Apprentissage d'une hiérarchie de modèles à paires spécialisés pour la résolution de la coréférence

Emmanuel Lassalle[1] Pascal Denis[2]
(1) Alpage : INRIA - Université Paris Diderot, Sorbonne Paris Cité
(2) Magnet : INRIA Nord Lille Europe - Université de Lille LIFL
emmanuel.lassalle@ens-lyon.org, pascal.denis@inria.fr

RÉSUMÉ

Nous proposons une nouvelle méthode pour améliorer significativement la performance des modèles à paires de mentions pour la résolution de la coréférence. Étant donné un ensemble d'indicateurs, notre méthode apprend à séparer au mieux des types de paires de mentions en classes d'équivalence, chacune de celles-ci donnant lieu à un modèle de classification spécifique. La procédure algorithmique proposée trouve le meilleur espace de traits (créé à partir de combinaisons de traits élémentaires et d'indicateurs) pour discriminer les paires de mentions coréférentielles. Bien que notre approche explore un très vaste ensemble d'espaces de trait, elle reste efficace en exploitant la structure des hiérarchies construites à partir des indicateurs. Nos expériences sur les données anglaises de la *CoNLL-2012 Shared Task* indiquent que notre méthode donne des gains de performance par rapport au modèle initial utilisant seulement les traits élémentaires, et ce, quelque soit la méthode de formation des chaînes ou la métrique d'évaluation choisie. Notre meilleur système obtient une moyenne de 67.2 en F1-mesure MUC, B^3 et CEAF ce qui, malgré sa simplicité, le situe parmi les meilleurs systèmes testés sur ces données.

ABSTRACT

Learning a hierarchy of specialized pairwise models for coreference resolution

This paper proposes a new method for significantly improving the performance of pairwise coreference models. Given a set of indicators, our method learns how to best separate types of mention pairs into equivalence classes for which we construct distinct classification models. In effect, our approach finds the best feature space (derived from a base feature set and indicator set) for discriminating coreferential mention pairs. Although our approach explores a very large space of possible features spaces, it remains tractable by exploiting the structure of the hierarchies built from the indicators. Our experiments on the CoNLL-2012 shared task English datasets indicate that our method is robust to different clustering strategies and evaluation metrics, showing large and consistent improvements over a single pairwise model using the same base features. Our best system obtains 67.2 of average F1 over MUC, B^3, and CEAF which, despite its simplicity, places it among the best performing systems on these datasets.

MOTS-CLÉS : résolution de la coréférence, apprentissage automatique.

KEYWORDS: coreference resolution, machine learning.

1 Introduction

La résolution de la coréférence consiste à partitionner une séquence de syntagmes nominaux (ou *mentions*) apparaissant dans un texte en un ensemble d'*entités* qui partagent chacune le même référent. Une approche désormais classique pour résoudre cette tâche consiste à la diviser en deux étapes : d'abord, on définit un modèle pour traiter les relations de coréférence indépendamment les unes des autres, en général via un classifieur binaire détectant les mentions coréférentielles. Ensuite, les liens détectés sont regroupés en *clusters* par un *décodeur* pour former une sortie cohérente. Typiquement, cette étape est réalisée par des méthodes heuristiques gloutonnes (McCarthy et Lehnert, 1995; Soon *et al.*, 2001; Ng et Cardie, 2002; Bengston et Roth, 2008), bien qu'il existe des approches plus sophistiquées telles que les méthodes de *graph cutting* (Nicolae et Nicolae, 2006; Cai et Strube, 2010) ou l'ILP (*Integer Linear Programming*) (Klenner, 2007; Denis et Baldridge, 2009). Malgré sa simplicité apparente, cette approche en deux étapes demeure compétitive même lorsqu'on la compare à des modèles plus complexes utilisant des mesures de perte globale (Bengston et Roth, 2008).

Avec ce type d'architecture, la performance du système complet dépend fortement de la qualité du classifieur local de paires. [1] Par conséquent, beaucoup de travaux de recherche ont consisté à essayer d'améliorer la performance de ce classifieur. Nombre d'entre eux se concentrent sur l'extraction de traits, typiquement en essayant d'enrichir le classifieur avec davantage de connaissances linguistiques et/ou de connaissances du monde (Ng et Cardie, 2002; Kehler *et al.*, 2004; Ponzetto et Strube, 2006; Bengston et Roth, 2008; Versley *et al.*, 2008; Uryupina *et al.*, 2011). D'autres travaux cherchent à utiliser des modèles locaux distincts pour différents types de mentions, en particulier pour différents types de mentions anaphoriques en se basant sur leur catégories grammaticales (telles que pronoms, noms propres, descriptions définies). On entraîne par exemple un modèle pour les pronoms, un autre pour les SN définis, etc (Morton, 2000; Ng, 2005; Denis et Baldridge, 2008) [2]. L'utilisation de modèles spécialisés trouve une justification importante en psycho-linguistique, dans des travaux théoriques sur la saillance ou l'accessibilité (Ariel, 1988). Du point de vue de l'apprentissage statistique, ces seconds travaux se rapprochent de ceux sur l'extraction de traits dans la mesure où les deux approches reviennent à poser le problème de la classification de paires dans un espace de plus grande dimension.

Dans ce travail, nous soutenons que les paires de mentions ne devraient pas être traitées par un seul classifieur, mais au contraire par des modèles spécifiques. En somme, nous nous intéressons à *apprendre* comment construire et sélectionner de tel modèles. Notre argumentation se fonde sur des considérations statistiques plutôt que purement linguistiques (l'approche est donc complémentaire aux études théoriques). La question que nous posons est, étant donné un ensemble d'indicateurs (tels que les types grammaticaux, la distance entre deux mentions ou le type d'entité nommée), comment séparer les paires de mentions afin de discriminer au mieux les paires coréférentielles par rapport à celles qui ne le sont pas. Ainsi, nous cherchons à apprendre les "meilleurs" espaces de représentation pour nos différents modèles : c'est-à-dire des espaces ni trop grossiers (c.-à-d. peu aptes à bien séparer les données), ni trop spécifiques (c.-à-d. pouvant souffrir du manque de données ou de bruit). Nous verrons que cette démarche est aussi équivalente à construire un seul très grand espace de traits pour représenter toutes les données.

1. Il n'y a toutefois aucune garantie théorique pour que l'amélioration de la classification locale ait toujours un impact positif sur la performance globale lorsque les deux modules sont optimisés séparément.

2. Parfois, des échantillonnages différents sont choisis lors de la phase d'apprentissage des modèles locaux distincts (Ng et Cardie, 2002; Uryupina, 2004).

Notre approche généralise les approches précédentes de plusieurs manières. D'une part, la définition des différents modèles n'est plus restreinte au simple typage grammatical (notre modèle permet d'utiliser n'importe quel type d'indicateurs) ni au seul typage de la mention anaphorique (nos modèles peuvent aussi être associés au typage de l'antécédent ou bien au types des deux éléments de la paire). D'autre part, nous proposons une méthode originale pour apprendre les meilleurs ensembles de modèles que l'on peut construire à partir d'un ensemble d'indicateurs donnés et des données d'apprentissage. Ces modèles sont organisés dans une hiérarchie où chaque feuille correspond à un ensemble de paires de mentions disjoint des autres et sur lequel un classifieur est entraîné. Nos différents modèles sont entraînés en utilisant l'algorithme *Online Passive-Aggressive*, ou *PA* (Crammer *et al.*, 2006), qui est une version à large marge du perceptron. Notre méthode peut être qualifiée d'exacte dans le sens où elle explore complètement l'espace des hiérarchies définissables à partir d'un ensemble d'indicateurs donné (on en dénombre au moins 2^{2^n} pour n indicateurs), tout en maîtrisant la complexité algorithmique par une technique de programmation dynamique qui exploite la structure particulière des hiérarchies. Cette approche obtient de très bonnes performances, et dépasse largement le modèle de départ qui utilise seulement les traits élémentaires. Comme le montreront diverses expériences sur les données anglaises de la *CoNLL-2012 Shared Task*, des améliorations importantes sont observables sur différentes métriques d'évaluation ; par ailleurs, celles-ci ne dépendent pas de la méthode de clustering choisie pour le décodeur.

La suite de cet article est organisée comme suit : dans la section 2, nous discutons les hypothèses statistiques sur lesquelles repose le modèle standard à paires de mentions, et nous définissons un modèle alternatif qui utilise une simple séparation des paires de mentions en fonction de leur type grammatical. Ensuite, dans la section 3, nous généralisons ce modèle en introduisant les hiérarchies d'indicateurs en expliquant comment apprendre le meilleur modèle possible à partir de celles-ci. La section 4 donne une brève description du système complet et la section 5 donne les résultats d'évaluation des différents modèles sur les données anglaises de CoNLL-2012.

2 Modélisation des paires

En principe, les modèles à paires emploient un seul classifieur local pour décider si deux mentions sont coréférentes ou non. Lorsque l'on utilise des techniques d'apprentissage automatique, cela entraîne quelques hypothèses sur le comportement statistique des paires de mentions.

2.1 Hypothèses statistiques

Pour commencer, adoptons un point de vue probabiliste pour décrire le prototype du modèle à paires. Étant donné un document, le nombre de mentions est fixé et chaque paire de mentions suit une certaine distribution (que l'on observe en partie en projetant les paires dans un espace de traits). L'idée fondamentale du modèle à paires est de considérer que les paires de mentions sont indépendantes les unes des autres (du coup, la propriété de transitivité n'est pas nécessairement vérifiée en sortie, c'est pourquoi il faut un décodeur la transformer en partition cohérente).

Utiliser un seul classifieur pour traiter toutes les paires de mentions revient à supposer qu'elles sont identiquement distribuées. Nous pensons que les paires ne sont pas identiquement dis-

tribuées, mais qu'il faut au contraire séparer différents "types" de paires et créer des modèles spécifiques pour ces types.

Séparer différents types de paires et les traiter avec des modèles spécifiques peut amener à des modèles globaux plus précis. Certains systèmes de résolution traitent déjà différents types d'anaphores séparément, ce qui revient à supposer que par exemple, les paires qui contiennent un pronom se comportent différemment des autres (Morton, 2000; Ng, 2005; Denis et Baldridge, 2008). Nous pourrions essayer de capturer ces différents comportements avec un ensemble très riche de traits, mais en réalité nous ne disposons que d'un nombre assez restreint de traits élémentaires (voir la section 4) et créer de nouveaux traits en les combinant doit être fait avec prudence pour éviter d'introduire du bruit dans le modèle. Au lieu de cela, nous montrerons qu'une séparation habile des instances apporte de bonnes améliorations au modèle à paires.

2.2 Espaces de traits

2.2.1 Définitions

Commençons par donner une vision plus formelle de la modélisation. Chaque paire de mentions m_i et m_j est représentée par une variable aléatoire :

$$\begin{aligned} P_{ij} : \Omega &\rightarrow \mathcal{X} \times \mathcal{Y} \\ \omega &\mapsto (x_{ij}(\omega), y_{ij}(\omega)) \end{aligned}$$

où Ω dénote classiquement l'aléatoire, $\mathcal{X}$ est l'espace des objets "paires de mentions" qui n'est pas directement observable et $y_{ij}(\omega) \in \mathcal{Y} = \{+1, -1\}$ sont les étiquettes indiquant si m_i et m_j sont coréférentes ou non. Pour alléger un peu ces notations, nous n'écrirons pas toujours l'indice ij. Maintenant nous définissons une fonction :

$$\begin{aligned} \phi_{\mathcal{F}} : \mathcal{X} &\rightarrow \mathcal{F} \\ x &\mapsto \phi_{\mathcal{F}}(x) \end{aligned}$$

qui projette les paires dans un espace de traits $\mathcal{F}$ à travers lequel elles sont observées. Pour nous, $\mathcal{F}$ est simplement un espace vectoriel sur $\mathbb{R}$ (dans notre cas, la plupart des traits sont booléens ; ils sont projetés sur $\mathbb{R}$ avec les valeurs 0 et 1). Pour des raisons de cohérence technique, nous supposons que $\phi_{\mathcal{F}_1}(x(\omega))$ et $\phi_{\mathcal{F}_2}(x(\omega))$ conservent les mêmes valeurs lorsqu'on les projette sur l'espace de traits $\mathcal{F}_1 \cap \mathcal{F}_2$: cela signifie simplement que les traits communs à deux espaces ont toujours les mêmes valeurs.

De ce point de vue formel, la tâche de résolution de la coréférence consiste à fixer un espace de traits $\mathcal{F}$, observer des échantillons étiquetés $\{(\phi_{\mathcal{F}}(x), y)_t\}_{t \in TrainSet}$ et, étant donné de nouvelles variables partiellement observées $\{(\phi_{\mathcal{F}}(x))_t\}_{t \in TestSet}$, tenter de retrouver la valeur correspondante de y.

2.2.2 Un autre point de vue sur les hypothèses statistiques

Nous avons écrit plus haut que les paires de mentions n'apparaissent pas identiquement distribuées puisque, par exemple, les pronoms ne se comportent pas de la même façon que les noms.

Nous pouvons maintenant formuler cela de façon plus rigoureuse : puisque nous ne pouvons pas observer directement l'espace des objets $\mathscr{X}$, nous en ignorons la complexité. En particulier, lorsque nous utilisons une projection vers un espace de traits trop petit, le classifieur ne parvient pas à capturer la distribution correctement : les données semblent trop bruitées.

Maintenant en remarquant que les anaphores pronominales ne se comportent pas de la même manière que les autres anaphores, nous distinguons deux types de paires, c'est-à-dire que nous voyons la distribution des paires dans $\mathscr{X}$ comme un mélange de deux distributions. De ce fait, nous pourrons peut-être séparer les paires positives et négatives plus facilement si nous projetons chaque type de paires dans un espace de traits spécifique. Appelons ces espaces de traits $\mathscr{F}_1$ et $\mathscr{F}_2$. Nous pouvons ou bien définir deux classifieurs indépendants sur $\mathscr{F}_1$ et $\mathscr{F}_2$ pour traiter chaque type de paires ou définir un seul modèle sur un espace plus grand $\mathscr{F} = \mathscr{F}_1 \oplus \mathscr{F}_2$. Si le modèle est linéaire, et ça sera notre cas, il se trouve que cela est équivalent.

En conséquence, nous pouvons de fait supposer que les variables P_{ij} sont identiquement distribuées. Et le nouveau problème à résoudre est de trouver une projection $\phi_{\mathscr{F}}$ qui représente au mieux la distribution des données (qui les rend facilement séparables).

D'un point de vue théorique, plus la dimension de l'espace des traits est grande (par exemple la somme directe de tous les espaces de traits dont nous disposons), plus nous avons de détails sur la distribution des paires de mentions et plus nous pouvons espérer séparer les positifs des négatifs avec précision. En pratique, nous sommes confrontés au problème de rareté des données : il n'y a pas assez de données pour entraîner correctement un modèle linéaire sur un tel espace. Au final, nous cherchons un espace de traits qui se situe entre les deux extrêmes que constituent un espace trop grand (données rares) ou trop petit (données bruitées). L'objectif principal de ce travail est de définir une méthode générale pour choisir l'espace $\mathscr{F}$ le plus adéquat parmi un très grand nombre de possibilités et lorsque nous ne savons pas *a priori* lequel peut être le meilleur.

2.2.3 Modèles linéaires et espaces indépendants

Dans ce travail, nous essayons de séparer linéairement les instances positives des négatives dans $\mathscr{F}$: le modèle apprend un vecteur paramètre $\mathbf{w}$ qui définit un hyperplan coupant l'espace en deux parties. La classe prédite pour la paire x avec vecteur de traits $\phi_{\mathscr{F}}(x)$ est donnée par :

$$C_{\mathscr{F}}(x) := sign(\mathbf{w}^T \cdot \phi_{\mathscr{F}}(x))$$

La propriété de linéarité rend équivalentes les séparations des instances de deux types t_1 et t_2, dans deux modèles indépendants avec pour espace de traits respectif $\mathscr{F}_1$ et $\mathscr{F}_2$ et pour paramètres $\mathbf{w}^1$ et $\mathbf{w}^2$, et un modèle simple sur $\mathscr{F}_1 \oplus \mathscr{F}_2$. Pour voir pourquoi, définissons la projection :

$$\phi_{\mathscr{F}_1 \oplus \mathscr{F}_2}(x) := \begin{cases} \begin{pmatrix} \phi_{\mathscr{F}_1}(x)^T & 0 \end{pmatrix}^T & \text{si } x \text{ est de type } t_1 \\ \begin{pmatrix} 0 & \phi_{\mathscr{F}_2}(x)^T \end{pmatrix}^T & \text{si } x \text{ est de type } t_2 \end{cases}$$

et le vecteur paramètre $\mathbf{w} = \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \end{pmatrix} \in \mathscr{F}_1 \oplus \mathscr{F}_2$. Nous avons alors :

$$C_{\mathscr{F}_1 \oplus \mathscr{F}_2}(x) = \begin{cases} C_{\mathscr{F}_1}(x) & \text{si } x \text{ est de type } t_1 \\ C_{\mathscr{F}_2}(x) & \text{si } x \text{ est de type } t_2 \end{cases}$$

Il faut maintenant s'assurer que cette propriété est vérifiée lors de l'apprentissage du paramètre **w**. Dans ce travail nous avons utilisé l'algorithme en ligne *Passive-Aggressive* pour la classification binaire (Crammer *et al.*, 2006). Ce modèle est une extension du perceptron, dont l'obectif à chaque itération est, d'une part de minimiser les changements apportés au modèle existant (d'où la caractéristique "*passive*") et, d'autre part, de faire en sorte que l'exemple courant soit correctement classifié avec une large marge (d'où la caractéristique "*aggressive*"). Plus précisément, la mise à jour du vecteur de poids à chaque itération prend la forme suivante :

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathscr{F}} \frac{1}{2}\left\|\mathbf{w}-\mathbf{w}_t\right\|^2 \quad tq \quad l(\mathbf{w};(x_t,y_t)) = 0$$

où $l(\mathbf{w};(x_t,y_t)) = \min(0, 1 - y_t(\mathbf{w}\cdot\phi_{\mathscr{F}}(x_t)))$, de sorte que lorsque $\mathscr{F} = \mathscr{F}_1 \oplus \mathscr{F}_2$, le minimum si x est de type t_1 est $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_{t+1} \\ \mathbf{w}^2_t \end{pmatrix}$ et si x est de type t_2 is $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_t \\ \mathbf{w}^2_{t+1} \end{pmatrix}$ où $\mathbf{w}^i_{t+1}$ correspond aux mises à jour dans l'espace $\mathscr{F}_i$ indépendamment du reste. Ce résultat peut être facilement étendu au cas de n espaces de traits. Par conséquent, avec une séparation déterministe des données, un modèle sur un grand espace peut être appris en le décomposant en modèles indépendants sur des espaces plus petits.

2.3 Un exemple : la séparation par *gramtype*

Pour motiver notre approche, nous commençons par introduire une séparation relativement simple des paires de mentions qui s'appuie sur les 9 modèles obtenus en considérant toutes les combinaisons possibles des types grammaticaux {*nominal, name, pronoun*} pour *les deux* mentions de la paire (une séparation fine similaire peut être trouvée dans (Chen *et al.*, 2011)).

Cela revient à utiliser 9 espaces de traits différents $\mathscr{F}_1, \ldots, \mathscr{F}_9$ pour capturer la distribution globale des paires. Avec des classifieur linéaires, nous obtenons un seul modèle sur l'espace de traits $\mathscr{F} = \mathscr{F}_1 \oplus \cdots \oplus \mathscr{F}_9$. Nous appellerons cela le modèle *gramtype*.

Comme nous le verrons dans la section 5, ces modèles séparés obtiennent des performances qui dépassent significativement celles d'un unique modèle qui utilise les mêmes traits élémentaires. Mais nous voudrions définir une méthode qui adapte l'espace de traits aux données en choisissant elle-même la séparation des paires la plus appropriée.

3 Hiérarchisation des espaces de traits

Dans cette section, nous présentons notre méthode pour trouver automatiquement une séparation optimale des paires de mentions. On gardera à l'esprit que séparer les paires dans différents modèles est la même chose que construire un grand espace de traits dans lequel le paramètre **w** peut être appris par parties dans des sous-espaces indépendants.

3.1 Indicateurs sur les paires

Pour définir des espaces de traits supplémentaires, nous utilisons des *indicateurs*, qui sont des fonctions déterministes sur les paires de mentions avec un nombre restreint de valeurs possibles.

Les indicateurs sont utilisés pour classer les paires dans des catégories prédéfinies et en bijection avec un ensemble d'espaces de traits élémentaires indépendants. Nous pouvons réutiliser les traits du système comme indicateurs, par exemple, le type grammatical ou celui des entités nommées. Nous pouvons également utiliser des fonctions qui ne sont pas des traits, par exemple la position approximative d'une des deux mentions dans le texte.

Le petit nombre de valeurs possibles pour un indicateur est requis pour des raisons pratiques : si une catégories de paires est trop fine, l'espace de traits associé souffrira de la rareté des données. Les indicateurs utilisant des distances doivent donc les approximer par des histogrammes assez grossiers. Dans nos expériences, le nombre de valeurs possibles ne dépassera jamais une douzaine (ce qui sera amplement suffisant pour générer assez de combinatoire). Une façon de réduire la taille de l'ensemble des valeurs d'un indicateur est de le binariser, de la même façon que l'on binarise un arbre (il y a plusieurs binarisations possibles). Cette opération produit une hiérarchie d'indicateurs imbriqués, qui est exactement la structure que nous exploitons dans la suite.

3.2 Des hiérarchies pour séparer les paires

Nous définissons les hiérarchies comme des combinaisons d'indicateurs créant des catégories de plus en plus fines de paires de mentions : étant donnée une suite d'indicateurs, une paire de mentions est classée en appliquant les indicateurs successivement, chaque fois en raffinant une catégorie en sous-catégories, de la même manière que dans un arbre de décision (chaque nœud ayant le même nombre d'enfants que le nombre de valeurs prises par son indicateur). Nous autorisons la classification à s'arrêter avant d'appliquer le dernier indicateur, mais le comportement doit être le même pour toutes les instances. Ainsi une hiérarchie est en principe un sous-arbre de l'arbre de décision complet qui contient des copies d'un même indicateur à chaque niveau.

Si toutes les feuilles de l'arbre de décision ont la même profondeur, cela correspond à prendre le produit cartésien des valeurs de tous les indicateurs pour indexer les catégories. Dans ce cas, nous parlerons de *hiérarchies-produit*. Le modèle *gramtype* peut être vu comme une hiérarchie-produit à deux niveaux (figure 1).

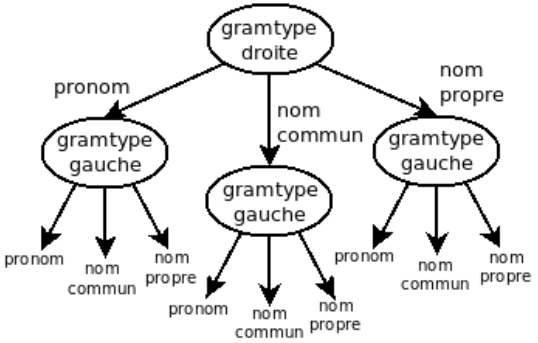

FIGURE 1 – Le modèle *gramtype* vu comme une hiérarchie-produit.

Les hiérarchies-produit seront le point de départ de notre méthode pour trouver un espace de

traits qui représente bien les données. Maintenant, pour choisir une suite d'indicateurs appropriés, il faut faire appel aux intuitions linguistiques et aux travaux théoriques sur le sujet. Le système trouvera lui-même la meilleure façon d'utiliser ces indicateurs lorsqu'il optimisera la hiérarchie. La suite d'indicateurs est donc un paramètre du modèle.

3.3 Lien entre les hiérarchies et les espaces de traits

Comme nous l'avons fait pour le modèle *gramtype*, nous associons un espace de traits $\mathscr{F}_i$ à chacune des feuilles de la hiérarchie. De la même manière, la somme $\mathscr{F} = \bigoplus_i \mathscr{F}_i$ définit un grand espace de traits, et le paramètre correspondant $\mathbf{w}$ d'un modèle linéaire peut être appris en apprenant les $\mathbf{w}^i$ dans les $\mathscr{F}_i$.

Étant donnée une séquence d'indicateurs, le nombre de hiérarchies différentes que nous pouvons définir est égal au nombre de sous-arbres entiers (chaque nœud a tous ses enfants possibles ou aucun) de l'arbre complet de décision (chaque nœud interne ayant tous ses enfants). Le cas minimal est celui d'indicateurs booléens. Le nombre d'arbre binaires entiers de taille au plus n peut être calculé par la récurrence suivante : $T(1) = 1$ et $T(n+1) = 1 + T(n)^2$. Donc $T(n) \geq 2^{2^n}$: même avec des petites valeurs de n, le nombre de hiérarchies différentes (ou de grand espaces de traits) définissables par une séquence d'indicateurs est gigantesque (p.ex. $T(10) \approx 3.8.10^{90}$).

Parmi toutes les possibilités pour un grand espace de traits, beaucoup ne sont pas appropriés parce qu'avec eux les données sont trop rares ou trop bruitées dans certains sous-espaces. Nous avons besoin d'une méthode générale pour trouver le meilleur espace sans avoir à énumérer et tester chacun d'eux.

3.4 Optimisation des hiérarchies

Considérons que la séquence d'indicateurs est fixée, soit n sa longueur. Pour trouver le meilleur espace de traits parmi un très grand nombre de possibilités, nous avons besoin d'un critère de sélection applicable sans trop de calculs supplémentaires. Pour cela, nous n'évaluons l'espace de traits que localement sur les paires, c'est-à-dire sans appliquer un décodeur à la sortie. Nous employons trois mesures sur les résultats de la classification des paires : la précision, le rappel et le F1-score. Sélectionner le meilleur espace pour une de ces mesures peut être réalisé en utilisant des techniques de programmation dynamique. Dans nos expériences, nous cherchons à optimiser le F1-score.

Entraînement de la hiérarchie : Partant de la hiérarchie-produit, nous associons un classifieur et son propre espace de traits à chacun des nœuds de l'arbre [3]. Les classifieurs sont alors entraînés comme suit : pour chaque instance, il existe un unique chemin de la racine vers une feuille de l'arbre complet. Chaque classifieur situé sur ce chemin est mis à jour avec cette instance. Le nombre d'itérations pour le *Passive-Aggressive* est fixé (nous n'avons pas cherché à optimiser ce paramètre).

Calcul des scores : Après la phase d'apprentissage, nous testons tous les classifieurs sur un autre

3. Dans les expériences, les classifieurs utilisent une copie d'un même espace de traits, mais pas les mêmes données, ce qui correspond à croiser les traits avec les catégories de l'arbre de décision.

ensemble de paires de développement [4]. Une fois encore, un classifieur est testé sur une instance seulement s'il est situé sur le chemin de la racine vers une feuille associé à l'instance. Nous obtenons des nombres TP/FP/FN [5] sur les classifications des paires, qui suffisent pour calculer le F1-score. Comme pour l'apprentissage, les données sur lesquelles un classifieur à un nœud donné est évalué sont les mêmes que la réunion de toutes les données utilisées pour évaluer les classifieurs correspondant aux enfant de ce nœud. C'est ainsi que nous sommes en mesure de comparer les scores obtenus au niveau d'un nœud à la "réunion des scores" obtenus au niveau de ses enfants.

Découpage de la hiérarchie : Pour le moment, nous avons un arbre complet avec un classifieur à chaque nœud. Nous utilisons une technique de programmation dynamique pour calculer la meilleure hiérarchie en coupant cet arbre et en ne gardant que les classifieurs situés au niveau des feuilles. L'algorithme assemble les meilleurs modèles locaux (ou espaces de traits) pour créer des modèles plus grands. Il part des feuilles pour remonter jusqu'à la racine et coupe le sous-arbre qui commence à un nœud à chaque fois qu'il ne fournit pas de meilleur score que le score du nœud seul, ou au contraire il propage le score du sous-arbre lorsqu'il y a une amélioration. Les détails sont donnés dans l'algorithme 1.

```
1 list ← list of nodes given by breadth-first search for node in reversed list do
2     if node.children ≠ ∅ then
3         if sum-score(node.children) > node.score then
4             node.TP/FP/FN ← sum-num(node.children)
5         else
6             node.children ← ∅
7         end
8     end
9 end
```

ALGORITHME 1 – Découpage de la hiérarchie

Discutons brièvement la validité et la complexité de l'algorithme. Chaque nœud n'est vu que deux fois donc la complexité est linéaire en le nombre de nœuds qui est au moins $\mathcal{O}(2^n)$. Toutefois, seulement les nœuds qui ont rencontré au moins une instance d'apprentissage sont utiles et il y en a $\mathcal{O}(n \times k)$ (où k est la taille de l'ensemble d'apprentissage). Donc nous pouvons optimiser l'algorithme pour tourner en temps $\mathcal{O}(n \times k)$ (qui est également le temps d'entraînement de la hiérarchie). En parcourant à l'envers la liste obtenue par le parcours en largeur de la hiérarchie, nous sommes assurés que chaque nœud sera traité après ses enfants donc que le modèle optimal sera construit de proche en proche jusqu'à la racine. (*node.children*) est l'ensemble des enfants de *node*, et (*node.score*) est son score. *sum-num* fournit les TP/FP/FN en sommant simplement les nombres correspondants des enfants et *sum-score* calcule le score basé sur ces nouveaux nombres TP/FP/FN. La (ligne 6) coupe les enfants d'un nœud quand ils ne sont pas utilisés pour définir le meilleur score. L'algorithme propage alors les meilleurs scores depuis les feuilles vers la racine, ce qui donne au final un seul score qui correspond à celui de la meilleure hiérarchie. Seulement les feuilles utilisées pour calculer le meilleur score sont gardées et elles définissent la meilleure hiérarchie.

Relation entre le découpage et l'espace de traits global : Nous pouvons voir l'opération de

4. Les données d'apprentissages sont coupées en deux parties, pour l'apprentissage et pour tester la hiérarchie.
5. "True positives", "false positives" et "false negatives".

découpage comme le remplacement d'un groupe de sous-espaces par un seul sous-espace dans la somme (voir figure 2). Découper la hiérarchie-produit revient donc à réduire l'espace de traits global (l'espace somme) de manière optimale. Nous voyons ici le lien entre la meilleure hiérarchie et l'espace de traits qui permet de séparer au mieux les paires.

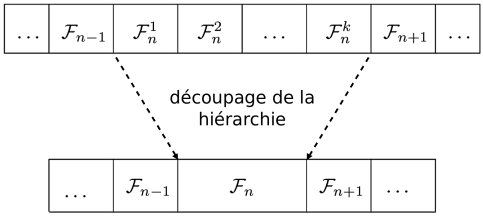

FIGURE 2 – Découper la hiérarchie réduit l'espace de traits

4 Description du système

Notre système se compose du modèle à paires séparées obtenu en découpant la hiérarchie (c'est donc un *PA* sur l'espace de traits somme) et un décodeur glouton pour créer des clusters à partir de sa sortie. Il est paramétré par le choix de la séquence initiale d'indicateurs.

Les traits élémentaires Nous avons utilisé un ensemble de traits classiques qui sont détaillés dans (Bengston et Roth, 2008) et (Rahman et Ng, 2011). Nous ne listons ici que les groupes de traits : types et sous-types grammaticaux des mentions, même chaîne/sous-chaîne de caractères, apposition, copule, distance (en nombre de mentions/phrases/mots), égalité en genre/nombre, synonymie/hyperonymie et caractère animé (en utilisant WordNet), nom de famille (à partir de liste), types d'entité nommée, traits syntaxiques (*gold parse tree*) et détection d'anaphoricité.

Indicateurs Comme indicateurs nous avons utilisé : types et sous-types grammaticaux pour les mentions gauche (antécédent) et droite (anaphore) selon l'ordre du texte, types d'entités nommées, un booléen indiquant si les mentions se trouvent dans la même phrase et un histogramme très grossier de la distance en nombre de phrase. Nous avons systématiquement commencé les séquences (de différentes longueurs) par les types grammaticaux droit et gauche, en ajoutant ensuite d'autres indicateurs. Le paramètre a été optimisé par catégorie de document en utilisant les données de développement, *après* avoir décodé la sortie du modèle à paires.

Décodeurs Nous avons testé trois stratégies gloutonnes classiques pour sélectionner les liens et former les clusters à partir des décisions du classifieur : *Closest-First* (fusionne les mentions avec la mention coréférente à gauche la plus proche, si elle existe) (Soon *et al.*, 2001), *Best-first* (fusionne les mentions avec la mention à gauche qui obtient le meilleur score positif) (Ng et Cardie, 2002; Bengston et Roth, 2008), et *Aggressive-Merge* (fermeture transitive sur les paires positives) (McCarthy et Lehnert, 1995). Chacun de ces décodeurs va typiquement de paire avec un échantillonnage particulier lors de l'apprentissage (même si ce n'est pas obligatoire).

Par exemple, *Closest-First* est combiné avec un échantillonnage où sont utilisées seulement les instances dans lesquelles la mention de gauche apparaît entre le l'anaphore et l'antécédent le plus proche (Soon *et al.*, 2001).

5 Expériences

5.1 Données

Nous avons évalué le système sur la partie anglaise du corpus fourni dans la *CoNLL-2012 Shared Task* (Pradhan *et al.*, 2012). Le corpus contient 7 catégories de documents (plus de 2k documents, 1.3M de mots). Nous avons utilisé les données d'entraînement/développement/test officielles.

5.2 Paramètres

Les hiérarchies ont été entraînées par validation croisée (*10-fold*) sur les données d'entraînement (découper les hiérarchies se fait après avoir cumulé les scores obtenus par la validation croisée) et les paramètres ont été optimisés par catégorie de documents sur les données de développement : la séquence d'indicateurs obtenant le meilleur score moyen (entre MUC, B3 et CEAF) après décodage a été sélectionné comme paramètre optimal pour la catégorie. Dans les résultats, nous appellerons *best hierarchy* la hiérarchie obtenue. Nous avons fixé le nombre d'itérations du Passive-Aggressive pour tous les modèles.

Nos baselines sont le modèle initial avec les traits élémentaires (*single model*) et le modèle *gramtype* (section 2) associés à chacun des décodeurs gloutons, et également les versions où l'on utilise ces décodeurs avec un échantillonnage particulier.

Dans nos expériences, nous ne prenons en compte que les mentions *gold* (pas de singletons ni de non-référentiels). Cela n'est pas tout à fait réaliste, mais notre but est de comparer les divers modèles à paires locaux plutôt que de mettre en place un système complet de résolution. De plus, nous voulons éviter d'avoir à considérer trop de paramètres dans nos expériences.

5.3 Métriques d'évaluation

Nous utilisons les trois métriques les plus communes, à savoir :

- **MUC** (Vilain *et al.*, 1995) calcule pour chaque vrai cluster-entité le nombre de clusters système nécessaires pour le recouvrir. La précision est cette quantité divisée par la taille du vrai cluster moins un. Le rappel est obtenu en inversant les clusters vrai et prédits. Le F1 est la moyenne harmonique du rappel et de la précision.
- **B**^{3} (Bagga et Baldwin, 1998) calcule les scores de rappel et de précision pour chaque mention, à partir de l'intersection entre le cluster système et le vrai cluster pour cette mention. La précision est le rapport des tailles de l'intersection et du cluster système, alors que le rappel est le rapport des tailles de l'intersection et du vrai cluster. Les rappel et précision globaux et le F1 sont obtenus en prenant la moyenne sur les scores des mentions.

– **CEAF** (Luo, 2005) : scores obtenus en calculant la meilleure bijection entre la vraie partition et la partition système, ce qui est équivalent à trouver l'alignement optimal dans le graphe bipartite formé par ces partitions. Nous utilisons la fonction de similarité ϕ_4 de (Luo, 2005).

Ces métriques ont été récemment utilisées dans les *Shared Task CoNLL-2011* et *2012*. Par ailleurs, ces campagnes utilisent une moyenne non pondérée sur les F1 scores donnés par ces trois métriques. Comme cela est fait normalement, nous utilisons le mode *micro-averaging* (moyennes sur le nombre de mention) lorsque nous donnons nos scores sur l'ensemble des données.

5.4 Résultats

Les résultats obtenus par le système sont repris dans les tableaux 1, 2 et 3. Les échantillonnages originaux associés aux décodeurs Closest-First et Best-First sont désignés par *Soon* et *NgCardie*. *single model* correspond à un modèle simple entraîné sans échantillonnage spécifique. Malgré l'utilisation de décodeurs gloutons, nous pouvons observer sur la sortie un effet positif très significatif sur la séparation des paires dans les modèles locaux. L'utilisation de modèles distincts plutôt qu'un seul modèle a un effet positif sur le score moyen, avec un incrément de 6.4 à 15.5 en fonction du décodeur. Il est intéressant de constater qu'indépendamment du décodeur utilisé, le modèle *gramtype* surpasse toujours le *single model*, et est lui-même dépassé par le modèle *best hierarchy*. Nous avons observé des variations dans le paramètre optimal des hiérarchies, toutefois un paramètre fréquemment bien classé était : gramtype droite → gramtype gauche → même phrase → type d'entité nommée droite.

	MUC			B^3			CEAF			
	P	R	F1	P	R	F1	P	R	F1	Mean
Soon	79.49	**93.72**	**86.02**	26.23	**89.43**	40.56	**49.74**	19.92	28.44	51.67
single model	78.95	75.15	77.0	51.88	68.42	59.01	37.79	43.89	40.61	58.87
gramtype	80.5	71.12	75.52	66.39	61.04	63.6	43.11	59.93	50.15	63.09
best hierarchy	**83.23**	73.72	78.19	**73.5**	67.09	**70.15**	47.3	**60.89**	**53.24**	**67.19**

TABLE 1 – Scores sur CoNLL-2012 avec mentions gold, décodeur *Closest-First*.

En regardant les trois différentes métriques, nous constatons que globalement, la séparation des paires améliore B^3 et CEAF (mais pas toujours MUC, à cause du très gros rappel du *single model*) après le décodage de la sortie : *gramtype* donne un meilleur score que le modèle simple, et *best hierarchy* donne les plus hauts B^3, CEAF et scores moyens.

La meilleure combinaison de classifieur-décodeur réalise un score de 67.19, ce qui la placerait au niveau des meilleurs systèmes qui ont pris part à la *CoNLL-2012 Shared Task* sur la configuration *gold mentions* (moyenne à 66.41, le premier isolé à 77, les meilleurs suivants à 68-69).

	MUC			B^3			CEAF			
	P	R	F1	P	R	F1	P	R	F1	Mean
NgCardie	**81.02**	**93.82**	**86.95**	23.33	**93.92**	37.37	**40.31**	18.97	25.8	50.04
single model	79.22	73.75	76.39	40.93	75.48	53.08	30.52	37.59	33.69	54.39
gramtype	77.21	65.89	71.1	49.77	67.19	57.18	32.08	47.83	38.41	55.56
best hierarchy	78.11	69.82	73.73	**53.62**	70.86	**61.05**	35.04	**46.67**	**40.03**	**58.27**

TABLE 2 – Score sur CoNLL-2012 avec mentions gold, décodeur *Best-First*.

	MUC			B^3			CEAF			
	P	R	F1	P	R	F1	P	R	F1	Mean
single model	83.15	**88.65**	**85.81**	35.67	**88.18**	50.79	36.3	28.27	31.78	56.13
gramtype	83.12	84.27	83.69	44.73	81.58	57.78	45.02	42.94	43.95	61.81
best hierarchy	**83.26**	85.2	84.22	**45.65**	82.48	**58.77**	**46.28**	**43.13**	**44.65**	**62.55**

TABLE 3 – Scores sur CoNLL-2012 avec mentions gold, décodeur *Aggressive-Merge*.

6 Conclusion et perspectives

Dans cet article, nous avons décrit une méthode pour construire un espace de traits séparant les paires, en exploitant la linéarité et en combinant des indicateurs pour séparer les instances. Nous avons mis en œuvre une technique de programmation dynamique pour calculer efficacement l'espace de traits fournissant la meilleure classification des paires parmi un très grand nombre de possibilités. Nous avons appliqué cette méthode pour optimiser le modèle à paires dans un système de résolution de la coréférence. En testant différents décodeurs gloutons, nous avons montré que cela apporte un gain significatif au système.

Pour ce travail, nous n'avons considéré que des stratégies heuristiques standards pour créer les clusters telles que *Closest-First* et *Best-First*. Donc une extension naturelle de ce travail serait de combiner notre méthode pour apprendre des modèles à paires avec des stratégies de décodage plus sophistiquées (comme *Mincut* ou *Integer Linear Programming*). Nous pourrons alors évaluer l'impact des hiérarchies dans des conditions plus réalistes.

Notre approche est adaptable dans le sens où elle peut s'appliquer avec des indicateurs très variés. Dans le futur, nous appliquerons les hiérarchies sur des espaces de traits plus fins pour pouvoir obtenir des optimisations plus précises. Par ailleurs, étant donné que la méthode générale de découpage des hiérarchies n'est pas spécifique à la modélisation des paires, mais peut être appliquer à d'autres problème ayant des aspects booléens, nous projetons d'employer les hiérarchies pour traiter d'autres tâches TAL (p.ex. détection d'anaphoricité, classification de relations de discours ou de relations temporelles).

La sélection d'espaces avec les hiérarchies, *si* les indicateurs sont *tous* des traits du modèle, s'apparente aux méthodes de noyaux polynomiaux. Il serait intéressant de les comparer. Par ailleurs, nous pourrons développer cette méthode en utilisant des critères statistiques pour choisir les indicateurs et construire des hiérarchies de départ plus complexes que les hiérarchies-produits, à la manière des arbres de décision. Le paramétrage du système sera alors facilité.

Références

ARIEL, M. (1988). Referring and accessibility. *Journal of Linguistics*, pages 65–87.

BAGGA, A. et BALDWIN, B. (1998). Algorithms for scoring coreference chains. *In Proceedings of LREC 1998*, pages 563–566.

BENGSTON, E. et ROTH, D. (2008). Understanding the value of features for coreference resolution. *In Proceedings of EMNLP 2008*, pages 294–303, Honolulu, Hawaii.

CAI, J. et STRUBE, M. (2010). End-to-end coreference resolution via hypergraph partitioning. *In COLING*, pages 143–151.

CHEN, B., SU, J., PAN, S. J. et TAN, C. L. (2011). A unified event coreference resolution by integrating multiple resolvers. *In Proceedings of 5th IJCNLP*, pages 102–110. Asian Federation of Natural Language Processing.

CRAMMER, K., DEKEL, O., KESHET, J., SHALEV-SHWARTZ, S. et SINGER, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

DENIS, P. et BALDRIDGE, J. (2008). Specialized models and ranking for coreference resolution. *In Proceedings of EMNLP 2008*, pages 660–669, Honolulu, Hawaii.

DENIS, P. et BALDRIDGE, J. (2009). Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 43.

KEHLER, A., APPELT, D., TAYLOR, L. et SIMMA, A. (2004). The (non)utility of predicate-argument frequencies for pronoun interpretation. *In Proceedings of HLT-NAACL 2004*.

KLENNER, M. (2007). Enforcing coherence on coreference sets. *In Proceedings of RANLP 2007*.

LUO, X. (2005). On coreference resolution performance metrics. *In Proceedings of HLT-NAACL 2005*, pages 25–32.

MCCARTHY, J. F. et LEHNERT, W. G. (1995). Using decision trees for coreference resolution. *In IJCAI*, pages 1050–1055.

MORTON, T. (2000). Coreference for NLP applications. *In Proceedings of ACL 2000*, Hong Kong.

NG, V. (2005). Supervised ranking for pronoun resolution : Some recent improvements. *In Proceedings of AAAI 2005*.

NG, V. et CARDIE, C. (2002). Improving machine learning approaches to coreference resolution. *In Proceedings of ACL 2002*, pages 104–111.

NICOLAE, C. et NICOLAE, G. (2006). Bestcut : A graph algorithm for coreference resolution. *In EMNLP*, pages 275–283.

PONZETTO, S. et STRUBE, M. (2006). Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. *In Proceedings of the HLT 2006*, pages 192–199, New York City, N.Y.

PRADHAN, S., MOSCHITTI, A., XUE, N., URYUPINA, O. et ZHANG, Y. (2012). Conll-2012 shared task : Modeling multilingual unrestricted coreference in ontonotes. *In Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

RAHMAN, A. et NG, V. (2011). Narrowing the modeling gap : a cluster-ranking approach to coreference resolution. *J. Artif. Int. Res.*, 40(1):469–521.

SOON, W. M., NG, H. T. et LIM, D. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

URYUPINA, O. (2004). Linguistically motivated sample selection for coreference resolution. *In Proceedings of DAARC 2004*, Furnas.

URYUPINA, O., POESIO, M., GIULIANO, C. et TYMOSHENKO, K. (2011). Disambiguation and filtering methods in using web knowledge for coreference resolution. *In FLAIRS Conference*.

VERSLEY, Y., MOSCHITTI, A., POESIO, M. et YANG, X. (2008). Coreference systems based on kernels methods. *In COLING*, pages 961–968.

VILAIN, M., BURGER, J., ABERDEEN, J., CONNOLLY, D. et HIRSCHMAN, L. (1995). A model-theoretic coreference scoring scheme. *In Proceedings fo the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, CA. Morgan Kaufmann.