Apprentissage symbolique et statistique pour le chunking: comparaison et combinaisons

Isabelle Tellier, Yoann Dupont
Laboratoire Lattice, 1 rue Maurice Arnoux, 92320 Montrouge
isabelle.tellier@univ-paris3.fr, yoa.dupont@gmail.com

RÉSUMÉ

Nous décrivons dans cet article l'utilisation d'algorithmes d'inférence grammaticale pour la tâche de chunking, pour ensuite les comparer et les combiner avec des CRF (Conditional Random Fields), à l'efficacité éprouvée pour cette tâche. Notre corpus est extrait du French TreeBank. Nous proposons et évaluons deux manières différentes de combiner modèle symbolique et modèle statistique appris par un CRF et montrons qu'ils bénéficient dans les deux cas l'un de l'autre.

ABSTRACT

Symbolic and statistical learning for chunking : comparison and combinations

We describe in this paper how to use grammatical inference algorithms for chunking, then compare and combine them to CRFs (Conditional Random Fields) which are known efficient for this task. Our corpus is extracted from the FrenchTreebank. We propose and evaluate two ways of combining a symbolic model and a statistical model learnt by a CRF, and show that in both cases they benefit from one another.

MOTS-CLÉS : apprentissage automatique, chunking, CRF, inférence grammaticale, k-RI, French TreeBank.

KEYWORDS: machine learning, chunking, CRF, grammatical inference, k-RI, French TreeBank.

1 Introduction

L'apprentissage automatique supervisé, surtout lorsqu'une grande quantité de données annotées est disponible, a largement prouvé son efficacité pour les tâches de fouille de textes classiques comme la classification ou l'annotation. Les bases théoriques des techniques d'apprentissage les plus performantes relèvent en général des statistiques (Naive Bayes), de l'optimisation (SVM) ou des deux (HMM, CRF). L'inconvénient principal des modèles produits par ces méthodes est qu'ils sont difficilement lisibles par un humain.

Il existe pourtant aussi d'autres branches de l'apprentissage automatique, qualifiées de *symbolique*, qui ont la particularité d'offrir une sortie généralement plus lisible par un être humain. Les plus illustres membres de cette famille sont les arbres de décision, la Programmation Logique Inductive (PLI) ou l'Inférence Grammaticale (IG par la suite) (de la Higuera, 2010). C'est cette dernière qui nous intéresse ici. On peut la définir comme l'étude des techniques permettant d'apprendre une grammaire formelle ou tout autre modèle capable de représenter *un langage* (comme un automate, une expression régulière, etc...) à partir d'exemples de séquences (éventuellement enrichies)

appartenant (ou non) à ce langage. Ce domaine, qui a son origine dans l'informatique théorique et la théorie des langages formels, est souvent méconnu. Les algorithmes d'IG sont en effet réputés ne pas très bien se comporter sur des données réelles : ils sont souvent algorithmiquement complexes, sensibles aux erreurs et peu adaptés aux langages fondés sur de grands alphabets (ce qui est le cas quand l'alphabet est l'ensemble des mots d'une langue naturelle).

Dans cet article, nous voulons donner leur chance à des algorithmes classiques d'IG pour les comparer aux méthodes d'apprentissage automatique statistique état de l'art, en l'occurrence les CRF (Lafferty et al., 2001). La tâche considérée est le *chunking* (Abney, 1991) du français, qui peut en effet très bien être réalisée à l'aide d'automates construits manuellement (Antoine et al., 2008; Blanc et al., 2010). À notre connaissance, essayer *d'apprendre automatiquement ces automates* au lieu de les écrire à la main n'a encore pas jamais été testé, pour quelque langue que ce soit. Par ailleurs, le chunking peut également être vu comme une tâche d'annotation (objet de la Shared Task CoNLL'2000) et de ce fait abordé via des méthodes d'apprentissage statistique. Ce contexte nous semblait par conséquent idéal pour comparer les deux approches.

Cette comparaison n'est cependant pas notre seul but. Notre intuition est que les deux techniques sont complémentaires car elles se concentrent sur des propriétés distinctes des données d'apprentissage. Nous proposons donc également dans cet article deux manières différentes de les combiner, en fonction du but visé. La première manière est orientée vers l'efficacité : elle vise à enrichir un modèle CRF à l'aide d'informations extraites des automates. La seconde privilégie la lisibilité : elle propose d'analyser les automates appris par IG à l'aide de poids calculés par un CRF, poids qui seront tous interprétables relativement à cet automate.

L'article suit le plan suivant. Dans la première section, nous introduisons la tâche de chunking et décrivons les données utilisées pour nos expériences. La deuxième section est dédiée à l'inférence grammaticale. Après un bref état de l'art, nous détaillons la famille des algorithmes k-RI (Angluin, 1982) et donnons les meilleurs résultats expérimentaux qu'ils permettent d'atteindre pour le chunking. Dans la section qui suit, nous appliquons les CRF à la même tâche. Comme on pouvait s'y attendre, les CRF donnent de bien meilleurs résultats que ceux obtenus par IG. Dans la dernière section, nous décrivons et évaluons deux manières de combiner automates et CRF. Les résultats obtenus pour chacune de ces combinaisons sont prometteurs et suggèrent des pistes originales pour associer modèles symboliques et apprentissage statistique.

2 Chunking: la tâche et les données

Nous décrivons ici la tâche de chunking par annotation et nous présentons les données d'apprentissage que nous avons utilisées pour nos expériences. Ces dernières reprennent et prolongent celles présentées dans (Tellier et al., 2012). Notre but étant de construire un chunker pour le français, nous sommes partis du French Tree Bank (Abeillé et al., 2003).

2.1 La tâche

La tâche de chunking, également appelée *analyse syntaxique de surface*, a pour but d'identifier les groupes syntaxiques élémentaires des phrases. Les chunks sont en effet des *séquences contigües et non-récursives d'unités lexicales liées à une unique tête forte* (Abney, 1991). Chacun est caractérisé

par le type (ou étiquette Part-Of-Speech (POS)) de sa tête. Il y a ainsi autant de types de chunks que de types de têtes fortes possibles.

La tâche de chunking a fait l'objet de de la compétition CoNLL'2000[1], dont le corpus d'apprentissage était constitué d'environ 9 000 phrases issues du Penn Treebank, associées à deux niveaux d'annotion : un niveau POS donné par l'étiqueteur Brill et un de chunking. Les vainqueurs avaient utilisé des SVM et des "Weighted Probability Distribution Voting". Ce même corpus a aussi servi plus tard à montrer l'efficacité des CRF (Sha and Pereira, 2003).

2.2 Les données

Le French TreeBank (FTB) est un recueil de phrases extraites d'articles du journal "Le Monde" publiés entre 1989 et 1993 (Abeillé et al., 2003). Les phrases ont été tokenisées (en conservant certaines unités multi-mots), lemmatisées, étiquetées et analysées syntaxiquement. Il existe plusieurs variantes du FTB, celle que nous avons utilisée contenait environ 8 600 arbres XML enrichis de fonctions syntaxiques (parfois nécessaires pour identifier certains chunks). Pour le POS, nous avons repris les 30 étiquettes morpho-syntaxiques définies dans (Crabbé and Candito, 2008), assurant ainsi la continuité avec nos précédents travaux (Constant et al., 2011).

Nous considérons 7 types de chunks distincts : AP (Adjectival Phrase), AdP (Adverbial Phrase), CONJ (Conjonctions), NP (Noun Phrase), PP (Prepositional Phrase), VP (verbal Phrase) et UNKONWN (coquilles ou certains mots étrangers, eux-mêmes étiquetés UNKNOWN). Les marques de ponctuations, sauf exceptions (certains guillemets par exemple) sont hors chunks (étiquette O comme Out). Nous avons décidé de modifier certains choix que nous avions faits dans (Tellier et al., 2012). Par exemple, le chunk CONJ contient seulement la conjonction. Le PP, en revanche, intègre toujours le chunk introduit par la préposition. Et, à l'inverse de (Paroubek et al., 2006), les adjectifs épithètes appartiennent toujours au chunk NP contenant le nom qu'ils qualifient, qu'ils soient situés avant ou après lui. Les chunks AP sont donc assez rares car ils ne correspondent qu'aux adjectifs séparés d'un groupe nominal, comme les attributs du sujet ou de l'objet (les fonctions syntaxiques disponibles dans les arbres XML sont nécessaires pour identifier ces derniers). La phrase suivante illustre notre notion de parenthésage en chunks[2] :
(la/DET dépréciation/NC)$_{NP}$ (par_rapport_au/P dollar/NC)$_{PP}$ (a/V été/VPP limitée/VPP)$_{VP}$ (à/P 2,5/DET %/NC)$_{pp}$

Nous avons extrait du FTB deux corpus distincts, chacun représentant un chunking différent :

• un corpus où tous les chunks sont extraits et étiquetés selon le modèle BIO (Begin/In/Out). Les proportions de chaque type de chunk trouvées dans le corpus sont les suivantes : PP : 33,86%, AdP : 7,23%, VP : 17,11%, AP : 2,21%, NP : 32,95%, CONJ : 6,61%, UNKNOWN : 0,03%.

• un corpus où seuls les NP sont étiquetés, tout autre groupe étant alors considéré O. Ce corpus n'est pas un sous-ensemble du précédent : par exemple, de nombreux PP incluent un NP qui ne devient visible que dans ce deuxième corpus. L'exemple précédent devient ainsi :
(la/DET dépréciation/NC)$_{NP}$ par_rapport_au/P (dollar/NC)$_{NP}$ a/V été/VPP limitée/VPP à/P (2,5/DET %/NC)$_{NP}$

[1] http://www.cnts.ua.be/conll2000/chunking
[2] guide complet disponible sur : `http://www.lattice.cnrs.fr/sites/itellier/guide.html`

3 L'inférence grammaticale

L'inférence grammaticale (IG) est un domaine de recherche très riche apparu dans les années 60 dont il n'est, par conséquent, pas aisé de faire un résumé. Nous nous plaçons ici dans le cadre de *l'IG d'automates par exemples positifs seuls*. Après un bref état de l'art, nous décrivons les algorithmes k-RI (Angluin, 1982) utilisés dans nos expériences et les résultats obtenus avec eux.

3.1 Bref état de l'art

L'IG étudie les différentes manières d'apprendre automatiquement un dispositif symbolique capable de représenter un *langage* (comme une grammaire formelle, un automate, etc...) à partir d'un ensemble de séquences (parfois enrichies) regroupées selon leur (non-)appartenance à ce langage (de la Higuera, 2010). Lorsque seules des séquences appartenant au langage cible sont disponibles, le problème est appelé *IG par présentation positive*. Nous nous situons dans ce cadre car les séquences à notre disposition ne comportent aucun contre-exemple. L'IG est, dans ce cas, notoirement plus difficile car, sans contre-exemple, on risque la *surgénéralisation*. Par exemple, si un programme d'IG fait l'hypothèse, lors de sa phase d'apprentissage, que le langage à apprendre est le langage universel (Σ^*, où Σ est l'alphabet du langage), aucun exemple positif n'est en mesure de le contredire, alors même qu'il a peut-être surgénéralisé.

La première chose que l'IG se doit de fournir est une définition précise de ce qu' "apprendre une langue par exemples positifs" signifie pour un programme. Le critère d'apprenabilité est théorique et formel et non pas empirique. Faisons le parallèle avec l'acquisition du langage chez les enfants. Un enfant n'est pas "programmé" pour une langue précise, il est capable d'acquérir n'importe laquelle parlée dans son environnement. De même, un programme d'IG par présentation positive doit être en mesure d'apprendre *une classe de langages formels*, c'est-à-dire d'identifier n'importe lequel de ses membres à l'aide d'exemples de séquences (phrases) lui appartenant. Les principaux critères possibles qui caractérisent la notion d'"apprenabilité d'une classe de langages" en IG (aussi appelés modèles d'apprentissage) sont "l'identification à la limite" (Gold, 1967) et "l'apprentissage PAC" (Valiant, 1984), que nous ne pouvons détailler ici.

Malheureusement, même pour la classe des langages réguliers, la plus simple dans la hiérarchie de Chomsky, ces critères sont impossibles à satisfaire : il n'existe aucun algorithme capable d'apprendre par présentation positive la classe complète des langages réguliers dans ces modèles (Gold, 1967; Kearns and Vazirani, 1994). Les recherches se sont donc orientées vers des classes plus petites, ou transverses à la hiérarchie de Chomsky, et apprenables, caractérisées notamment dans (Angluin, 1980). Les classes de langages *k*-réversibles (Angluin, 1982) entrent dans ce cadre, elles constituent le point de départ de nos expériences. Depuis, bien d'autres classes apprenables par présentation positive ont été décrites et étudiées (Garcia and Vidal, 1990; Denis et al., 2002; Kanazawa, 1998; Koshiba et al., 2000; Yokomori, 2003). Des avancées récentes dans le domaine concernent aussi l'apprenabilité de dispositifs intégrant des probabilités, comme les automates probabilistes et leurs liens avec les HHM (Thollard et al., 2000; Dupont et al., 2005). Parallèlement, des compétitions [3] ont permis de tester l'efficacité des algorithmes proposés lorsqu'ils sont confrontés à des données réelles.

[3] les plus récents étant Stamina (`http://stamina.chefbe.net`) et Zulu (`http://labh-curien.univ-st-etienne.fr/zulu`)

3.2 L'algorithme k-RI

Dans cette section, nous décrivons les algorithmes d'IG par présentation positive utilisés dans nos expériences. Ils sont destinés à apprendre un automate pour un type spécifique de chunk, à partir uniquement des différentes séquences de POS aparaissant dans ce type de chunks dans les données d'apprentissage. Les algorithmes d'IG par exemples positifs semblent adaptés à ce problème en raison du vocabulaire restreint mis en jeu (au maximum les 30 étiquettes POS) et de la relativement faible variabilité des séquences de POS pouvant décrire un même chunk.

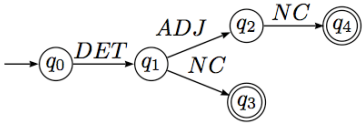
L'algorithme k-Reversible Inference (k-RI) (Angluin, 1982) a la propriété d'identifier à la limite tout langage k-réversible, pour tout $k \in \mathbb{N}$ fixé. Les langages k-réversibles sont réguliers, ils sont donc représentables par des automates finis. Un automate fini définit un langage k-réversible s'il est déterministe et si son miroir [4] est déterministe avec anticipation k. Pour $k = 0$, les langages 0-réversibles peuvent être représentés par un automate déterministe dont le miroir l'est également, l'algorithme correspondant étant appelé Zéro Réversible (ZR). Si $k_1 < k_2$, la classe des langages k_1-réversibles est strictement incluse dans celle des langages k_2-réversibles.

Soit un ensemble de séquences positives S, la première étape de k-RI est de construire PTA(S), le Prefix Tree Acceptor de S. PTA(S) est le plus petit (en nombre d'états) AFD (Automate Fini Déterministe) en forme d'arbre reconnaissant exactement le langage S. La racine de PTA(S) est son état initial. L'espace de recherche de tout algorithme d'IG partant de S est un trelli dont la borne inférieure est PTA(S) et la borne supérieure l'automate universel construit sur l'alphabet observé dans S (Dupont et al., 1994). La plupart des algorithmes d'IG suivent le même schéma : ils partent de PTA(S) pour ensuite généraliser le langage défini par fusions d'états, la connaissance de k permettant d'éviter la surgénéralisation. k-RI, détaillé ci-dessous, fonctionne selon ce principe. La fusion qu'il emploie est appelée déterministe car elle se propage récursivement à travers l'automate pour préserver son déterminisme.

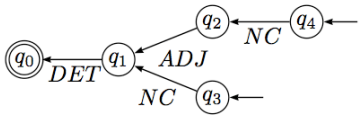
Algorithme k-RI
Entrée : S : un ensemble de séquences (positives), k : un entier naturel ;
Sortie : A : un automate k-réversible ;
début
    A := PTA(S);
    tant que non(A k-réversible) **faire**
        // soient N1 et N2 deux nœuds empêchant la k-réversibilité de A.
        Fusion_Déterministe(A, N1, N2);
    fin tant que;
    renvoyer A;
fin k-RI;

Dans la figure 1, nous illustrons le comportement de ZR (k-RI pour $k = 0$) sur les séquences de POS suivantes : $S = \{DET\ NC,\ DET\ ADJ\ NC\}$. Sur cet exemple très simple, nous voyons que ZR généralise PTA(S) pour obtenir un automate reconnaissant le langage défini par l'expression régulière : $DET\ ADJ^*\ NC$. Cette généralisation est sensée d'un point de vue linguistique. Mais si on ajoute aux exemples précédents la simple séquence NC, alors ZR mène à un automate reconnaissant le langage $\{DET|ADJ\}^*NC$, ce qui est une généralisation plus discutable.

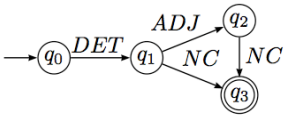
[4] l'automate miroir est obtenu en transformant les états initiaux de l'automate de départ en états finaux, ses états finaux en initiaux, et en retournant le sens de ses transitions



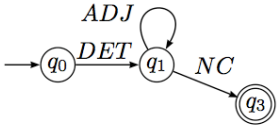
étape 1 : PTA(S)



étape 1 : miroir de PTA(S)



étape 2 : les états finaux de PTA(S) sont fusionnés (sinon le miroir n'est pas déterministe)



étape 3 : q_1 et q_2 sont fusionnés (sinon le miroir n'est pas déterministe)

Figure 1: Démonstration pas-à-pas de ZR

3.3 Résultats des expériences d'IG sur le chunking NP

Nous avons appliqué k-RI pour différentes valeurs de k ($k = 0, k = 1, k = 2$) sur les séquences d'étiquettes POS correspondant à un même chunk. Les annotations BIO sont obtenues en utilisant les automates appris comme des expressions régulières selon un parcours séquentiel de la phrase. Nous avons cherché à reconaître soit les séquences les plus longues ("Longest Match", LM) soit les séquences les plus courtes ("Shortest Match", SM). L'étiquetage en NP seuls est la tâche pour laquelle l'IG est la plus appropriée. Il est aussi évidemment possible d'apprendre un automate distinct pour chaque type de chunk. Mais l'application de plusieurs automates distincts sur une nouvelle donnée pose des problèmes de recouvrements de frontières. Nous n'utiliserons donc ces automates que dans le cadre d'une combinaison avec un modèle statistique, en section 5.

k-RI est connu pour être très sensible aux données d'entrée : une seule séquence incorrecte peut mener à de multiples fusions d'états, et donc à une surgénéralisation. C'est le cas pour notre jeu de données issues du FTB, d'où les cas aberrants et les erreurs d'étiquetage ne sont pas absents. Quelques mauvais exemples étaient cependant faciles à détecter : par exemple, des séquences d'étiquettes POS ne contenant aucune tête nominale possible peuvent être retirées sans risque. Nous avons envisagé diverses autres façons de nettoyer les données. Un nettoyage retirant toutes les séquences apparaissant moins d'une certaine proportion fixée s'est révélé le plus efficace. Cette stratégie entraîne néanmoins la suppression de séquences utiles, en raison de la faible proportion de certaines têtes (clitiques notamment).

Nos expériences ont été réalisées selon un protocole de validation croisée partitionant les données en cinq (4/5 pour l'apprentissage d'un automate, 1/5 pour le test). L'égalité requise sur les chunks est stricte, c'est-à-dire que pour être égaux, ils doivent partager exactement les même frontières. Les précision, rappel et F-mesure des chunks NP sont calculés sans prendre en compte les étiquettes O. Le tableau 1 contient diverses F1-mesures obtenues par inférence grammaticale sur les chunks NP seuls, en appliquant sur les données de test une stratégie de correspondance LM. Les versions dîtes nettoyées ont été obtenues en supprimant toute séquence de POS apparaissant strictement moins de 0.01%. Les valeurs entre parenthèses sont les nombres moyens d'états des

xp	PTA pur	PTA net.	0-RI net. (1)	1-RI net. (19)	2-RI net. (68.6)
F-mes.	51.92	88.05	26.95	72.74	88.25

Table 1: Résultats de l'IG pour le chunking NP

5 automates calculés pendant la phase d'apprentissage. Les versions PTA, dont les performances ne sont pas négligeables, peuvent être vues comme un apprentissage "par cœur", puisqu'ils n'ont donné lieu à aucune généralisation. Les automates de taille 1 correspondent à ceux reconnaissant le langage universel des étiquettes POS présentes au moins une fois dans un chunk NP. Il faut atteindre $k = 2$ pour obtenir un automate meilleur que le PTA sur des données nettoyées.

4 Apprentissage statistique pour l'annotation

Dans cette section, nous nous concentrons sur la meilleure approche statistique actuelle pour une tâche d'annotation : les Conditional Random Fields (CRF), qui se comportent très bien sur notre problème (Tellier et al., 2012). Nous rappelons aussi comment un HMM peut être "transformé" en un CRF, parce que cette transformation sera une source d'inspiration pour une des combinaisons présentées par la suite.

4.1 Conditional Random Fields et HMMs

Les CRF, introduits par (Lafferty et al., 2001) sont de la famille des modèles graphiques. Lorsque que le graphe exprimant les dépendances entre étiquettes est linéaire (ce qui est généralement le cas pour étiqueter des séquences), la distribution de probabilité d'une séquence d'annotations y connaissant une séquence observable x est définie par :

$$p(y|x) = \frac{1}{Z(x)} \prod_t \exp\Big(\sum_{k=1}^{K} \lambda_k f_k(t, y_t, y_{t-1}, x) \Big)$$

Où $Z(x)$ est un facteur de normalisation dépendant de x et les K features (ou fonctions caractéristiques) f_k des fonctions fournies par l'utilisateur. Une feature f_k est vérifiée (i.e. $f_k(t, y_t, y_{t-1}, x) = 1$) si, à la position courante t, une configuration entre x, y_t et y_{t-1} est observée (elle vaut 0 sinon). À chaque feature f_k est associé un poids λ_k. Ces poids constituent les paramètres du modèle devant être estimés au cours de l'apprentissage. Pour définir un grand nombre de features, les programmes implémentant les CRF permettent d'avoir recours à des *patrons* (ou templates) qui seront instanciés en autant de features qu'il y a de positions sur les données d'entraînement où ils peuvent s'appliquer. L'implémentation la plus efficace à l'heure actuelle des CRF linéaires est fournie par Wapiti[5], qui utilise des pénalisations pour sélectionner les features les plus pertinentes (Lavergne et al., 2010). C'est le logiciel que nous avons utilisé.

Les CRF se sont montrés efficaces sur de nombreuses tâches d'annotation, notamment l'étiquetage POS (Lafferty et al., 2001), la reconnaissance d'entités nommées (McCallum and Li, 2003), le chunking (Sha and Pereira, 2003) et même le parsing complet (Finkel et al., 2008; Tsuruoka

[5] `http://wapiti.limsi.fr/`

Feature	Type	Fenêtre
Mot	Unigram	[-2..1]
POS	Bigram	[-2..1]

chunking	Complet	NP seuls
micro	97.53	N/A
macro	90.49	N/A
F1-mesure	N/A	96.43

Table 2: Le patron de template et les résultats obtenus avec les CRF seuls pour chaque tâche

et al., 2009). Leur principal inconvénient est qu'ils apparaissent comme des "boîtes noires". Un modèle issu d'un apprentissage par CRF est simplement une liste de features pondérées pouvant avoir plusieurs millions d'éléments, ce qui le rend difficile à interpréter.

Les HMM, qui étaient parmi les meilleures méthodes d'annotation statistique avant que les CRF n'apparaissent, présentent quant à eux l'avantage d'être plus interprétables. Cependant, tout HMM peut être "transformé" en un CRF définissant la même distribution de probabilité (Sutton and McCallum, 2006; Tellier and Tommasi, 2011). Pour ce faire, pour un HMM donné, nous devons définir deux familles de features :

• les features de la forme $f(y_t, x_t)$ associant une seule étiquette y_t avec une seule entrée de même position x_t : elles valent 1 quand l'états y_t du HMM émet x_t ;

• les features de la forme $f(y_{t-1}, y_t)$ qui associent deux états y_{t-1} et y_t du HMM ; elles valent 1 quand la transition entre ces deux états est utilisée.

Si θ est une probabilité d'émission ou de transition du HMM, alors on choisit $\lambda = log(\theta)$ comme poids pour la feature correspondant dans le CRF. Le calcul de $p(y|x)$ s'écrira alors exactement de la même façon dans les deux cas. Un HMM peut ainsi être vu comme un cas particulier de CRF. Mais les CRF sont plus généraux car ils permettent d'avoir recours à d'autres features que celles utilisées dans la transformation. Cette correspondance nous a inspirés pour exploiter les CRF afin de *diagnostiquer* les automates appris par IG. Cette idée sera étudiée dans la section 5. Mais auparavant, nous présentons les résultats obtenus avec les CRF seuls sur nos données.

4.2 Résultats des expériences

Les tableaux 2 montrent les patrons de features utilisés ainsi que les résultats obtenus avec les CRF seuls sur les deux tâches de chunking. Pour ces expériences, comme en section 3.3, nous avons suivi un protocole de validation croisée à 5 plis et un critère d'égalité stricte des chunks. Pour la tâche de chunking complet, nous avons calculé les micro et macro-average, qui correspondent aux moyennes des F1-mesures des différents types de chunks, pondérées (micro) ou pas (macro) par leur proportion. Comme attendu, les CRF seuls sont très performants. Remarquons toutefois qu'ils exploitent dans leurs features à la fois des mots et des étiquettes POS présents dans les données, alors que les algorithmes d'IG n'ont accès qu'aux seuls POS.

On peut comparer ces résultats avec ceux obtenus lors de la campagne PASSAGE (Paroubek et al., 2006), même si les notions de chunks adoptées de part et d'autre diffèrent (dans PASSAGE, les adjectifs épithètes situés après un nom ne font pas partie du chunk nomimal, par exemple) et si les corpus ne sont pas les mêmes. Les meilleurs participants de la campagne PASSAGE atteignaient une micro-average de 92,7, ce qui situe tout de même la performance de nos CRF.

mot	POS	auto. NP	auto. VP	auto. PP	...	label correct	auto. NP	NP-label correct
la	DET	B	O	O	...	B-NP	B	B
dépréciation	NC	I	O	O	...	I-NP	I	I
par_rapport_au	P	O	O	B	...	B-PP	O	O
dollar	NC	B	O	I	...	I-PP	B	B
a	V	O	B	O	...	B-VP	O	O
été	VPP	O	I	O	...	I-VP	O	O
limitée	VPP	O	I	O	...	I-VP	O	O
à	P	O	O	B	...	B-PP	O	O
2,5	DET	B	O	I	...	I-PP	B	B
%	NC	I	O	I	...	I-PP	I	I

Table 3: Données enrichies par des sorties d'automates spécifiques pour chaque chunk

5 Combinaisons

Dans les sections précédentes, nous avons appliqué à la tâche de chunking une approche soit purement symbolique soit purement statistique. Dans cette section, nous allons combiner les deux approches, cette combinaison pouvant s'envisager selon deux axes distincts :

• Soit le but est la seule performance, auquel cas il faut privilégier l'apprentissage statistique. Cependant, les automates obtenus par IG offrent une vision globale (et non locale, comme c'est le cas dans les features) des relations entre les étiquettes POS d'un même chunk qui pourrait s'avérer utile dans un CRF. Nous pouvons donc chercher à intégrer les résultats de l'apprentissage symbolique en tant que ressource externe de l'apprentissage statistique.

• Soit nos fins sont plus en rapport avec la lisibilité, auquel cas nous favoriserons les automates produits par IG. Or, comme évoqué en 4.1, il est tout à fait possible de simuler la structure d'un HMM (et, similairement, d'un automate) avec les features d'un CRF. On pourrait donc évaluer la qualité des états et des transitions d'un automate en fonction des poids associés aux features qui les représentent dans un CRF, offrant ainsi par la même occasion un moyen de l'améliorer.

5.1 Les automates en tant que ressource externe

Nous nous attaquons ici aux deux types de chunking. Le premier mode de combinaison envisagé consiste à enrichir les données du CRF avec des attributs provenant de la ressource externe, à la façon de (Constant and Tellier, 2012). Dans le cas du chunking complet, nous appliquons l'IG à chaque type de chunk distinct, produisant ainsi autant d'automates qu'il y a de types de chunks selon un protocole de validation croisée à 5 plis (les PTA dans ces expériences sont donc uniquement extraits des corpus d'apprentissage). Chacun des automates de chunk fournit un étiquetage BIO indépendant, comme dans le tableau 3 (les automates sont ici supposés fournir un étiquetage parfait). Il y a donc dans nos données autant d'attributs nouveaux que de chunks.

Les tableaux de gauche dans les tables 4 donnent les patrons aboutissant aux meilleurs résultats (micro resp. macro-average resp. F-mesure) pour le chunking complet ou le chunking NP. La ligne "Automate" prend en compte la sortie de chaque automate indépendamment alors que "POS+Automates" représente la concaténation des colonnes POS et des sorties de tous les automates. Les résultats correspondants sont donnés dans les tableaux de droite. Ils montrent que les attributs provenant des automates permettent d'améliorer significativement les résultats des CRF. C'est particulièrement vrai pour la macro-average, qui donne un poids équivalent à la

F1-mesure de chaque type de chunk. Les informations issues des automates améliorent donc surtout les performances de reconnaissance des chunks rares. Dans l'expérience permettant d'obtenir la meilleure macro, les trois améliorations les plus significatives en terme de F-mesure sont : UNKNOWN (de 41.67 à 61.22), AP (de 96.78 à 97.44) et AdP (de 98.72 à 98.92).

Feature	Type	Fenêtre
Mot	Unigram	[-2..1]
Automate	Bigram	[-2..1]
POS	Bigram	[-2..1]

F-mesure	pur 1-RI LM
micro	97.66
macro	92.22

Feature	Type	Fenêtre
Mot	Unigram	[-2..1]
Automate	Unigram	[-1..1]
POS	Bigram	[-2..1]
POS+Automates	Bigram	[-1..1]

F-mesure	pur 1-RI SM
micro	97.62
macro	93.52

Feature	Type	Fenêtre
Mot	Unigram	[-2..1]
POS	Bigram	[-2..1]
Automate	Bigram	[-1..1]
POS+Automate	Bigram	[-1..1]

	pur 2-RI LM
F-measure	96.75

Table 4: Patrons et meilleure micro-average (resp. macro-average) pour le chunking complet, idem pour la F-mesure du chunking NP seul

5.2 Diagnostiquer un automate à l'aide d'un CRF

Nous voulons ici obtenir des informations sur l'automate produit par IG à l'aide des CRF, en faisant un apprentissage n'utilisant que des features interprétables relativement à lui. Les poids associés par le CRF à ces features fourniront un diagnostic fin de l'automate. Cette idée se rapproche de (Roark and Saraclar, 2004), où un CRF était appris selon la structure d'un automate pondéré pour le "corriger" grâce à l'estimation des poids. Elle en diffère toutefois car nous ne cherchons pas à obtenir un automate pondéré mais à trouver d'éventuelles modifications à effectuer sur l'automate selon le diagnostic fourni par le CRF, tout en préservant sa nature purement symbolique. Pour illustrer cette approche, nous nous concentrons sur la tâche de chunking NP seul car elle ne nécessite la prise en compte que d'un seul automate. Il peut être plus facile pour comprendre la suite de se représenter les automates finis déterministes (AFD) "à la Unitex" (http://www-igm.univ-mlv.fr/ unitex/). Ainsi, le résultat de l'algorithme ZR sur la figure 1 (l'automate final, en bas à droite) est identique à celui de la figure 2. Cette représentation a l'avantage de montrer les étiquettes POS et les transitions entre deux étiquettes POS comme deux objets différents. Pour construire un CRF à partir d'un tel automate, nous considérons surtout les sorties en termes d'étiquetage BIO que cet automate produit (partie droite de la Table 3).

Nous inspirant de la relation entre les HMM et les CRF évoquée en section 4.1, nous définissons des patrons de features qui peuvent s'interpréter relativement à l'automate :

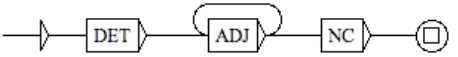


Figure 2: Automate représenté "à la Unitex"

• Un patron unigramme qui observe une étiquette POS et l'étiquette BIO prédite par l'automate à la même position, conjointement avec le label BIO correct. L'étiquette POS correspond à un (ou plusieurs) état(s) de l'automate. Si les étiquettes BIO coïncident pour un POS donné, cela signifie que l'automate a en quelque sorte raison d'être dans cet état en analysant la donnée.

• Un patron bigramme qui observe un couple d'étiquettes POS successives et le couple d'étiquettes BIO prédites par l'automate correspondant, associé au couple de labels BIO correct. Le couple de POS représente une (ou plusieurs) transition(s) de l'automate. Si les deux couples d'étiquettes BIO coïncident, cela signifie que la transition est correctement utilisée.

Il est à noter que les mots eux-mêmes ne sont pas pris en compte dans ces patrons, afin de préserver l'interprétation des features relativement à l'automate, d'où les mots sont absents.

La Table 5 est une matrice de confusion qui met en relation les étiquettes BIO prédites par un automate (EP) et les étiquettes BIO correctes (EC), pour une étiquette POS donnée (ici, l'étiquette DET d'un automate appris). On peut construire autant de tables que d'étiquettes POS présentes dans un chunk NP, chaque case de chaque table correspondant à une feature unigramme. Les cases de la Table 5 sont remplies par les poids appris par le CRF pour les features en question, les couleurs montrent comment elles s'interprètent relativement à l'automate de départ. Comme espéré, les poids sur la diagonale, qui signalent un étiquetage correct, sont plus grands que ceux en dehors, qui désignent une erreur d'étiquetage. Les features bigrammes sont un peu plus compliquées mais il est également possible d'en tirer des matrices de confusion interprétables.

EP \EC	B	I	O
B	1.66	-4.05	-0.84
I	-0.44	0.46	-2.51
O	N/A	N/A	N/A

vert : les deux sorties sont identiques.
rouge : début prématuré de chunk.
jaune : début de chunk manqué.
bleu : continuation intempestive de chunk.
cyan : arrêt prématuré de chunk.

Table 5: Une matrice de confusion colorée pour l'étiquette DET (2-RI, tableau 1)

De manière générale, le poids associé à une feature d'un CRF représente *son pouvoir discriminant*. Ces poids sont donc bien plus pertinents que de simples comptes d'occurences sur le nombre de fois qu'une feature a été satisfaite ou pas dans les données d'apprentissage. Les poids sur les diagonales peuvent ainsi être vus comme évaluant la qualité des états / transitions de l'automate, alors que les poids dans les autres cases correspondent aux gains obtenus en prenant une décision d'étiquetage non préconisée par l'automate. L'ensemble des matrices de confusion offre donc une mesure extrêmement fine et précise de la qualité de l'automate.

Le tableau 6 rappelle le meilleur résultat obtenu par IG "pure" sur le chunking NP de la section 3.3 et donne les résultats des CRF construits comme précédemment sur le meilleur automate

Expérience	baseline (IG seule)	0-RI	1-RI	2-RI
chunk	88.25	93.00	93.07	93.08

Table 6: Résultats du chunking NP avec les CRF construits sur les automates

produit par k-RI pour chaque valeur de k. Comme on pouvait s'y attendre, les CRF construits sur les automates NP sont meilleurs que les automates NP seuls, mais moins bons qu'un CRF exploitant plus d'attributs et de features. Les résultats des matrices de confusion doivent encore être examinés en détail. Nous espérons en tirer un diagnostic précis pour analyser où et pourquoi les automates prennent de bonnes ou de mauvaises décisions, et les modifier en conséquence. Les améliorations observées dans la Table 6 laissent en effet supposer qu'à de nombreuses occasions le CRF a eu raison de prendre une décision différente de celle préconisée par l'automate.

Conclusion et perspectives

Dans cet article, nous avons appliqué deux méthodes d'apprentissage automatique sur le même jeu de données et avons proposé deux façons différentes de les combiner.

Pour ce qui est de l'apprentissage symbolique seul, il est possible que d'autres algorithmes d'IG par présentation positive pourraient donner de meilleurs résultats que les nôtres, comme ceux de (Garcia and Vidal, 1990; Denis et al., 2002). Le choix d'une grande valeur de k dans certains cas peut être important, mais il s'accompagne d'une plus grande complexité de calculs[6].

Mais la partie la plus originale de notre travail concerne les combinaisons automates/CRF. Notons que ces combinaisons peuvent tout autant s'appliquer à des automates écrits à la main, généralement plus pertinents d'un point de vue linguistique que ceux obtenus par IG. Nous nous sommes concentrés ici sur des automates appris automatiquement pour montrer que, même sans ressource ni expertise linguistique, il est possible de combiner modèles symboliques et statistiques. L'intuition derrière ce travail est que ces deux types de modèles sont complémentaires, et qu'ils peuvent chacun bénéficier de l'autre. Les CRF sont basés sur un grand nombre de configurations locales pondérées. Il est théoriquement possible d'utiliser dans un CRF des features portant sur l'intégralité de la séquence x mais dans la pratique, cela est rarement fait. L'IG au contraire s'applique à un ensemble de séquences globales qu'elle est capable de généraliser. Il a déjà été observé que les CRF gagnent à recourir à des features exprimant des propriétés plus générales que de simples configurations locales (Pu et al., 2010). Notre pari était que l'IG pouvait fournir ce type de généralisation, via le premier mode de combinaison. Les résultats obtenus vont dans ce sens. Il est aussi intéressant de constater que les modèles symboliques permettent d'améliorer le traitement des cas rares, mal pris en compte par les modèles statistiques.

Les CRF construits sur des automates restent encore à étudier, notamment pour interpréter et exploiter au mieux les matrices de confusion qu'ils produisent. Certaines cases de ces matrices sont vides car Wapiti élimine les features non pertinentes de l'ensemble de départ selon un critère de pénalité. Il devrait être possible, à l'aide de ces informations, de modifier l'automate sur lequel se base le CRF en supprimant ou ajoutant des états ou des transitions pour se conformer au diagnostic fourni par une table. Une IG dirigée par des CRF reste encore à définir ! Un autre

[6] la complexité algorithmique de k-RI est $|\Sigma|^k|Q|^{k+3}$ où $|Q|$ est le nombre d'états du PTA

défi serait l'étude du lien entre les automates associés aux poids calculés par CRF que nous définissons et les plus classiques HMM ou automates probabilistes pour lesquels des algorithmes d'apprentissage existent déjà (Thollard et al., 2000).

6 Références

Abeillé, A., Clément, L., and Toussenel, F. (2003). Building a treebank for french. In Abeillé, A., editor, *Treebanks*. Kluwer, Dordrecht.

Abney, S. (1991). Parsing by chunks. In Berwick, R., Abney, R., and Tenny, C., editors, *Principle-based Parsing*. Kluwer Academic Publisher.

Angluin, D. (1980). Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135.

Angluin, D. (1982). Inference of reversible languages. *Journal of the ACM*, 29(3):741–765.

Antoine, J.-Y., Mokrane, A., and Friburger, N. (2008). Automatic rich annotation of large corpus of conversational transcribed speech: the chunking task of the epac project. In *Proceedings of LREC'2008*.

Blanc, O., Constant, M., Dister, A., and Watrin, P. (2010). Partial parsing of spontaneous spoken french. In *Proceedings of LREC'2010*.

Constant, M. and Tellier, I. (2012). Evaluating the impact of external lexical resources unto a crf-based multiword segmenter and part-of-speech tagger. In *Proceedings of LREC 2012*.

Constant, M., Tellier, I., Duchier, D., Dupont, Y., Sigogne, A., and Billot, S. (2011). Intégrer des connaissances linguistiques dans un CRF : application à l'apprentissage d'un segmenteur-étiqueteur du français. In *Actes de TALN'11*.

Crabbé, B. and Candito, M. H. (2008). Expériences d'analyse syntaxique statistique du français. In *Actes de TALN'08*.

de la Higuera, C. (2010). *Grammatical Inference: Learning Automata and Grammars*. CU Press.

Denis, F., Lemay, A., and Terlutte, A. (2002). Some language classes identifiable in the limit from positive data. In *ICGI 2002*, number 2484 in LNAI, pages 63–76. Springer Verlag.

Dupont, P., Denis, F., and Esposito, Y. (2005). Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–1371.

Dupont, P., Miclet, L., and Vidal, E. (1994). What is the search space of the regular inference. In *ICGI'94 - LNCS*, volume 862 - Grammatical Inference and Applications, pages 25–37, Heidelberg.

Finkel, J. R., Kleeman, A., and Manning, C. D. (2008). Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL'2008*, pages 959–967.

Garcia, P. and Vidal, E. (1990). Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE TPAMI*, 12(9):920–925.

Gold, E. (1967). Language identification in the limit. *Information and Control*, 10:447–474.

Kanazawa, M. (1998). *Learnable Classes of Categorial Grammars*. FoLLI. CLSI Publications.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Koshiba, T., Mäkinen, E., and Takada, Y. (2000). Inferring pure context-free languages from positive data. *Acta Cybernetica*, 14(3):469–477.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *Proceedings of ACL'2010*, pages 504–513. Association for Computational Linguistics.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields. In *Proceedings of CoNLL'2003*.

Paroubek, P., Robba, I., Vilnat, A., and C., A. (2006). Data annotations and measures in easy, the evaluation campain for parsers of french. In *Proceedings of LREC'2006*, pages 315–320.

Pu, X., Mao, Q., Wu, G., and Yuan, C. (2010). Chinese named entity recognition with the improved smoothed conditional random fields. *Research in Computing Science*, 46:90–103.

Roark, B. and Saraclar, M. (2004). Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL'2004*, pages 47–54.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213 – 220.

Sutton, C. and McCallum, A. (2006). *Introduction to Statistical Relational Learning*, chapter An Introduction to Conditional Random Fields for Relational Learning. MIT Press.

Tellier, I., Duchier, D., Eshkol, I., Courmet, A., and Martinet, M. (2012). Apprentissage automatique d'un chunker pour le français. In *Actes de TALN'12, papier court (poster)*.

Tellier, I. and Tommasi, M. (2011). Champs Markoviens Conditionnels pour l'extraction d'information. In *Modèles probabilistes pour l'accès à l'information textuelle*. Hermès.

Thollard, F., Dupont, P., and de la Higuera, C. (2000). Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. of ICML'2000*, pages 975–982.

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). Fast full parsing by linear-chain conditional random fields. In *Proceedings of EACL 2009*, pages 790–798.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Yokomori, T. (2003). Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 1.