoqiang Luo et al. "An Extension of BLANC to System Mentions". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*. 2014, pp. 24–29. URL: http://69.195.124.161/~aclwebor/anthology/P/P14/P14-2005.pdf.

[9] Lesly Miculicich Werlen and Andrei Popescu-Belis. "Using Coreference Links to Improve Spanish-to-English Machine Translation". In: Jan. 2017, pp. 30–40. DOI: 10.18653/v1/W17-1505.

[10] Jerry R. Hobbs. "Resolving pronoun references". In: *Lingua* 44 (Apr. 1978), pp. 311–338. DOI: 10.1016/0024-3841(78)90006-2.

[11] Emili Sapena, Lluís Padró, and Jordi Turmo. "A Constraint-Based Hypergraph Partitioning Approach to Coreference Resolution". In: *Computational Linguistics* 39 (Dec. 2013), pp. 847–884. DOI: 10.1162/COLI_a_00151.

[12] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. "A Machine Learning Approach to Coreference Resolution of Noun Phrases". In: *Computational Linguistics* 27.4 (2001), pp. 521–544. DOI: 10.1162/089120101753342653. URL: https://www.aclweb.org/anthology/J01-4004.

[13] Olga Uryupina. “Corry: A System for Coreference Resolution”. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. SemEval ’10. Los Angeles, California: Association for Computational Linguistics, 2010, pp. 100–103. URL: http://dl.acm.org/citation.cfm?id=1859664.1859684.

[14] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[15] Marc Vilain et al. “A Model-theoretic Coreference Scoring Scheme”. In: *Proceedings of the 6th Conference on Message Understanding*. MUC6 ’95. Columbia, Maryland: Association for Computational Linguistics, 1995, pp. 45–52. ISBN: 1-55860-402-2. DOI: 10.3115/1072399.1072405. URL: https://doi.org/10.3115/1072399.1072405.

[16] Xiaofeng Yang, Jian Su, and Chew Lim Tan. “Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge”. In: *ACL*. 2006.

[17] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. “Kernel Methods for Relation Extraction”. In: *Journal of Machine Learning Research* 3 (Aug. 2003), pp. 1083–1106. DOI: 10.3115/1118693.1118703.

[18] Dmitry Zelenko, Chinatsu Aone, and Jason Tibbetts. “Coreference Resolution for Information Extraction”. In: *ACL 2004: Workshop on Reference Resolution and its Applications*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 24–31. URL: https://www.aclweb.org/anthology/W04-0704.