MajecSTIC 2010
Bordeaux, France, du 13 au 15 octobre 2010

HybridTagger : un étiqueteur hybride pour le Français

Anthony Sigogne[1]

1 : Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge. 5, bd Descartes, Champs-sur-Marne, 77454 Marne-la-Vallée, France.

Contact : sigogne@univ-mlv.fr

Résumé
Dans cet article, nous présentons une approche hybride de l'étiquetage morphosyntaxique. Nous combinons les approches de désambiguisation symbolique et statistique dans un même processus d'étiquetage. De plus, ce processus utilise massivement des ressources lexicales externes qui nous permettent d'obtenir un étiquetage performant des mots inconnus.

Abstract
In this paper, we present a hybrid approach for morpho-syntactic tagging. We combine both symbolic and statistical approaches in a unique tagging process. Furthermore, this process massively uses external lexical resources that allow for obtaining an efficient tagging of unknown words.

Mots-clés : Traitement Automatique des Langues,étiqueteur morphosyntaxique probabiliste, approche hybride

Keywords: Natural Language Processing, probabilistic morpho-syntactic tagger, hybrid approach

---

1. Introduction

Dans le domaine du Traitement Automatique des Langues (TAL), l'étiquetage morphosyntaxique consiste à identifier pour chaque mot d'un texte, sa classe morphosyntaxique (Nom, Verbe, ...) à partir de son contexte et de connaissances lexicales. Les performances des étiqueteurs morphosyntaxiques ne cessent d'augmenter depuis quelques années avec, pour la plupart, un taux avoisinant les 98% de mots correctement étiquetés.
Ceux qui atteignent ces résultats suivent une approche statistique basée sur un modèle probabiliste. Les étiqueteurs les plus simples intègrent un modèle génératif de Markov caché (HMM) [2, 14]. D'autres étiqueteurs plus récents utilisent des modèles discriminatifs comme Maximum d'entropie (ME) [5, 13, 15], Champs conditionnels aléatoires (CRF) [8] ou encore Séparateurs à vaste marge (SVM) [6].
Une autre approche dite symbolique consiste à désambiguiser automatiquement les étiquettes morphosyntaxiques par des règles écrites manuellement par des linguistes. On pourra citer notamment l'étiqueteur EngCG [7, 16] et Elag [9].
Dans cet article, nous décrivons une approche *hybride* de l'analyse morphosyntaxique en essayant de trouver un compromis entre efficacité computationnelle et qualité d'étiquetage. Elle consiste à combiner les avantages des approches statistiques et symboliques à travers différents modules de désambiguisation. Ces modules, couplés à un modèle probabiliste, conduisent à la création d'un étiqueteur appelé *HybridTagger*. Nous avons choisi d'utiliser le modèle de Markov caché qui a les avantages d'être léger et performant.
Dans un premier temps, dans la section 2, nous décrivons la représentation des textes puis l'architecture générale de l'étiqueteur, en détaillant succintement le modèle de Markov caché. Dans la section 3, nous parlons des divers modules statistiques et symboliques de désambiguisation. Ensuite, dans la section 4, nous évaluons et discutons des performances de notre étiqueteur à travers ces différents modules, suivi d'une évaluation comparative avec d'autres étiqueteurs de référence. Nous terminons sur une conclusion et des perspectives pour HybridTagger.

2. Représentation des textes et architecture

2.1. Représentation des textes

Dans le cadre de l'analyse morphosyntaxique, les ambiguités inhérentes à la langue naturelle sont généralement représentées par un automate acyclique où chaque token [1] d'une phrase est associé à un ensemble possible d'étiquettes morphosyntaxiques. On considérera que, dans notre cas, les tokens sont des mots simples ou des mots composés. [2] Contrairement aux approches traditionnelles où les étiquettes proviennent du lexique issu d'un corpus d'apprentissage annoté, nous avons utilisé les étiquettes présentes dans des dictionnaires électroniques écrits par des linguistes. Ces dictionnaires sont de plusieurs types, généraliste pour le *Delaf* et le *Delacf* [4] (respectivement dictionnaire des formes fléchies des mots simples ayant 683824 entrées fléchies et des mots composés ayant 108436 entrées fléchies), toponymique pour les dictionnaires *Prolex* [10] (12317 entrées fléchies). La composition des étiquettes de chaque entrée de ces dictionnaires est très fine puisque qu'elles contiennent des informations sur la catégorie morphosyntaxique, des traits morphologiques (masculin, pluriel,...), le lemme et d'éventuels traits syntaxiques et sémantiques. Par exemple, on pourra citer l'entrée *quels,quel.DET+Dadj*: *mp* indiquant que le mot *quels* est un déterminant *DET*, masculin pluriel *mp*, issu du lemme *quel* et de trait syntaxique *déterminant adjectival Dadj*. Si le mot est un composé, sa structure interne en terme d'étiquettes est également précisée. Par exemple, l'entrée *facultés mentales,faculté mentale.N+NA*: *fp* désigne un nom composé de structure interne *nom* suivi de *adjectif*, indiquée par l'étiquette *N+NA*. La Figure 1 illustre l'automate ambigü de la phrase *Il a des facultés mentales impressionnantes* après application des dictionnaires.
L'utilisation de dictionnaires nous permet d'avoir une plus large couverture des mots qui pourraient apparaître dans les textes à analyser. Nous pouvons généralement assigner des étiquettes aux mots inconnus, les mots qui ne sont pas présents dans le corpus d'apprentissage. Ces observations sont également décrites dans [5]. Ces derniers montrent, dans le cadre du français, que des ressources lexicales externes au corpus d'apprentissage réduisent significativement le nombre d'erreurs d'étiquetage.

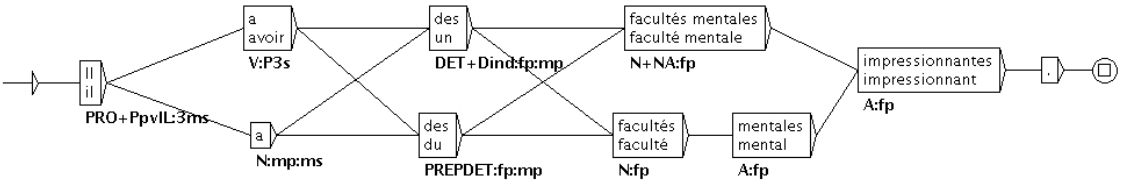


FIGURE 1 – Automate de la phrase *Il a des facultés mentales impressionnantes.*

2.2. Architecture de l'étiqueteur

L'architecture de notre étiqueteur, illustrée par la Figure 2, est formée de trois étapes successives. La tokenisation et l'analyse lexicale ambigüe ont pour but de segmenter le texte brut en tokens et d'assigner ensuite pour chacun d'eux les étiquettes possibles à partir des dictionnaires. Cela conduit à la création de l'automate ambigü de la phrase. Cette étape est réalisée automatiquement avec le logiciel de traitements linguistiques *Unitex* [3] [11]. Ensuite, nous appliquons différents algorithmes d'élagage statistiques et symboliques sur cet automate afin de supprimer des ambiguités lexicales. La dernière étape consiste à effectuer une linéarisation stochastique de l'automate dans le but de trouver la meilleure séquence d'étiquettes pour la phrase en entrée. L'étiquetage produit en sortie de l'étiqueteur associe à chaque mot du texte une étiquette morphosyntaxique composée d'une catégorie grammaticale et parfois de traits morphologiques.
Par exemple, la linéarisation de l'automate illustré Figure 1 doit nous donner :

1. Les tokens sont les unités délimitées par la phase de segmentation en mots d'une phrase.
2. Suite de mots simples dont le sens global ne peut être déduit par l'addition des sens de chaque mot.
3. Le logiciel Unitex est disponible à l'adresse : http: //www-igm.univ-mlv.fr/~unitex/

Il	a	des	facultés_mentales	impressionnantes	.
PRO: 3ms	V: P3s	DET: fp	N: fp	A: fp	PONCT

La linéarisation stochastique consiste à déterminer quelle est la séquence d'étiquettes $\overrightarrow{y_\star}$ qui maximise la probabilité d'avoir une séquence d'étiquettes $\overrightarrow{y}$ étant donnée une séquence de tokens $\overrightarrow{x}$, parmi les séquences d'étiquettes possibles Y. Pour estimer ces probabilités, nous utilisons un modèle probabiliste de Markov caché de second ordre. Ce modèle, dit génératif, calcule la probabilité jointe de $\overrightarrow{y}$ et $\overrightarrow{x}$. Les formules mathématiques liées à ce modèle sont les suivantes :

$$\overrightarrow{y_\star} = arg\ max_{\overrightarrow{y}\in Y} P(\overrightarrow{y}|\overrightarrow{x}) \simeq arg\ max_{\overrightarrow{y}\in Y} P(\overrightarrow{y}, \overrightarrow{x}) \quad (1)$$

$$P(\overrightarrow{y}, \overrightarrow{x}) = p(y_1|x_1) \prod_{i=2}^{n} p(y_i|x_i).p(y_i|y_{i-2}y_{i-1}) \quad (2)$$

L'apprentissage du modèle probabiliste est effectué à partir d'un corpus d'apprentissage. Afin de traiter les mots inconnus lors de l'étiquetage, on extrait également de ce corpus des données basées sur les suffixes de la même façon que dans la plupart des étiqueteurs [2, 12, 14]. La recherche de la meilleure séquence dans l'automate du texte est résolue par un algorithme de programmation dynamique de Viterbi qui a une complexité en $O(n^2)$ avec n le nombre de transitions de l'automate.

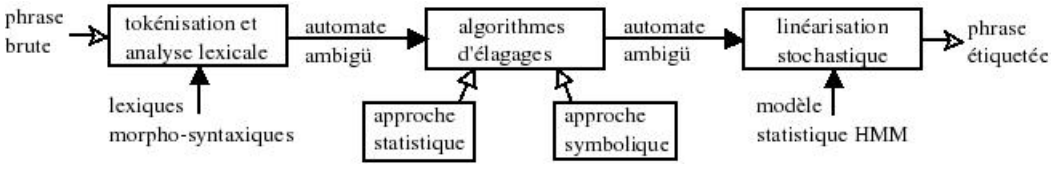


FIGURE 2 – Architecture de l'étiqueteur

3. Algorithmes d'élagages

Certaines ambiguités de l'automate peuvent être supprimées en appliquant des prétraitements efficaces. Par exemple, l'étiquette *nom* du mot *bleu* est impossible avec un nom comme contexte gauche. En réduisant les possibilités d'étiquettes pour un mot donné, nous réduisons également les possibilités d'erreurs provoquées par l'algorithme de Viterbi. Nous avons donc appliqué une chaîne de traitements d'élagage sur l'automate ambigü initial qui vont s'aider des contextes lexicaux. Cette chaîne est composée d'un module statistique appelé *LearningErrors* et d'un module symbolique *Elag*. [4]

3.1. Règles de Brill, LearningErrors

Le module LearningErrors nous permet d'élaguer des chemins présents dans toute zone de l'automate. Il est dérivé de l'étiqueteur de [3] basé sur des règles de transformation. Le fonctionnement de cet étiqueteur consiste, en premier lieu, à effectuer un étiquetage simple du corpus d'apprentissage brut. Puis, l'étiquetage du corpus généré est comparé à celui du corpus original. D'après les différences observées, des règles de transformation sont calculées pour les n mots provoquant le plus d'erreurs d'étiquetage. Les règles créées permettent de corriger les erreurs d'un texte étiqueté. Ces règles sont de deux types. La première, dite lexicale, associe un mot à une unique étiquette. Si ce mot a peu d'étiquettes assignables possibles et qu'une de ses étiquettes est majoritaire pour ce mot (en terme de distribution dans le corpus) alors le mot sera toujours associé à cette unique étiquette. Par exemple, le mot *aussi* aura toujours l'étiquette *adverbe*, même s'il peut

4. L'ordre d'application des modules peut être inverse dans la chaîne.

(a) Automate ambigü de *mange ce que* (b) Automate après application de la règle

FIGURE 3 – Application d'une règle contextuelle sur l'automate de *mange ce que*

obtenir l'étiquette *conjonction de subordination* très rare. Le deuxième type de règle est la règle contextuelle. Selon un contexte particulier, un mot sera toujours associé à une étiquette particulière. Par exemple, le mot *ce* est associé à l'étiquette *pronom* si le contexte droit est une étiquette *pronom*. Une règle contextuelle peut contenir des contextes gauches et droits.
L'étiqueteur de Brill utilise ces règles de manière à transformer un étiquetage initial. Ce principe diffère du module LearningErrors car nous utilisons l'ensemble des règles générées, non pas pour corriger, mais pour élaguer l'automate de la phrase. Les règles lexicales et contextuelles sont appliquées automatiquement et suppriment toutes les hypothèses d'étiquettes, excepté celle précisée dans la règle. En revanche, si plusieurs règles contextuelles sont possibles pour un mot donné de l'automate alors aucun élagage n'est effectué pour ce mot. La Figure 3 illustre le résultat de l'application de la règle contextuelle du mot *ce*, décrite précédemment, sur l'automate de la phrase *Je mange ce que je veux* (représentée partiellement).

3.2. Un élagueur symbolique, Elag

Les modules d'élagage précédents étaient basés sur des statistiques ou sur des heuristiques. Si nous voulons être précis sur certains points de grammaire, il faut absolumment utiliser des ressources externes créées par des linguistes. Nous avons donc introduit un système d'élagage symbolique appelé *Elag* [5] [9]. Elag est un système qui applique un ensemble de grammaires sur un automate ambigü. Elle sont écrites manuellement et définissent des règles contextuelles et lexicales. Ces règles fonctionnent sensiblement de la même façon que dans le module LearningErrors. Cependant le formalisme est quelque peu différent puisqu'une règle est représentée par un automate, à la manière de [7]. Chaque catégorie grammaticale possède un ensemble de règles associées. La Figure 4 illustre l'automate d'une règle Elag. Après l'adverbe *si*, on ne peut trouver qu'un adjectif, un adverbe, un verbe au participe passé ou une ponctuation.
La langue d'évaluation de notre étiqueteur étant le français, nous avons pris en compte la totalité des 45 règles définies dans les grammaires d'Elag originelles ainsi que 6 autres créées au cours du développement de l'étiqueteur.

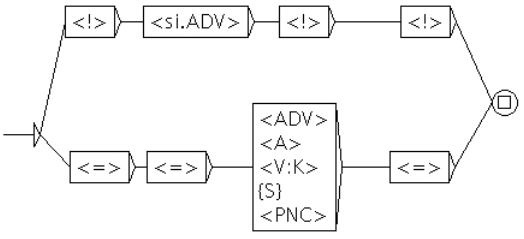


FIGURE 4 – Règle Elag concernant l'adverbe *si*

4. Évaluations

L'étiqueteur présenté dans cet article a été implémenté en partie en langage C pour la linéarisation stochastique et en langage Python pour le module LearningErrors. Il est en cours d'intégration à

5. Le système Elag est disponible dans la plateforme linguistique Unitex (écrit en langage C).

Unitex. La machine utilisée pour les évaluations est un Athlon 1.9 Ghz disposant de 2Go de RAM. Le corpus annoté du français employé pour nos expérimentations est le French TreeBank ou FTB [1]. Ce corpus étant de petite taille (21062 phrases), nous avons effectué nos évaluations selon la méthode dite de validation croisée. Cette méthode consiste à découper le corpus en p parties égales puis on effectue l'apprentissage sur $p - 1$ parties et l'étiquetage sur la partie restante. On peut itérer p fois ce processus. Cela permet donc de calculer un taux moyen d'erreurs sur un échantillon aussi grand que le corpus initial. Dans notre cas, nous avons choisi p (fixé à 5) proportionnellement à la faible taille du corpus.
Nous avons considéré que les mots simples internes aux mots composés du corpus sont fusionnés pour donner un unique mot. Par exemple, le nom composé *facultés mentales* sera représenté par l'étiquette lexicale *facultés_mentales,faculté_mentale.N*. Nous faisons cette opération dans un soucis de comparaison aux autres étiqueteurs qui, pour la plupart, ne traitent qu'avec des mots simples. La Table 1 reporte les caractéristiques de chaque ensemble de données.

TABLE 1 – Jeux de données (partitions du FTB)

Parties (p)	# de phrases	# de mots	# de mots inconnus
1	4212	89263	7206 (8,1%)
2	4212	89748	5119 (5,7%)
3	4212	81822	5405 (6,6%)
4	4212	86120	4934 (5,7%)
5	4214	79960	5002 (6,2%)

Nous avons effectué nos évaluations selon deux jeux d'étiquettes différents issus des lexiques utilisés. Le simple jeu est composé de 12 étiquettes, chacune contenant uniquement la catégorie grammaticale (*PRO, N, V*,...). Le jeu complexe, quant à lui, contient 94 étiquettes qui sont des étiquettes simples étoffées de traits morphologiques. L'étiquette *N* est par exemple divisée en plusieurs étiquettes enrichies *N*: *fs* pour le nom féminin singulier, *N*: *mp* pour le nom masculin pluriel, etc. De plus, lorsqu'on utilise l'étiqueteur avec ce jeu complexe, on peut également évaluer l'étiquetage selon un jeu d'étiquettes plus simple. Si l'étiquette *N*: *ms* est donnée à un mot alors on peut évaluer le fait que le mot ait cette étiquette complète mais aussi la partie grammaticale *N* uniquement. Cette sous-spécialisation des étiquettes sera notée *simple(comp.)* lors des évaluations. La plupart des expérimentations portent sur trois critères de performance majeurs : pourcentage de mots connus ayant une étiquette morphosyntaxique correcte (équivalente à celle du corpus d'évaluation pour le même mot), pourcentage de mots inconnus en suivant le même principe et enfin le pourcentage global des mots connus et inconnus.

Etiquetage basique : Les premières évaluations concernent les algorithmes *baseline* et *basic*. L'algorithme *baseline* consiste à assigner l'étiquette la plus fréquente à chaque mot des phrases du corpus d'évaluation. Les fréquences des étiquettes sont estimées à partir du corpus d'apprentissage. Le lexique provient donc du corpus et non de dictionnaires dans cet algorithme. Afin de donner une étiquette aux mots inconnus, nous appliquons une analyse basée sur les suffixes [2, 12, 14]. Quant à l'algorithme *basic*, il correspond à notre étiqueteur appliquant uniquement la linéarisation stochastique sur l'automate sans effectuer de traitements d'élagage. Nous avons fait deux expérimentations différentes pour *basic*. Une première expérience, *basic(dico)*, consiste à créer l'automate avec un lexique provenant de dictionnaires électroniques et une deuxième, *basic(corp)*, dont le lexique est issu du corpus d'apprentissage. Il s'agit de tester l'impact de l'utilisation de ressources lexicales externes dans un étiqueteur morphosyntaxique. Les résultats de ces expériences sont donnés dans la Table 2.
L'algorithme *baseline* malgré son principe très simpliste donne des performances élevées avec un jeu d'étiquettes simple pour les mots connus. En revanche quel que soit le jeu d'étiquettes, les mots inconnus sont mal étiquetés (en dessous de 80%). De plus, les performances sont moins bonnes que les autres deux algorithmes. *basic(corp)* étiquette mieux les mots connus que *basic(dico)* pour

TABLE 2 – Performances des algorithmes Baseline et Basic

Algorithme	jeu d'étiquettes	mots connus	mots inconnus	global
baseline	simple	96,12	77,17	94,88
basic(corp)	simple	**96,81**	77,17	95,53
basic(dico)	simple	96,67	**92,74**	**96,41**
baseline	complexe	91,41	61,06	89,44
basic(corp)	complexe	**94.26**	61,06	92,10
basic(dico)	complexe	93,78	**82,27**	**93,03**

les deux jeux d'étiquettes. Les mots inconnus ont en général une ou plusieurs étiquettes possibles dans les dictionnaires, ce qui justifie les performances d'étiquetage des mots inconnus, *basic(dico)* les étiquetant mieux que *basic(corp)*. En globalité, *basic(dico)* est meilleur que les deux autres algorithmes. Dans la suite des évaluations, nous parlerons de *basic* pour faire référence à l'algorithme *basic(dico)*.

Evaluation de l'étiqueteur : Nous allons maintenant évaluer l'étiqueteur complet en intégrant la chaîne de traitements combinant les différents modules d'élagage. La chaîne qui donne les meilleures performances quel que soit le jeu d'étiquettes suit l'ordre suivant, *LearningErrors* puis *Elag*. Les performances optimales de l'étiqueteur HybridTagger basé sur cette chaîne sont montrées dans la Table 3. On voit que l'étiqueteur hybride améliore les résultats, quel que soit le jeu d'étiquettes, avec des gains absolus d'environ 0,7% pour un jeu complexe et 0,5% pour un jeu simple, par rapport à l'algorithme *basic*.

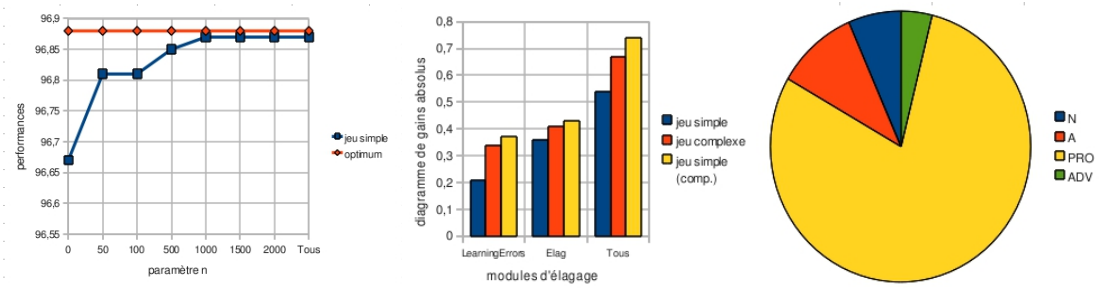
TABLE 3 – Performances optimales de l'étiqueteur

Algorithme	jeu d'étiquettes	mots connus	mots inconnus	global
basic	simple	96,67	92,74	96,41
optimal	simple	**97,15**	**92,80**	**96,95**
basic	complexe	93,78	82,27	93,03
basic	simple(comp.)	95,47	**83,12**	94,67
optimal	complexe	**94,30**	**83,02**	**93,70**
optimal	simple(comp.)	**95,90**	82,99	**95,41**

Nous allons maintenant évaluer les performances de chaque module d'élagage lorsqu'il est intégré seul à l'étiqueteur. La Figure 5(b) nous donne les résultats pour les divers jeux d'étiquettes. On peut tout d'abord observer que les modules améliorent plus l'étiquetage pour un jeu complexe que pour un jeu simple. Ceci s'explique par le plus grand nombre d'erreurs d'étiquetage à corriger dans le cadre de l'étiquetage basé sur un jeu complexe. Globalement, les performances sont équivalentes pour chaque module d'élagage selon un jeu complexe (gain absolu d'environ 0,4%). En revanche, Elag a un impact deux fois plus fort que LearningErrors selon le jeu simple. La Figure 5(c) illustre la répartition des corrections effectuées par les modules selon les catégories grammaticales. [6] On peut observer que les pronoms sont majoritaires (environ 75% des corrections), ceci à cause de leur grande ambiguité.

Evaluation de LearningErrors : Afin d'évaluer l'impact des règles contextuelles et lexicales du module LearningErrors, nous avons fait varier le paramètre n. C'est-à-dire le nombre de mots qui provoquent le plus d'erreurs d'étiquetage et dont on calcule des règles lexicales et contextuelles applicables sur l'automate (cf. 3.1). La Figure 5(a) illustre les performances d'étiquetage obtenues

6. Les catégories grammaticales non représentées ont subi un impact marginal.



(a) LearningErrors (b) Modules d'élagage (c) Répartition des corrections

FIGURE 5 – Evolution de l'étiquetage selon les modules d'élagage

pour un jeu d'étiquettes simple [7] en fonction de n. On voit que l'application de règles issues des 50 premiers mots uniquement nous rapprochent de l'optimum très rapidement. Le nombre de règles associées est de seulement 13 en moyenne. L'optimum est atteint avec un paramètre n de 1000 et les performances sont d'environ 0,21% meilleures que l'étiquetage *basic*. Puis, nous obtenons un plateau lors de la prise en compte des mots restants, un paramètre n plus grand n'influe donc plus sur les performances d'étiquetage. Ces différentes observations se retrouvent dans l'étiqueteur de Brill où seules les 82 premières règles [8] de transformation sont nécessaires afin d'obtenir des performances d'étiquetage proche de l'optimum. Cependant certaines erreurs d'étiquetage minoritaires dans les corpus d'évaluation peuvent être de plus grande ampleur dans un texte particulier. Ce qui veut dire qu'une règle peu utilisée lors des évaluations pourrait avoir une plus grande importance ultérieurement. Nous incluons donc dans la phase d'étiquetage l'ensemble des règles générées par LearningErrors durant la phase d'apprentissage avec le paramètre n maximum (en moyenne 950 règles lexicales et 19 contextuelles), ceci afin d'obtenir un module robuste dans toutes les situations.

Évaluation comparative : Nous avons comparé les performances de notre étiqueteur HybridTagger, à d'autres étiqueteurs disponibles, *TreeTagger* et *TnT*. TreeTagger [12] [9] est basé sur les arbres de décisions. Il est parmi les plus utilisés car il est très rapide et donne de bonnes performances avec 96,36% pour l'anglais sur le Penn TreeBank. Quant à TnT [2] [10], il est basé sur les HMM comme HybridTagger. Il donne des performances de 96,7% sur le Penn Treebank. La Table 4 reporte les performances des divers étiqueteurs sur le FTB.

TABLE 4 – Performances comparatives de différents étiqueteurs

Étiqueteur	jeu d'étiquettes	mots connus	mots inconnus	global	durée (min.)
HybridTagger	simple	97,15	**92,80**	**96,95**	1'10
TreeTagger	simple	**97,57**	87,27	96,89	0'22
TnT	simple	97,47	88,11	96,86	**0'15**
HybridTagger	complexe	94,30	**83,02**	93,70	1'15
TreeTagger	complexe	95,85	70,34	94,19	0'24
TnT	complexe	**95,99**	74,65	**94,61**	**0'20**

Les premières remarques que l'on peut faire concernent les mots inconnus qui sont globalement bien étiquetés par HybridTagger quel que soit le jeu d'étiquettes. De plus, ils sont mieux étiquetés

7. La courbe de performances a une croissance équivalente pour un jeu complexe.
8. Brill indique un nombre total de 447 règles de transformation.
9. Disponible à l'adresse http: //www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
10. Disponible à l'adresse http: //www.coli.uni-saarland.de/~thorsten/tnt/

par HybridTagger que par les autres étiqueteurs mais on observe également une baisse de performances pour tous les étiqueteurs dans le cadre du jeu complexe par rapport au jeu simple. Cette baisse est moins marquée pour HybridTagger avec une perte absolue d'environ 10% contre 15% et 17% pour les autres. Ceci confirme l'intérêt de l'utilisation de ressources lexicales externes pour l'étiquetage fiable de phrases contenant des mots inconnus. En revanche, concernant les mots connus, TnT et TreeTagger obtiennent des performances surpassant celles de HybridTagger, notamment pour le jeu complexe.

5. Conclusion et perspectives

À travers l'étiqueteur HybridTagger, nous avons décrit une approche hybride de l'étiquetage morphosyntaxique. La combinaison de méthodes statistiques et symboliques permet d'obtenir un étiqueteur performant sur le français quelle que soit la complexité du jeu d'étiquettes. Nous obtenons de bons résultats dans le cadre de l'étiquetage des mots inconnus, qui est en général le point négatif des étiqueteurs. On voit ici la pertinence de l'utilisation de ressources lexicales externes que sont les dictionnaires de mots simples et de mots composés. Dans un futur proche, nous prévoyons d'intégrer un modèle discriminatif, plus performant, comme CRF qui est une généralisation du modèle HMM. Nous évaluerons également notre étiqueteur sur d'autres langues comme l'anglais ou l'espagnol car nous disposons de ressources lexicales pour ces langues.

Bibliographie

1. A. Abeillé, L. Clément, et F. Toussenel. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*, Kluwer, Dordrecht, 2003.
2. T. Brants. Tnt - a statistical part-of-speech tagger. In *ANLP 2000*, pages 224–231, Seattle, 2000.
3. E. Brill. Transformation-based error-driven learning and natural language processing : A case study in part-of-speech tagging. In *ACL 1995*, pages 543–565, Cambridge, 1995.
4. B. Courtois et M. Silberztein. Dictionnaires électroniques du français. présentation. In Larousse, editor, *Langue Française, 87*, 1990.
5. P. Denis et B. Sagot. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art pos tagging with less human effort. In *PACLIC 2009*, Hong Kong, 2009.
6. J. Giménez et L. Màrquez. Svmtool : A general pos tagger generator based on support vector machines. In *Language Resources and Evaluation*, 2004.
7. F. Karlsson, A. Voutilainen, J. Heikkilä, et A. Antilla. Constraint grammar : A language-independent system for parsing unrestricted text. In Mouton de Gruyter, editor, *Natural Language Processing, No 4*, Berlin, 1995.
8. J.D. Lafferty, A. McCallum, et F.C.N Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289, Williamstown, USA, 2001.
9. E. Laporte et A. Monceaux. Elimination of lexical ambiguities by grammars. the elag system. In *Lingvisticae Investigationes XXII*, pages 341–367, 1999.
10. D. Maurel et O. Piton. Un dictionnaire de noms propres pour intex : les noms propres géographiques. In *Linguisticae Investigationes XXII*, pages 279–289, 1999.
11. S. Paumier. Unitex manual. In *http ://igm.univ-mlv.fr/ unitex/UnitexManual2.0.pdf*, Université Paris-Est, 2008.
12. H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *ICNMLP1995*, Manchester, 1995.
13. L. Shen, G. Satta, et A.K. Joshi. Guided learning for bidirectional sequence classification. In *ACL 2007*, pages 760–767, Prague, 2007.
14. S.M. Thede et M.P. Harper. A second-order hidden markov model for part-of-speech tagging. In *ACL 1999*, pages 175–182, College Park, Maryland, 1999.
15. K. Toutanova, D. Klein, C.D. Manning, et Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003*, pages 173–180, Edmonton, 2003.
16. A. Voutilainen. A syntax-based part-of-speech analyser. In *EACL 1995*, pages 157–164, University College Dublin, Belfield, 1995.