Grammaires locales étendues approches pour l'extraction del'information

Dans ce chapitre nous étudions quelques approches pour l'extraction d'information à l'aide des grammaires locales étendues. Nous abordons plusieurs problématiques : la désambiguïsation des catégories grammaticales à l'aide de méthodes fondées sur l'apprentissage automatique, l'analyse sémantique prédicat–argument à l'aide d'un moteur d'inférence logique, la recherche adaptative de motifs dans un dictionnaire électronique et la reconnaissance des entités nommées bruités. Pour chacun des sujets, nous allons d'abord contextualiser la problématique et ensuite étudier comment utiliser le formalisme des grammaires locales étendues pour apporter des alternatives de traitement.

7.1 Désambiguïsation des catégories grammaticales

Rappelons qu'une grammaire est ambiguë (cf. définition 2.13) s'il existe plus d'un arbre de dérivation (cf. définition 2.12) pour une phrase du langage. Dans le même sens, l'automate qui modélise une grammaire locale est ambigu lorsque deux chemins réussis ont la même étiquette d'entrée.

(a) Reconnaisance d'un nom **(b)** Reconnaissance d'un adjectif

Graphe 7.1 – Grammaire locale ambiguë pour le cas d'analyse nom ⟨N⟩ ou adjectif ⟨A⟩

Dans certains cas, pour lever l'ambiguïté de l'automate, il est possible d'effecteur à l'avance sa déterminisation [1] afin d'obtenir un automate qui par définition n'est pas ambigu. Cependant, dans la pratique, comme dans les cas du graphe 7.1, pouvant avoir des ambiguïtés issues des catégories grammaticales [2], réaliser une telle opération n'est pas possible. En effet, à l'inverse des langages artificiels, les langages naturels sont très expressifs et bien connus pour avoir beaucoup d'ambiguïtés et il n'est pas alors toujours possible de les supprimer. Comme alternative, plusieurs approches peuvent être prises en compte. Nous considérons brièvement celles fondées sur la construction d'un automate du texte pour lever l'ambiguïté grammaticale, pour ensuite nous consacrer à certaines techniques qui peuvent être mise en place pour traiter l'ambiguïté (deux chemins réussis qui ont la même étiquette d'entrée) dans la conception d'une grammaire locale. Finalement, nous proposons une approche fondée sur la construction d'une grammaire locale étendue pour réduire l'ambiguïté des catégories grammaticales.

7.1.1 Construction d'un automate du texte

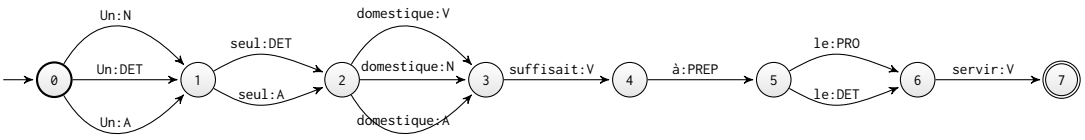


Figure 7.1 – Ambiguïté dans l'analyse de la phrase *Un seul domestique suffisait à le servir*

Prenons comme exemple la figure 7.1, elle modélise toutes les interprétations grammaticales de la phrase « *Un seul domestique suffisait à le servir* » au moyen d'un automate, aussi appelé *automate du texte*. Les codes grammaticaux possibles d'une unité lexicale, par exemple Un ⇐ {A, DET, N}, étiquettent chacun une transition partant de la même origine : $0 \xrightarrow{\text{Un:A}} 1$, $0 \xrightarrow{\text{Un:DET}} 1$, $0 \xrightarrow{\text{Un:N}} 1$ pour indiquer que le mot « Un » peut être un adjectif (A), un déterminant (DET) ou un nom (N). La question est alors comment, à partir de l'automate du texte, lever l'ambiguïté, plusieurs approches peuvent être utilisées :

1. Pour une discussion sur la déterminisation des grammaires locales, voir Sastre (2011, pp. 199–204).
2. Prenons comme exemple le mot *domestique*, avec les entrée lexicales **domestique**,.A+z1:ms:fs ; **domestique**,.N+z1:ms:fs et **domestique**,domestiquer.V+P1s:P3s:S1s:S3s:Y2s:ms:fs, le graphe 7.1 reconnaît *domestique* à la fois comme un nom (N) et comme un adjectif (A)

- Modifier manuellement l'automate du texte à l'aide d'un éditeur graphique comme celui fourni par NooJ (Silberztein, 2003, p. 156) ou par Unitex (Paumier, 2016, p. 191),
- Appliquer des méthodes fondées sur la construction de règles symboliques comme celles des grammaires de contraintes (Karlsson et al., 1995) ou de grammaires de levée d'ambiguïté (elag, Laporte et Monceaux, 1999).
- Utiliser une stratégie fondée sur l'apprentissage statistique afin de linéariser l'automate et d'obtenir un seul chemin, c'est-à-dire, une seule interprétation du texte, comme celles qui s'appuient sur des modèles de Markov cachés (hmm, Kupiec, 1992), des champs aléatoires conditionnels (crf, Lafferty et al., 2001), des machines à vecteurs de support (svm, Giménez et Marquez, 2004) ou des réseaux de neurones (Andor et al., 2016),
- Faire appel à des méthodes capables de combiner les approches de désambiguïsation symboliques et statistiques dans un même processus, tel qu'est le cas dans l'étiqueteur hybride pour le français HybridTagger (Sigogne, 2010).

Définition 7.1 (Entrée ambiguë)**.** Une séquence en entrée d'une grammaire locale est ambiguë si elle est étiquetée par deux chemins réussis, en d'autres termes, lorsque la séquence est reconnue par plus d'un chemin.

Définition 7.2 (Sortie ambiguë)**.** Étant donné une entrée ambiguë, l'ensemble des sorties produit par une grammaire locale ambiguë, est dénommée **sortie ambiguë** de la grammaire.

Considérons l'exemple 7.3 ci-après.

Exemple 7.3. Analysons les séquences (42) et (43) suivantes :

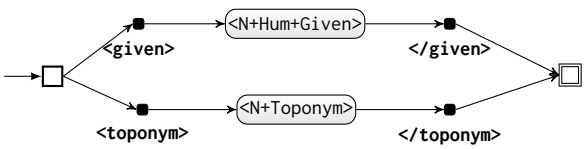
(42) *The role of Virginia Woolf was played by Nicole Kidman*

(43) *The state of Virginia is divided into 95 counties*

en tenant compte des entrées lexicales :

```
Virginia,.N+Hum+Given:fs
Virginia,.N+Toponym+Region
```

ainsi que de la grammaire locale définie par le graphe 7.2 :



Graphe 7.2 – Grammaire locale ambiguë

L'analyse de (42) produira deux sorties (séquences en noire) différentes :

(44) *The role of* ⟨*given*⟩ *Virginia*⟨/*given*⟩ *Woolf was played by Nicole Kidman*

(45) *The role of* ⟨*toponym*⟩ *Virginia*⟨/*toponym*⟩ *Woolf was played by Nicole Kidman*

il en est de même pour l'analyse de (43) :

(46) *The state of* ⟨*given*⟩ *Virginia*⟨/*given*⟩ *is divided into 95 counties*

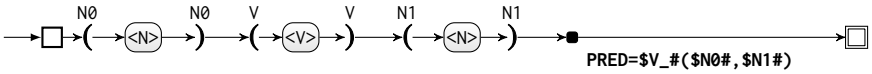
(47) *The state of* ⟨*toponym*⟩ *Virginia*⟨/*toponym*⟩ *is divided into 95 counties*

Étant donné que la grammaire locale de l'exemple 7.3 produit deux analyses pour (42) et (43), elle est ambiguë. En outre, (44) et (45) ainsi que (46) et (47) sont des sorties ambiguës de la grammaire.

Remarquons qu'une sortie ambiguë d'une grammaire locale est produite lorsqu'une séquence en entrée est acceptée par plus d'un chemin de la grammaire. Observons également que pour qu'une grammaire locale soit ambiguë, il suffit qu'elle produise au moins une sortie ambiguë. Dans l'exemple 7.3, la sortie constitue par (44) et (45) suffit à rendre la grammaire ambiguë.

7.2 Analyse sémantique prédicat–argument

Considérons le graphe d'exemple 7.3 tiré de Silberztein (2003, p. 197), ce graphe est utilisé pour reconnaître des séquences ⟨N_0⟩ ⟨V⟩ ⟨N_1⟩ et produire en sortie une analyse sémantique exprimée dans un formalisme du type prédicat–argument telle que celui utilisé par Prolog (Clocksin et Mellish, 2003), un langage de programmation logique.



Graphe 7.3 – Analyse prédicat–argument passif

Étudions la phrase :

(48) *John sees Mary*

Cette phrase décrit une action « *see* » impliquant deux personnes : *John* et *Mary*. Tandis que l'action est définie autour d'un noyau verbal ⟨V⟩, appelé **prédicat**, les deux noms, ⟨N_0⟩ et ⟨N_1⟩, définissent les participants de l'action et sont dénommés ces **arguments**. La représentation de cette relation prédicat–argument en Prolog est donnée par l'expression `see(John,Mary)`. Le graphe 7.3 est capable de produire des expressions de type prédicat–argument. En particulier, il est en mesure de reconnaître une phrase en entrée, telle que (48), et de produire une analyse sémantique en sortie, telle que (49) :

(49) `PRED=see(John,Mary)`

Notons que, quant à la capacité transformationnelle, l'analyse produite est puissante, en particulier, l'expression de sortie peut être adaptée facilement pour exprimer d'autres formalismes (Silberztein). Cependant, cette analyse est retreinte à produire une sortie passive, autrement dit, la grammaire locale est nullement capable de comprendre les expressions du formalisme qui est en train de produire, ainsi (49) n'a aucun effet ni sur (48) ni sur les autres phrases qui pourrait être reconnues. Ceci restreint énormément la capacité d'analyse sémantique. En effet, pour cet exemple, la seule stratégie pour créer une base de connaissances à partir des prédicats générés afin de faire des requêtes déductibles des faits est d'attendre la fin de l'analyse effectuée par la grammaire et ensuite d'utiliser les sorties résultantes comme entrée d'un interpréteur logique comme Prolog.

Naturellement, la question qu'on se pose est de savoir si en utilisant le formalisme des grammaires locales étendues il est possible de 1. Produire en sortie des expressions du type prédicat–argument, 2. Utiliser ces expressions pour ajouter, à la volé, des faits dans une base de connaissances et 3. Faire des requêtes à la base au cours des analyses. Nous appelons cette démarche une analyse sémantique active, ceci en opposition à l'analyse sémantique des grammaires locales classiques, comme celle du graphe 7.3, limitée à produire en sortie des expressions logiques.

L'analyse sémantique effectuée par une grammaire étendue permet d'utiliser deux niveaux de connaissances : d'abord un niveau *endogène* basée sur la reconnaissance des entrées et la construction de faits qui découle, ensuite un niveau *exogène* où le point de départ est une base de connaissances avec des faits préétablies, c'est-à-dire, avec des prédicats qui ne viennent pas de l'entrée mais de l'extérieur.

7.3 Recherche adaptative de motifs dans un dictionnaire électronique

Dans cette section nous étudions le problème de la recherche approximative de motifs dans un dictionnaire. Étant donné un dictionnaire W sur un alphabet fini Σ, un motif p sur le même alphabet, une distance dans l'espace métrique $\delta : \Sigma^* \times \Sigma^* \to \mathbb{R}^+$ et un seuil maximal autorisé k, le problème consiste à retrouver l'ensemble des entrées de W qui coïncide avec p tel que la distance δ entre chaque candidat et p est plus petite ou égale à k. Nous nous intéressons donc à toutes les occurrences de W qui peuvent être transformées en p avec k opérations au plus : $S = \{s \in W : \delta(s, p) \leq k\}$

Comme pour le problème de la recherche exacte de motifs [1], il existe deux grandes approches pour effectuer une recherche approximative. D'une part, celles fondées sur l'algorithmie, aussi appelées méthodes de recherche en ligne (*on-line*), qui permettent de résoudre une nouvelle instance du problème à chaque fois que l'algorithme est

1. Voir à ce sujet Faro et Lecroq (2011)

exécuté. D'autre part, les approches basées sur les structures de données, ou méthodes de recherche hors ligne (*off-line*), qui réalisent d'abord une phase de prétraitement de W pour ensuite répondre à toute requête impliquant p.

Le temps d'exécution des méthodes de recherche approximative en ligne ont une complexité linéaire en la taille du texte, ce qui les rend inefficaces pour exploiter de grands volumes de données textuelles (Boytsov, 2011). Dans le cas de la recherche approximative de motifs dans un dictionnaire, cette contrainte motive la réflexion sur la façon de concevoir des algorithmes hors-ligne plus efficaces.

Nous présenterons à la fin de ce chapitre une structure de données et des algorithmes qui permettent de chercher efficacement p en W avec une métrique δ et un seuil de distance k, ainsi que connaître combien de mots de W coïncident avec p et quelle est la position de ces occurrences. Nous verrons aussi comment l'approche proposée ne se limite pas à une seule métrique d'édition, mais peut aussi s'adapter à la nature du problème et des données.

7.3.1 Approche naïve

L'approche naïve pour rechercher approximativement un motif p dans un dictionnaire W consiste à parcourir toutes les entrées du dictionnaire en calculant chaque fois la distance entre p et l'entrée s, soit l'ensemble d'occurrences $S = \{s \in W : \delta(s,p) \leq k\}$.

Algorithme 1 : Recherche approximative d'un motif p dans un dictionnaire W

Entrées : p,k,W
Résultat : $S = \{s \in W : \delta(s,p) \leq k\}$

```
1 n ← |p|;
2 s ← { };
3 pour chaque s ∈ W faire
4 |   d ← δ(s,p);
5 |   si d ≤ k alors
6 |   |   empiler s dans S ;
7 |   finsi
8 finprch
9 return S;
```

Si la longueur moyenne d'une entrée est n, alors la complexité de cet algorithme est bornée par le coût du calcul de la distance d'édition effectuée à la ligne 4, soit $\mathcal{O}(n^2)$ en utilisant l'algorithme standard de programmation dynamique proposé par Wagner et Fischer (1974), en rapport à la taille du dictionnaire $|W|$, c'est-à-dire $\mathcal{O}(|W|\, n^2)$.

7.3.2 Motivation et objectifs

Les algorithmes de la littérature pour la recherche approximative de motifs se sont concentrés à traiter la nature du problème ainsi qu'à améliorer la complexité temporelle ou spatiale. Contrairement aux algorithmes pour la recherche exacte, peu de références existent faisant état d'approches qui soient efficaces à la fois en temps et en espace, par exemple qui mobilisent des méthodes hors-ligne avec une complexité sous-linéaire en la taille du texte en restant compétitives en rapport à l'utilisation de l'espace mémoire.

7.3.3 Distance d'édition

La distance d'édition entre deux mots est définie comme étant le nombre minimum d'insertions, de suppressions ou de substitutions de symboles nécessaires pour transformer un mot en un autre.

Pour le cas des mots de taille n, l'algorithme standard de programmation dynamique de Wagner et Fischer (1974) a une complexité en temps quadratique $\mathcal{O}(n^2)$. Wong et Chandra (1976) ont démontré que ceci est optimal dans le modèle restreint où l'on ne peut que comparer les caractères les uns aux autres. Les autres algorithmes connus, qui essentiellement calculent la distance d'édition en traitant des blocs de caractères comme un seul, ont une complexité en temps sous-quadratique. Notamment, pour un alphabet fini et un coût discret des opérations d'édition, la meilleure borne supérieure connue, due à Masek et Paterson (1980), est $\mathcal{O}(n^2/\log n)$ [1]. Concernant ce résultat, le problème d'avoir un coût d'opérations non limité aux rationnels est resté ouvert pendant deux décennies, jusqu'à ce que Crochemore et al. (2002, 2003) proposent un algorithme en temps $\mathcal{O}(n^2/\log n)$ [2]. La question de trouver une meilleure borne supérieure est, de nos jours, encore ouverte. Cependant, il pourrait ne pas en exister une, en particulier, Backurs et Indyk (2014) fournissent une preuve démontrant l'impossibilité d'existence d'un algorithme pour calculer la distance d'édition en temps fortement sous-quadratique, c'est-à-dire en temps $\mathcal{O}(n^{2-c})$ pour une constante $c > 0$, à moins que la conjecture SETH (*Strong Exponential Time Hypothesis*) sur la résolution du problème de satisfaisabilité booléenne (SAT) soit fausse [3]. Ce qui continuerait à restreindre le calcul à une complexité en temps quadratique avec un facteur logarithmique.

En raison du coût élevé du calcul de la distance d'édition, plusieurs approches ont vu le jour, parmi lesquelles celles fondées sur des schémas de filtrage-vérification comme les q-grammes.

1. Si le longueur du deuxième mot est égal à m et $n > m$, alors la borne supérieure est $\mathcal{O}(n \max(1, m/\log n))$

2. Entre autre, les auteurs montrent que pour des séquences compressibles, la borne supérieure est $\mathcal{O}(hn^2/\log n)$, ou h est un nombre réel, $0 \leq h \leq 1$, qui mesure l'entropie de la séquence.

3. La preuve apportée par les auteurs consiste à démontrer que si le calcul de la distance d'édition peut être effectué en $\mathcal{O}(n^{2-c})$, pour une constante $c > 0$, alors une instance du problème SAT de N variables et M clauses sous forme normale conjonctive (CNF-SAT) pourrait être résolue en temps $M^{O(1)}2^{(1-\epsilon)N}$ pour une constante $\epsilon > 0$, ce qui contredirait la conjecture SETH formulée par Impagliazzo et Paturi (2001) qui revendique qu'un tel algorithme n'existe pas.

7.3.4 Recherche approximative dans un dictionnaire classique

T. Bocek (2007) présentent un algorithme hors ligne appelé FastSS (*Fast Similarity Search*) pour la recherche approximative de motifs. L'approche utilisée par FastSS met en œuvre une variante de la stratégie de génération de mots voisins (*word neighborhoods*) présentée auparavant par Myers (1994). L'algorithme se déroule en deux étapes : la première (indexation) : étant donnée un dictionnaire W de n entrées avec une longueur moyenne de m caractères et un nombre maximal d'erreurs autorisés égal à k, consiste à construire un dictionnaire T de taille $\mathcal{O}(nm^k)$ de toutes les k-suppressions de chaque entrée (*deletion neighbourhoods*). Ensuite, dans une deuxième étape (recherche), l'algorithme tente de créer un dictionnaire de taille $\mathcal{O}(m^k)$ de toutes les k-suppressions du motif p en les comparant à T en $\mathcal{O}(\log |T|)$ opérations. L'intuition sous-jacente est la suivante : précalculer et chercher les k-suppressions d'une entrée est plus rapide que traiter également les insertions et remplacements.

7.3.5 Dictionnaires électroniques du LADL

Comme nous l'avons évoqué au chapitre 3, les expressions décrites par les grammaires locales ne sont pas simplement des unités lexicales. En effet, elles peuvent aussi faire appel à des listes et des classes de mots contenus dans des lexiques qui sont stockés sous la forme de dictionnaires électroniques.

Ces derniers se distinguent des lexiques classiques par les informations associées à chaque unité lexicale qui constitue les entrées du dictionnaire. La capacité d'avoir recours aux informations morphosyntaxiques des dictionnaires représente une des caractéristiques les plus importantes des grammaires locales en leur conférant un grand pouvoir descriptif.

Les dictionnaires utilisés dans les grammaires locales sont ceux du LADL (Courtois et Silberztein, 1990, Courtois et al., 1997), le tableau 7.1 résume les types de dictionnaires disponibles.

Nom	Formes	Catégorie
DELAS	Canoniques	Mots simples
DELAF	Fléchies	Mots simples
DELAC	Canoniques	Mots composés
DELACF	Canoniques et fléchies	Mots composés
DELAP	Canoniques et fléchies	Mots phonétiques

Tableau 7.1 – Dictionnaires électroniques

Définition 7.4 (Trait)**.** Un trait est une 2-tuple $T = (\text{attribut}, \text{valeur})$.

Définition 7.5 (Structure de traits)**.** Une structure de traits $S = (t_1, t_2, \ldots, t_n)$ est un ensemble ordonné de traits.

Définition 7.6 (Dictionnaire)**.** Un *dictionnaire* $W = (s_1, s_2, \ldots, s_n)$ est un ensemble ordonné de mots, aussi appelés entrées, où $n = |W|$ est le nombre d'entrées du dictionnaire.

Définition 7.7 (Dictionnaire morphosyntaxique)**.** Un *dictionnaire morphosyntaxique* est un ensemble de tuples $(d_1, d_2, \ldots, d_n)$.

7.3.6 k-suppressions

Nombre de k-suppressions. Soit un mot de longueur n. Alors le nombre de k-suppressions est égal à $f(n,k) = \sum_{m=1}^{k} \binom{n}{m}$.

Par exemple, le nombre de k-suppressions du mot *page*, listées au tableau 7.2, pour $k = 1..4$, est égal a :

1**-suppressions.** $f(4,1) = \binom{4}{1} = 4$

2**-suppressions.** $f(4,2) = \binom{4}{1} + \binom{4}{2} = 10$

3**-suppressions.** $f(4,3) = \binom{4}{1} + \binom{4}{2} + \binom{4}{3} = 14$

4**-suppressions.** $f(4,4) = \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 15$

Un extrait du nombre total de k-suppressions pour un mot de longueur $n \leq 10$ est donné dans le tableau 7.3.

$f(4,1)$	ε a g e p ε g e p a ε e p a g ε	4
$f(4,2)$	ε a g e ε ε g e ε a ε e ε a g ε p ε g e p ε ε e p ε g ε p a ε e p a ε ε p a g ε	10
$f(4,3)$	ε a g e ε ε g e ε ε ε e ε ε g ε ε a ε e ε a ε ε ε a g ε p ε g e p ε ε e p ε ε ε p ε g ε p a ε e p a ε ε p a g ε	14
$f(4,4)$	ε a g e ε ε g e ε ε ε e ε ε ε ε ε ε g ε ε a ε e ε a ε ε ε a g ε p ε g e p ε ε e p ε ε ε p ε g ε p a ε e p a ε ε p a g ε	15

Tableau 7.2 – k-suppressions du mot *page*

n										
1	1									
2	2	3								
3	3	6	7							
4	4	10	14	15						
5	5	15	25	30	31					
6	6	21	41	56	62	63				
7	7	28	63	98	119	126	127			
8	8	36	92	162	218	246	254	255		
9	9	45	129	255	381	465	501	510	511	
10	10	55	175	385	637	847	967	1012	1022	1023
	1	2	3	4	5	6	7	8	9	10
					k					

Tableau 7.3 – Nombre de k-suppressions pour un mot de longueur n

Déroulement. Pour un mot de longueur n et un entier k, le nombre de k-suppressions est égal à la somme partielle des $k+1$ premiers nombres de la $n^{\text{ième}}$ ligne du triangle de Pascal moins 1 :

$$\sum_{m=0}^{k} \binom{n}{m} - 1$$

ou, en simplifiant,

$$\binom{n}{0} + \sum_{m=1}^{k} \binom{n}{m} - 1 = \sum_{m=1}^{k} \binom{n}{m}$$

nous avons alors les cas suivants :

$$f(n,k) = \begin{cases} 0 & \text{si } k = 0 \\ n & \text{si } k = 1 \\ 2^k - 1 & \text{si } k = \text{n} \\ 2^k - 2 & \text{si } k = n-1 \\ \displaystyle\sum_{m=1}^{k} \binom{n}{m} & \text{si } k \geq 2 \end{cases}$$

i					
5	\$	p	a	g	e
2	a	g	e	\$	p
4	e	\$	p	a	g
3	g	e	\$	p	a
1	p	a	g	e	\$
	1	2	3	4	5
			j		

Tableau 7.4 – Possibles rotations du mot *page*

$f(4,1)$	ε	a	g	e	$\$_{\overrightarrow{1}}$	ε	$\$_{\overleftarrow{2}}$
	p	ε	g	e	$\$_{\overrightarrow{2}}$	ε	$\$_{\overleftarrow{3}}$
	p	a	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	p	a	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
$f(4,2)$	ε	a	g	e	$\$_{\overrightarrow{1}}$	ε	$\$_{\overleftarrow{2}}$
	ε	ε	g	e	$\$_{\overrightarrow{2}}$	ε	$\$_{\overleftarrow{3}}$
	ε	a	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	ε	a	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	ε	g	e	$\$_{\overrightarrow{2}}$	ε	$\$_{\overleftarrow{3}}$
	p	ε	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	p	ε	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	a	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	p	a	ε	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	a	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
$f(4,3)$	ε	a	g	e	$\$_{\overrightarrow{1}}$	ε	$\$_{\overleftarrow{2}}$
	ε	ε	g	e	$\$_{\overrightarrow{2}}$	ε	$\$_{\overleftarrow{3}}$
	ε	ε	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	ε	ε	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	ε	a	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	ε	a	ε	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	ε	a	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	ε	g	e	$\$_{\overrightarrow{2}}$	ε	$\$_{\overleftarrow{3}}$
	p	ε	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	p	ε	ε	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	ε	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	a	ε	e	$\$_{\overrightarrow{3}}$	ε	$\$_{\overleftarrow{4}}$
	p	a	ε	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$
	p	a	g	ε	$\$_{\overrightarrow{4}}$	ε	$\$_{\overleftarrow{5}}$

Tableau 7.5 – Génération des k-suppressions du mot *page*

7.4 Tolérance au bruit

Nous avons exposé qu'une des principales limitations des approches fondées sur les grammaires locales découle de la difficulté de concevoir des grammaires complètes. En d'autres termes, il est difficile d'augmenter la couverture d'une grammaire locale pour capturer davantage de combinaisons de motifs ou d'accroître les possibilités de traitement de variations attendues et inattendues dans un texte. Au-delà de cela, les grammaires locales ne disposent pas de mécanismes leur permettant accéder à des connaissances exogènes, c'est-à-dire des informations extraites de ressources externes au cours de l'analyse et qui peuvent s'avérer utiles pour normaliser, transformer, enrichir ou valider les motifs reconnus.

Cette section concerne l'extraction d'information dans des textes bruités. L'approche exposée mobilise le développement de trois sortes de ressources :

- Des dictionnaires électroniques, tel que des dictionnaires d'anthroponymes et de toponymes, permettant des recherches approximatives (cf. section 7.3).
- Des fonctions étendues pour la manipulation des chaînes de caractères, de variables et, en général, pour la mise en place des stratégies de recherche approximative de motifs.
- Des grammaires locales étendues pour l'extraction d'informations dans des textes bruités.

Nous étudions chacune ces ressources en apportant une attention particulière à présenter la façon dont elles sont agencées afin de permettre l'extraction d'information (normalisation, validation et enrichissement d'entités) même dans des textes bruités.

Par rapport au pouvoir d'analyse d'une grammaire locale classique, le fait de construire des grammaires locales étendues dotées de fonctions étendues, nous permet de :

- Prendre en charge l'analyse de cas non traités, par exemple, la faculté de rapprocher un mot inconnu, ou des mots connus qui forment un mot composé inconnu, d'une entrée lexicale disponible dans un dictionnaire ou dans n'importe quelle autre type de ressource (fichier texte, base de données, etc.)
- Étendre les capacités pour convertir une phrase en une autre, par exemple, au-delà des opérations transformationnelles traitant le temps, l'aspect, le mode ou la voix, une phrase comme *Jean avale une pomme* peut être transformée en *Jean mange un fruit*, avec une grammaire qui décrit la structure syntaxique et qui fait à la fois appel a une ressource externe, capable de rapprocher le mot *avale* à *mange* et celui de *pomme* à *fruit*.

C'est dans le principe de l'ingénierie des grammaires que s'inscrit notre approche. D'une part, nous utilisons une approche classique pour reconnaître les séquences correctes et plus fréquentes :

- Premièrement, nous cherchons les unités élémentaires d'un sous-langage, par exemple, lorsque nous étudions le problème de géolocalisation d'une adresse, nous considérons les codes postaux, les noms des villes et des pays ; ensuite, nous les répertorions soit sous la forme de dictionnaires électroniques, soit sur des listes de mots ou encore en identifiant des masques lexicaux capables de les repérer.

- Deuxièmement, nous réalisons une description formelle des contraintes morphosyntaxiques d'une phrase ou segment de phrase sous la forme de grammaires locales. Dans l'exemple d'une adresse postale, nous décrivons les enchaînements les plus fréquents des unités élémentaires : ⟨Ville⟩ , ⟨Pays⟩ [1] ; ⟨CodePostal⟩ ⟨Ville⟩ , ⟨Pays⟩ [2] ; ⟨CodePostal⟩ ⟨Ville⟩ ⟨NB⟩ [3] ; ⟨Ville⟩ ⟨CodePostal⟩ [4], etc.

- Finalement, une fois la description linguistique réalisée, les dictionnaires et grammaires sont transformés en automates finis qui peuvent être appliqués de façon efficiente à un texte afin d'extraire des informations.

D'autre part, nous implémentons et faisons appel à partir des graphes à des fonctions étendues.

7.4.1 Grammaires locales et tolérance au bruit

Concernant la reconnaissance de motifs, tels que des dates, dans des documents bruités à l'aide des grammaires locales, la littérature disponible n'est pas exhaustive. Citons par exemple, le travail de Sagot et Gábor (2014) pour la détection et correction automatique d'entités nommées dans des corpus OCRisés.

Dans son étude, Sagot propose une architecture d'identification et de correction d'erreurs dans des entités nommées à l'aide de SXPipe (Sagot et Boullier, 2008). Le cœur de la démarche consiste à construire des grammaires locales SXPipe [5]. Ces grammaires, appliquées en cascade, reconnaissent des dates, des adresses et des formules chimiques sous une forme standard ainsi que sous une forme légèrement bruitée. Par la

1. *Vitry Sur Seine, France*
2. *75123 Uppsala, Sweden*
3. *13385 Marseille 05*
4. *London WC1E 6BT*
5. Les grammaires locales de SXPipe ne sont pas construites à l'aide d'une interface graphique, en revanche, il est possible d'utiliser un module nommé UNITEX2SXPIPE pour convertir un graphe d'Unitex version 2.0 en une grammaire `dag2dag` (Sagot et Boullier, 2008). Les grammaires `dag2dag` sont exprimées dans un langage proche de la forme de Backus–Naur (BNF) et analysées à l'aide du système SYNTAX (Boullier et Deschamp (1991)).

suite, des règles de correction sont appliquées [1] pour obtenir, si nécessaire, leur version normalisée et corrigée.

Les résultats d'évaluation sur des corpus appartenant aux domaines de la littérature, la jurisprudence et les brevets, montrent un bon rappel et une haute précision en correction, ce qui conduit les auteurs à déduire l'adéquation de l'architecture proposée pour une tâche de reconnaissance et de correction de corpus OCRisés. L'étude conclut en ouvrant deux perspectives, la première vers l'intégration des grammaires locales SXPipe avec des modèles d'erreurs et des modèles statistiques de langage [2], la seconde en permettant aux grammaires locales SXPipe de proposer plus d'une correction et laisser le modèle d'erreurs choisir la plus pertinente.

Concernant ces deux perspectives de travaux futurs, nous verrons comment une grammaire locale étendue permet de les mettre en place sous une forme intuitive. Ceci sans avoir recours à un couplage en amont des modèles statistiques à partir des sorties de la grammaire comme envisagé par l'étude précitée. En effet, une approche basée sur des grammaires locales étendues peut alors faire appel à un éventail de techniques, dont celles fondées sur des méthodes statistiques, qui sont sollicitées au fur et à mesure que l'analyse se déroule et pas simplement utilisées à l'issue de la reconnaissance.

7.4.2 Reconnaissance tolérante au bruit à l'aide des grammaires étendues

Si la construction des grammaires locales peut s'appuyer sur la même technique utilisée pour reconnaître des langages contextuels (cf. section 3.5.8), quelle est la limite des grammaires locales classiques pour tolérer du bruit, par exemple une faute lexicale inattendue ? Si ces grammaires locales sont capables de définir un langage non contextuel, la première partie de l'approche est accomplie, il suffit donc de définir la façon d'exclure les séquences qui n'appartiennent pas au langage que nous souhaitons décrire pour y parvenir. Cependant, c'est dans la définition de l'opération à réaliser pour exclure ces séquences que les grammaires locales classiques trouvent leur limite.

Dans le cas pratique, pour le problème qui nous concerne, définir l'opération à réaliser pour exclure les séquences revient à définir premièrement la notion de bruit dans la reconnaissance, sujet lié à ce que nous cherchons à reconnaître. Deuxièmement, il faut spécifier les opérations qui permettent de distinguer les séquences bruitées proches de celles recherchées de séquences non désirées. Enfin, il est indispensable de définir comment incorporer ces opérations dans l'algorithme d'analyse syntaxique et comment les utiliser lors la conception des grammaires locales.

Malheureusement, cette stratégie devient rapidement problématique. D'une part il

1. L'identification de règles de correction est effectuée par exploration manuelle d'un corpus d'écarts. Par exemple pour la reconnaissance des nombres, les règles construites par inspection pour faire correspondre un caractère à un chiffre peuvent avoir la forme : $0 \Leftarrow \{o, ^\circ\}$, $1 \Leftarrow \{I, l, i\} \cdots 9 \Leftarrow \{g\}$.

2. L'approche envisagée pour ce faire est d'utiliser l'architecture exposée comme un pré-traitement qui viendra en amont du composant reposant sur les modèles statistiques

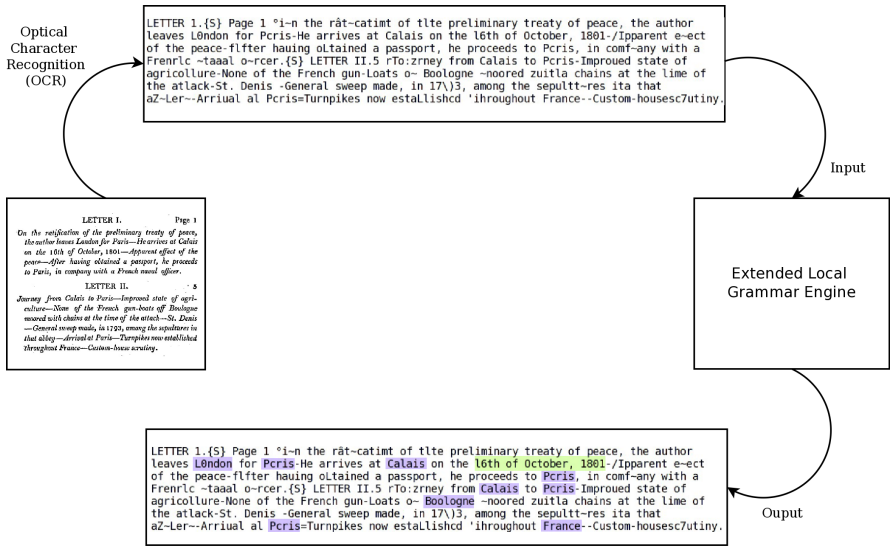


Figure 7.2 – Reconnaissance d'entités nommées à l'aide des grammaires locales étendues

n'est pas possible de définir à l'avance le type du bruit à traiter, ce qui permettrait d'implémenter l'opération de filtrage la plus adéquate. D'autre part, même avec quelques opérations standards, elles seront très restreintes en l'absence de mécanismes permettant de :

- paramétrer, modifier et par conséquent adapter, corriger ou améliorer l'implémentation de l'opération,
- accéder aux informations au-delà des données fournies par l'analyse syntaxique courante,

Cela veut dire qu'au niveau de l'implémentation, un couplage fort entre l'algorithme d'analyse syntaxique et les opérations sur les grammaires, implique également d'alourdir la taille de l'analyseur et sa maintenance, tâche toujours sujette aux erreurs. Même en négligeant ces points, l'analyseur résultant, sera soumis à un ensemble d'opérations prédéfinies à l'avance, ni modifiables ni extensibles.

La stratégie qui peut être déployée à l'aide des grammaires locales étendues est abordée différemment, les opérations sont des fonctions étendues définies hors de l'algorithme d'analyse syntaxique et ne sont pas connues à l'avance. Cette adaptabilité permet de :

- définir librement la notion de bruit dans la reconnaissance, pouvant prendre en compte, par exemple, des aspects dérivés de la morphologie, la syntaxe, la sémantique ou la pragmatique ;
- implémenter la fonction la plus adaptée pour exclure les séquences qui ne sont pas considérés comme bien formées.

En outre, comme vue dans le chapitre 4, une fonction peut recevoir des arguments pour adapter son comportement, être corrigée, améliorée et, en particulier, utiliser des connaissances issues à partir de 3 sources d'information différentes [1] :

1. **endogènes** : des informations locales issues de l'analyse de la séquence courante ainsi que des informations globales correspondant à l'analyse des autres séquences qui sont regroupées dans une même analyse,

2. **exogènes** : des informations issues de ressources externes qui ne sont pas directement disponibles à partir de l'analyse, soit celles qui ne sont pas directement ou indirectement intégrées dans l'algorithme d'analyse syntaxique, et

3. **anthropogènes** : des informations qui peuvent être fournies par l'utilisateur au cours de l'analyse.

Par la suite nous présentons des exemples de grammaires locales étendues capables d'extraire de l'information de données bruitées.

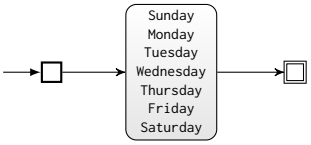
7.4.3 Reconnaissance de dates et tolérance au bruit

Il est bien connu que les expressions de temps, telles que les dates, peuvent être définies aisément à l'aide de grammaires locales. Cependant, la reconnaissance est limitée aux séquences bien formées et parfois peut tolérer certaines séquences mal formées connues à l'avance.

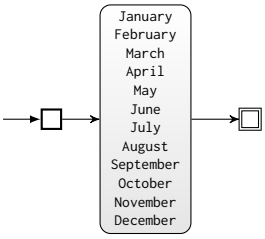
Considérons le graphe 7.4 capable de reconnaître certaines dates en anglais. Ce graphe est composé de 4 sous-graphes en charge d'identifier respectivement les jours de la semaine (7.4b), les mois (7.4c), les nombres du jour entre 1 et 31 (7.4d), ainsi que les années entre 1200 et 2099 (7.4e).
Aussi bien le nombre du jour comme l'année sont optionnels. De plus, une virgule peut être présente ou pas après le jour ou avant l'année. En général, les dates reconnues auront une des formes présentées dans le tableau 7.6.

1. Ces trois sources correspondent aux processus d'immersion documentaire développés largement dans Andreani (2011)

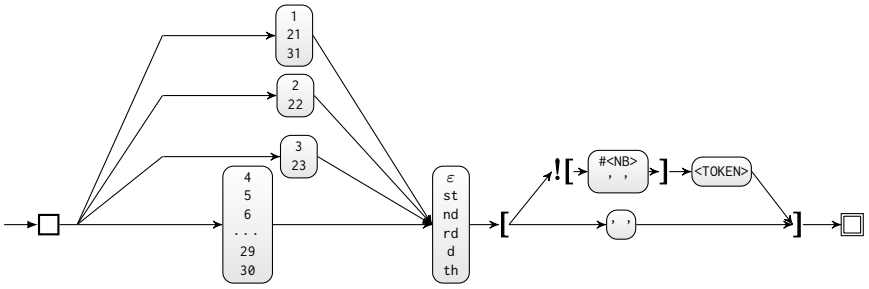
(a) Graphe principal



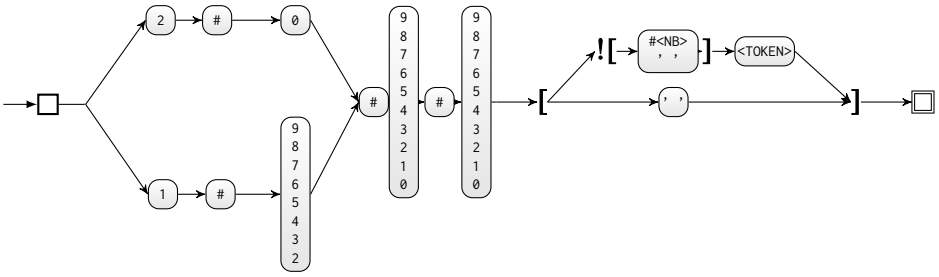
(b) Jour



(c) Mois



(d) Nombre du jour



(e) Année

Graphe 7.4 – Reconnaissance des dates en anglais

Type du motif	Exemple
⟨month⟩ ⟨day-number⟩	January 13th
⟨month⟩ ⟨day-number⟩ ⟨year-number⟩	December 18 1987
⟨month⟩ ⟨day-number⟩, ⟨year-number⟩	April 12, 2016
⟨day⟩ ⟨month⟩ ⟨day-number⟩	Sunday September 17
⟨day⟩ ⟨month⟩ ⟨day-number⟩ ⟨year-number⟩	Thursday March 15 2012
⟨day⟩ ⟨month⟩ ⟨day-number⟩ , ⟨year-number⟩	Wednesday August 29, 1792
⟨day⟩, ⟨month⟩ ⟨day-number⟩	Friday, August 3rd
⟨day⟩, ⟨month⟩ ⟨day-number⟩ ⟨year-number⟩	Saturday, May 12th 2001
⟨day⟩, ⟨month⟩ ⟨day-number⟩, ⟨year-number⟩	Tuesday, October 11, 1492

Tableau 7.6 – Types de dates reconnues par le graphe 7.4

Il est évident que face à un texte avec des séquences bien formées, c'est-à-dire, correspondant aux entrées attendues, la précision de la reconnaissance du graphe 7.4 sera haute [1]. Cependant, quelle sera la performance lorsque l'analyse se déroule sur des entrées bruitées ? Par exemple, quand l'analyse s'effectue sur des corpus qui ont été constitués en transformant automatiquement des images en texte grâce à un logiciel de reconnaissance optique de caractères (OCR). Considérons les dates présentées dans le tableau 7.7 provenant d'un corpus océrisé [2].

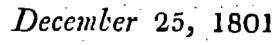
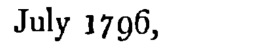
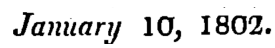
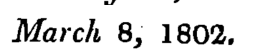
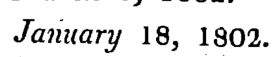
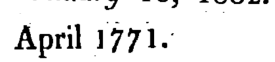
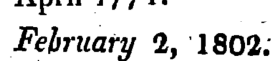
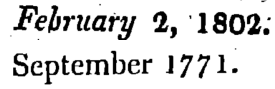
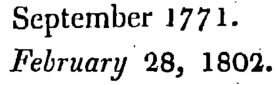
Image	Résultat OCR
	Decémler 25, 1801
	July] 7g6
	JanuarJ 10, 1802
	Marcia 8., 18.02
	.Ta iuar y 18, 180 2
	April]771
	Fe,Lr:cary 2 '1802
	September J 771
	Februartj 28, 1802

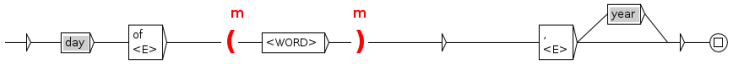
Tableau 7.7 – Exemples de dates provenant d'un corpus océrisé

Le résultat OCR des exemples des dates listées dans le tableau 7.7 n'est pas conforme aux sous-séquences décrites par le graphe 7.4. D'une part certains noms de mois contenant du bruit ne sont pas répertoriés, à l'instar de « *JanuarJ* » ou « *Fe,Lr :cary* ». D'autre part les années ne sont plus formées par des nombres mais contiennent d'autres caractères, comme dans « *] 7g6* » ou « 180 2 ». À partir de cet exemple de base, la question est de savoir s'il est possible de concevoir une grammaire locale étendue capable d'identifier des dates bien formées, ainsi que des dates comportant du bruit. Nous verrons par la suite comment l'utilisation des grammaires locales étendues peut aider à obtenir une réponse affirmative.

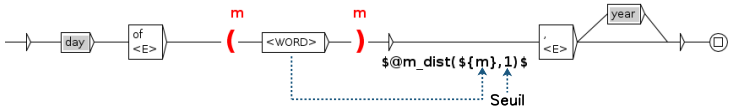
Approche utilisant une grammaire locale étendue Nous cherchons à vérifier, à l'aide de la grammaire locale étendue présentée à la figure figure 7.3, si les entrées suivantes sont de dates :

1. Notons que dans Unitex les chiffres son traités comme des tokens indépendants. En conséquence, dans le graphe 7.4, le sous-graphe **(e)** *Année* peut reconnaître par exemple l'année 2017 mais aussi le chiffre 12017. Surmonter ce problème à l'aide d'une grammaire locale étendue devient facile, il suffit de vérifier, à l'aide d'une fonction étendue, que le chiffre est composé par 4 numéros.

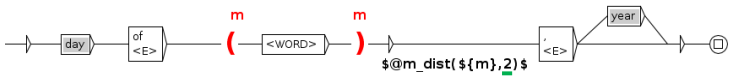
2. *Paris as it was and as it is, or a Sketch of the French capital illustrative of the effects of the Revolution.* Bibliothèque nationale de France. http://gallica.bnf.fr/ark:/12148/bpt6k102153w



(a) Dates : reconnaître un langage plus grand



(b) Dates : vérifier le mois à l'aide d'une fonction étendue, seuil 1



(c) Dates : vérifier le mois à l'aide d'une fonction étendue, seuil 2

Figure 7.3 – Dates : grammaire locale étendue

- 28th of January.
- 5 of Nlarch 1792
- 8th of Nivôse
- 3 OctUber
- 3 of Brumaire
- 23d of Scptember
- 11 CQuI1trppen

Évaluation : Nous utilisons un corpus océrisé de la BnF [1], comportant $\sim$ 128.000 mots et 100 dates. Nous comparons la performance de la grammaire étendue (avec les fonctions étendues activées et désactivées) par rapport à trois autres systèmes : une grammaire locale classique, l'extracteur des expressions temporelles de l'outil OpenNLP qui est fondée sur un modèle d'entropie maximale (Ratnaparkhi, 1997) et le module SUTime du Standford CoreNLP (Chang et Manning, 2012).

1. *Paris as it was and as it is, or a Sketch of the French capital illustrative of the effects of the Revolution.* Bibliothèque nationale de France. http://gallica.bnf.fr/ark:/12148/bpt6k102153w

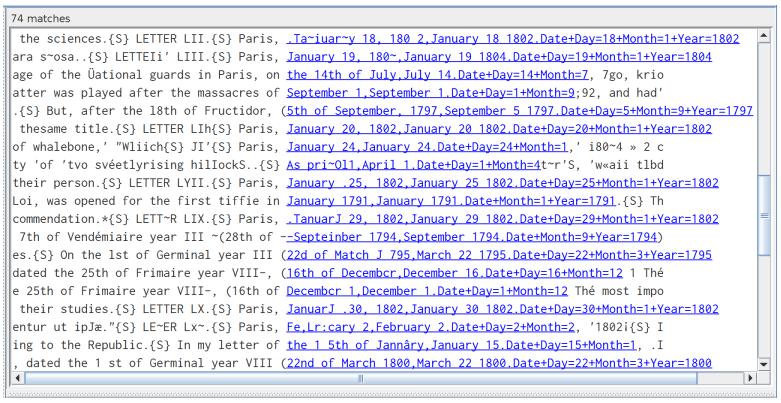


Figure 7.4 – Exemple de dates reconnues à l'aide d'une grammaire locale étendue

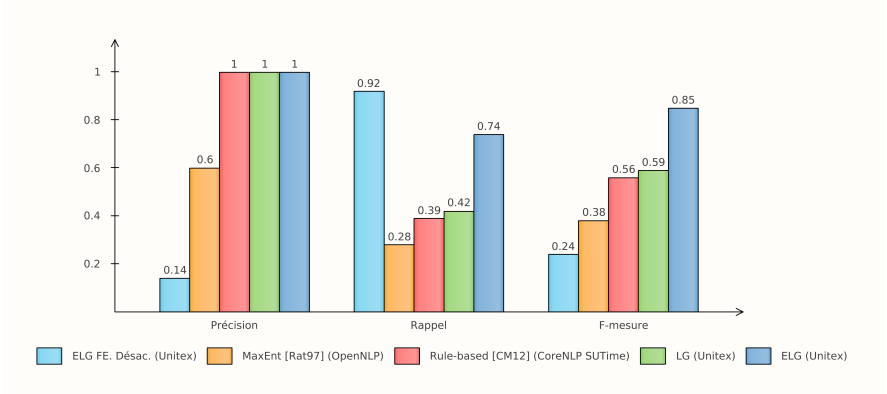


Figure 7.5 – Évaluation reconnaissance de dates

7.4.4 Reconnaissance d'anthroponymes et classification du genre des prénoms

Nous cherchons à reconnaître des noms de personnes provenant d'un corpus journalistique ainsi qu'à classifier le genre du prénom en deux catégories : masculin ou féminin :

i. A World Health Organization official, Margaret Chan, told the meeting there is a great risk.
- A World Health Organization official, Margaret Chan, told the meeting there is a great risk
- `A World Health Organization official, <START:person gender="f"> Margaret Chan <END>, told the meeting there is a great risk.`

ii. The Swiss star was upset Wednesday by German Tommy Haas in the opening match.
- The Swiss star was upset Wednesday by German Tommy Haas in the opening match.
- `The Swiss star was upset Wednesday by German <START:person gender="m"> Tommy Haas <END> in the opening match.`

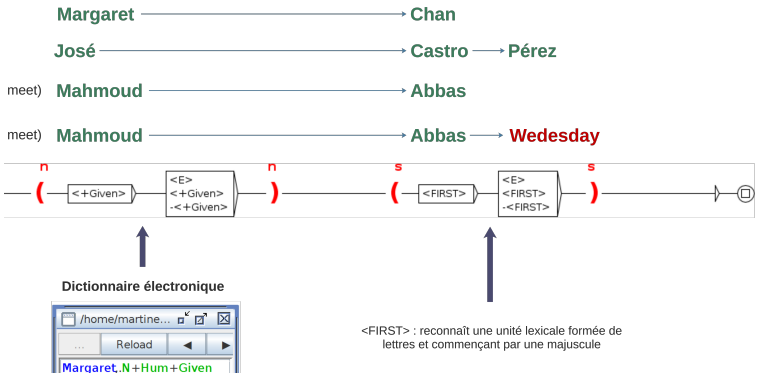


Figure 7.6 – Anthroponymes : reconnaître un langage plus grand

Évaluation : Nous utilisons un sous-corpus du GMB(Groningen Meaning Bank) [1], au format CoNLL 2002, comportant ∼ 24.000 mots et 215 anthroponymes. Deux évaluations sont réalisées, la première en conservant la casse originale du texte, la deuxième en mettant en majuscule la première lettre de chaque mot. Nous comparons la performance de la grammaire étendue (avec les fonctions étendues activées et

1. Disponible sur https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus

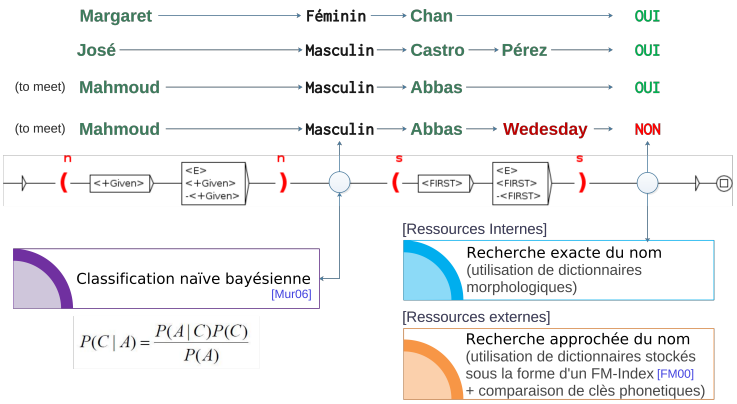


Figure 7.7 – Anthroponymes : vérifier les noms à l'aide d'une fonction étendue

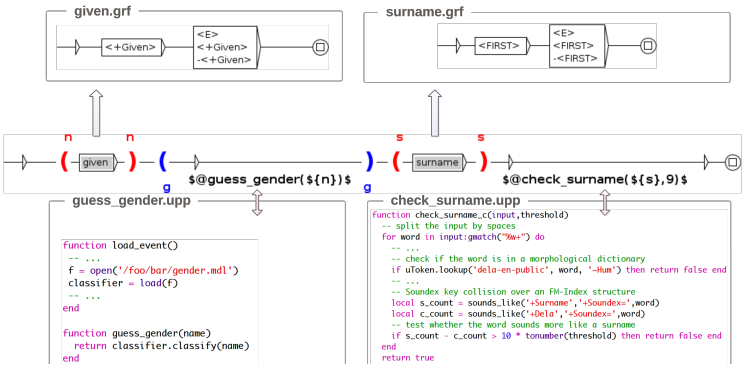


Figure 7.8 – Anthroponymes : implémentation de la fonction étendue

désactivées) par rapport à quatre autres systèmes : une grammaire locale classique, une cascade de transducteurs de l'outil CasEN (Maurel et al., 2011) dans sa version ISTEX anglaise, l'extracteur de noms de personnes de l'outil OpenNLP qui est fondé sur un modèle d'entropie maximale (Ratnaparkhi, 1997) et le modèle à 3 classes (personne, organisation lieu) du Standford CoreNLP fondé sur des champs aléatoires conditionnels (Manning et al., 2014).

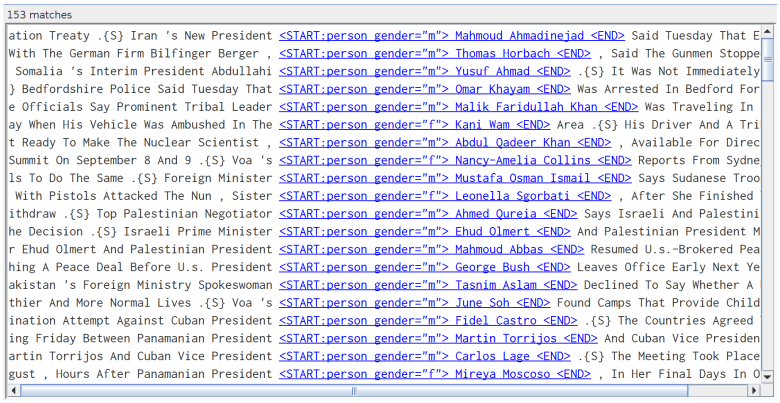


Figure 7.9 – Exemple de noms reconnus à l'aide d'une grammaire locale étendue

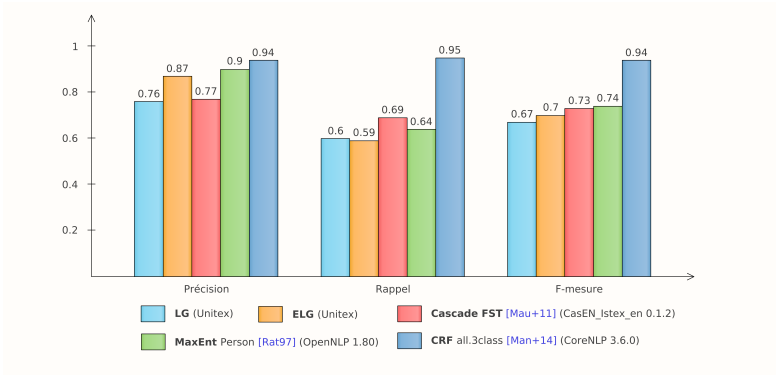


Figure 7.10 – Évaluation I reconnaissance d'anthroponymes, texte original

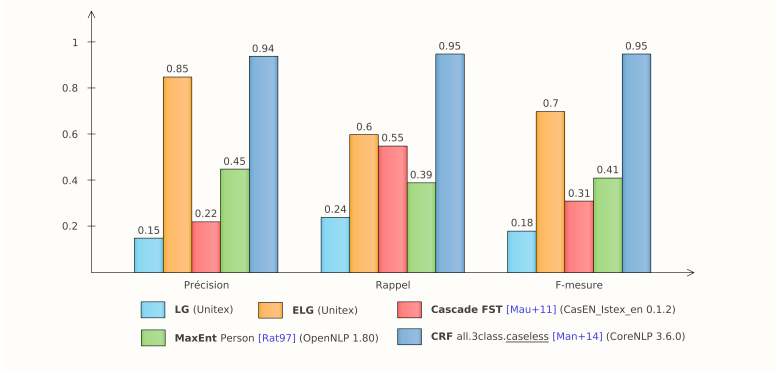


Figure 7.11 – Évaluation II reconnaissance d'anthroponymes, changement de casse

7.4.5 Reconnaissance de toponymes et géolocalisation des adresses d'organisations

Nous cherchons à géolocaliser des adresses d'organisations provenant de notices bibliographiques (*Web of Knowledge*) :

i. GlaxoSmithKline Biol, 1330 Rixensart, Belgium
- GlaxoSmithKline Biol, 1330 Rixensart, Belgium
- GlaxoSmithKline Biol, 1330 Rixensart, Belgium
- `GlaxoSmithKline Biol, 1330 Rixensart, Belgium,4.541692,50.709794`

ii. Bayer Anim Hlth GmbH, Leverkusen, Germany
- Bayer Anim Hlth GmbH, Leverkusen, Germany
- Bayer Anim Hlth GmbH, Leverkusen, Germany
- `Bayer Anim Hlth GmbH, Leverkusen, Germany,7.029976,51.048254`

iii. Univ Havre, 76058 Le Havre, France
- Univ Havre, 76058 Le Havre, France
- Univ Havre, 76058 Le Havre, France
- `Univ Havre, 76058 Le Havre, France,0.1077,49.4938`

`micro`, `meso`, `macro`
`meso`, `macro`, `longitude`, `latitude`

La grammaire locale étendue utilisée est présentée à la figure 7.12.

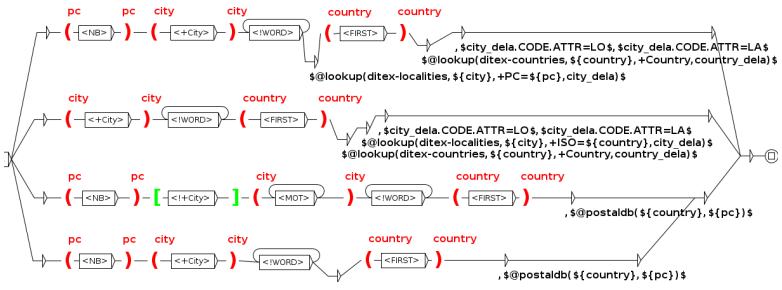


Figure 7.12 – Toponymes : grammaire locale étendue

Évaluation : Nous utilisons un corpus de 1000 adresses françaises extraites à partir de notices bibliographiques (chimie et pharmacologie). Nous comparons la performance

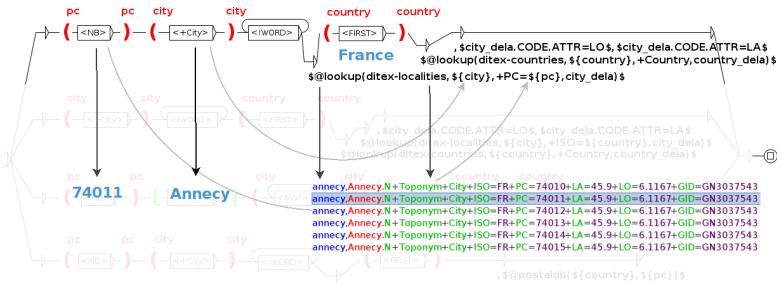


Figure 7.13 – Toponymes : analyse du première cas

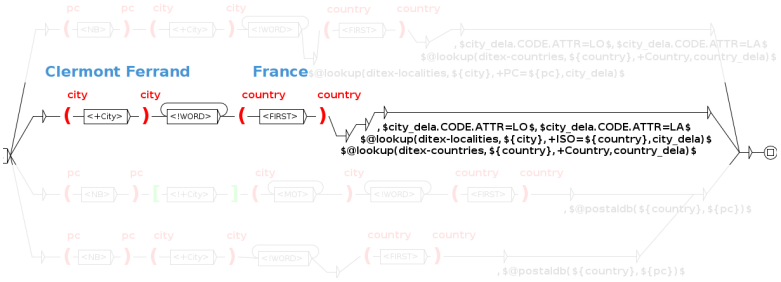


Figure 7.14 – Toponymes : analyse du deuxième cas

de la grammaire étendue par rapport à Pelias [1], un outil de géolocalisation fondé sur l'analyse de l'adresse à l'aide de champs aléatoires conditionnels (LibPostal) et sa recherche dans une base de données d'environ 500 millons d'adresses.

1. https ://github.com/pelias/pelias

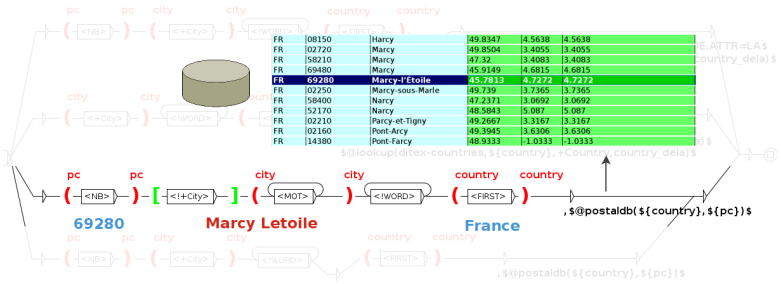


Figure 7.15 – Toponymes : analyse du troisième cas

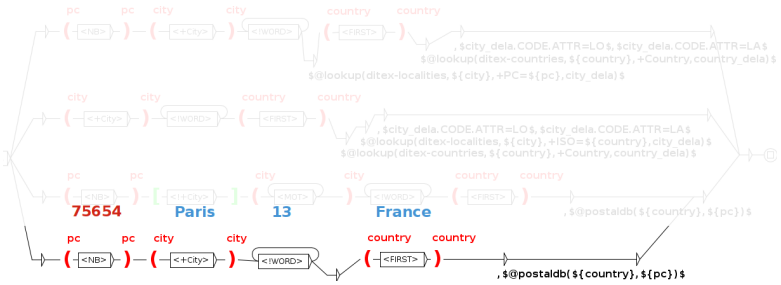


Figure 7.16 – Toponymes : analyse du quatrième cas

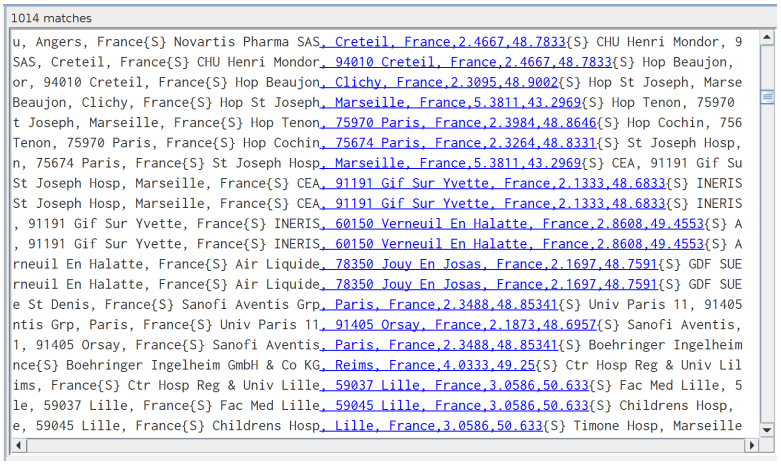


Figure 7.17 – Exemple d'adresses géolocalisées à l'aide d'une grammaire locale étendue

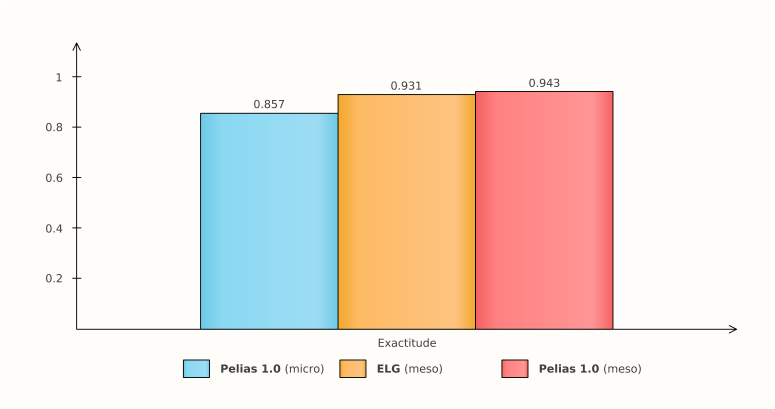


Figure 7.18 – Évaluation géolocalisation des adresses

Features	ELG	Pelias 1.0
Niveau du geocodage	Meso	Meso Micro
Adresses postales	✗	✓
Facile à adapter	✓	✗
Fouille de textes	✓	✗

Figure 7.19 – Comparaison grammaires locale étendue et Pelias