Grammaires locales

Est-il possible de concevoir un modèle abstrait et universel du langage ? Tel est le projet de Noam Chomsky qui jusqu'à présent n'a pas pu être atteint. En revanche, les travaux initiés par Maurice Gross fixent pour objectif de réaliser une description satisfaisante des langues naturelles, ceci en contraste avec les modèles formels des langages qui ne s'attardent pas sur les descriptions, à l'instar des grammaires génératives transformationnelles proposées par Chomsky.

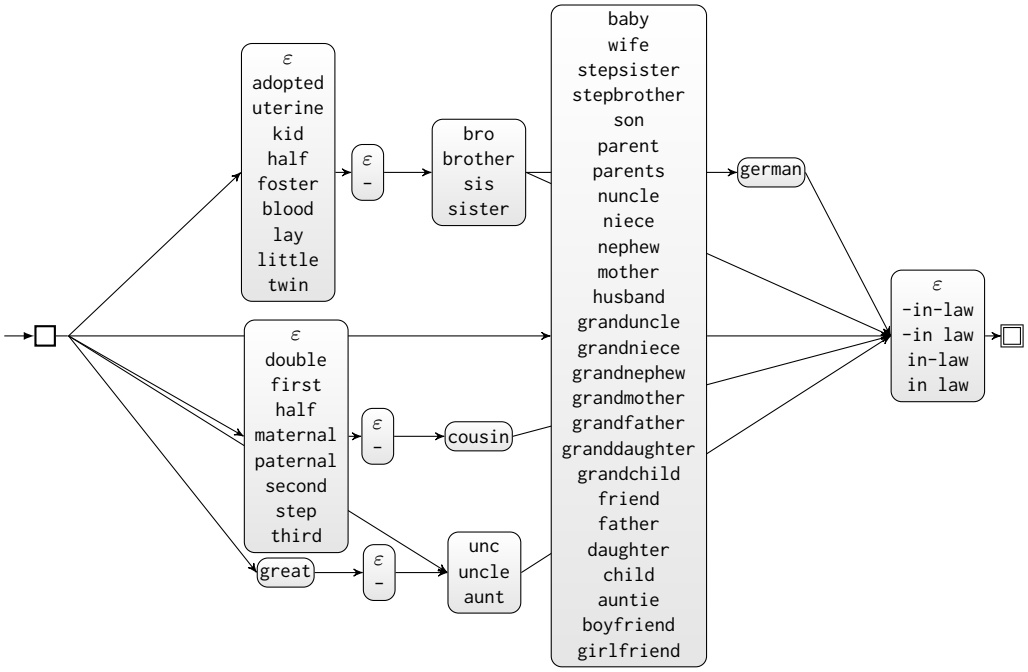
Pour parvenir à une description satisfaisante des langues naturelles Maurice Gross préconise, à la place de concevoir un modèle qui s'appliquerait à toutes les langues, plutôt un de nature strictement locale où la phrase se constitue comme l'unité élémentaire de base au niveau syntaxique et sémantique.

Ce chapitre est dédié aux grammaires locales (LGs). Les grammaires locales sont un formalisme, proche des automates finis, pour décrire, sous forme de graphes, des contraintes syntaxiques ou sémantiques d'une langue. Le concept de grammaire locale est issu des travaux menés par Maurice Gross (1993, 1996, 1997). Nous présentons d'abord un aperçu général des grammaires locales pour étudier ensuite leur définition et principales caractéristiques.

3.1 Aperçu général

Le graphe 3.1 représente une grammaire locale pour reconnaître de forme simplifiée quelques relations filiales (père, mère, fis, etc.) dans un texte en langue anglaise. La façon classique de représenter une grammaire locale est d'utiliser des graphes connectés

Silberztein (1993), lus de gauche à droite. Le graphe se lit de gauche à droite, les nœuds sont dénommés « *boîtes* », dont celle carrée le plus à gauche est désignée comme boîte de départ et celle en double carrée plus à droite est désignée comme boîte d'arrivée. Une liste complete de symboles utilisées pour schématiser les LGs est disponible au tableau I.



Graphe 3.1 – Reconnaissance de relations filiales

Une chaîne textuelle est reconnue, s'il existe un chemin (cf. définition 2.18) allant de la boîte initiale à la boîte finale où chacune des boîtes parcourues reconnaît une partie de la chaîne présentée en entrée. Le graphe 3.1 acceptera la phrase (8) mais non la (9) :

(8) Hemings was the half brother-in-law of Thomas Jefferson

(9) Hemings was the first African-American graduate of Vassar College

Les étiquettes d'entrée, composées de symboles appartenant à l'alphabet d'entrée, peuvent contenir des mots, des références à des classes de mots, des traits lexicaux sur les mots, le mot vide. De plus, les symboles d'entrée peuvent appartenir à un vocabulaire préétabli peuvent influencer l'analyse, certains concernent la mémorisation des symboles d'entrée ou de sortie : variables d'entrée, variables de sortie. D'autres permettent de lire des symboles d'entrée sans être « *consommées* » : contextes droits (pré-lecture) et contextes gauches (exclusion). En fin, d'autres permettent de changer l'unité minimale d'analyse : *mode morphologique*.

De plus, les arcs d'un graphe peuvent être chaînés pour former des boucles. Aussi, un graphe peut appeler d'autres sous-graphes, ce qui leur confère une grande puissance

expressive pour caractériser des structures lexicales dans un texte en langue naturelle. Nous reviendrons sur ces aspects plus tard.

Dans la pratique, les graphes peuvent s'organiser dans des bibliothèques de graphes facilitant par la suite leur réutilisation.

3.2 Définition

Les grammaires locales (Gross, 1993, 1997) sont un formalisme de description de règles syntaxiques ou sémantiques. Depuis leur conception, leur pertinence a été prouvée pour traiter plusieurs problèmes du traitement automatique des langues (TAL) liés à la représentation succincte de descriptions linguistiques fines : découpage de phrases, traitements morphologiques, étiquetage, analyse syntaxique de surface, analyses transformationnelles, résolution des ambiguïtés, extraction de l'information, entre autres (Nam et Choi, 1997, Friburger, 2002, Constant, 2003, Traboulsi, 2004, Laporte, 2005, Geierhos et al., 2008, Martineau et al., 2011, Krstev et al., 2013, Ezzat, 2014).

Le formalisme de LGs est proche de celui des automates à états finis et plus largement associée à des réseaux de transitions récursifs (RTNs) (Woods, 1970, Bates, 1978) et des RTNs comportant des sorties (Sastre et Forcada, 2009). Bien qu'il soit aussi fréquemment assimilé au concept de transducteur fini, de part la capacité d'une grammaire locale à gérer des sorties, observons que même si la notion de sortie est comparable à celle de transduction, les grammaires locales et les transducteurs ne sont pas toujours équivalents. En effet, comme souligné par Blanc (2006, p. 62), l'application d'un transducteur sur un texte en entrée produit comme résultat un nouveau texte composé par la concaténation des sorties, tandis qu'avec une grammaire locale, le résultat peut être en plus égal au texte en entrée combiné avec les séquences de sortie.

En général dans la littérature, la notion de grammaire locale renvoie à des définitions et des mises en œuvre différentes. D'une part, par la variété de types de grammaires locales concernées, ces type peuvent être divisées selon l'analyse effectuée : lexicale (flexion, prétraitement), syntaxique (locale ou structurelle) ou transformationnelle. D'autre part, par les appellations utilisées pour y faire référence, entre autres : transducteurs finis (Fairon et Watrin, 2003, Sætre, 2003, Ranchhod et al., 2004, Kevers, 2006), automates finis (Traboulsi, 2005), automates lexicaux (Blanc et Dister, 2004), automates syntaxiques (Monnier et al., 2003). Finalement, la notion de grammaire locale dépend aussi de l'outil informatique utilisée, nous pouvons distinguer deux groupes :

- Ceux qui ne disposent pas de la capacité de concevoir des grammaires locales à l'aide d'une interface graphique, mais sous la forme d'ensembles de règles de dérivation ou de tableaux de transition d'état. Nous citons par exemple, les grammaires locales du type `dag2dag` (Sagot et Boullier, 2008) de SXPipe(Sagot et Boullier, 2005), exprimées dans un langage proche de la forme de Backus–Naur (BNF) et analysées à l'aide du système Syntax (Boullier et Deschamp, 1991). De plus, des outils comme

OpenFst (Allauzen et al., 2007), pouvant manipuler des RTNs aplatis [1] (Allauzen et Riley, 2012).

- Ceux qui offrent la possibilité de développer des grammaires locales à l'aide d'interfaces graphiques, en faisant appel à l'utilisation de ressources linguistiques, comme des dictionnaires électroniques (Maurel, 1993, Courtois et al., 1997, Chrobot et al., 1999) qui décrivent la syntaxe et la morphologie des entrées lexicales et de leurs attributs sémantiques. Ainsi que d'avoir des sorties, d'utiliser des variables, c'est-à-dire, des registres qui stockent des sous-séquences d'une séquence en entrée, et d'établir de contraintes sur les variables, sous la forme d'opérateurs primitifs. Nous énumérons, par exemple, Intex (Silberztein, 1994), Unitex (Paumier, 2003a), NooJ (Silberztein et Tutin, 2005) ou Outilex (Blanc et al., 2006).

Dans notre recherche une grammaire locale concerne des grammaires non contextuelles (cf. sous-section 2.4.1) qui décrivent des langues naturelles. Elles sont représentées par des graphes orientés *lexicalisés* et récursifs. En plus, les grammaires locales peuvent être dérécursivées (cf. section 4.5) et modélisés par des machines à états finis.

Les grammaires locales sont aussi enrichies par des registres et des contraintes qui ne font pas partie du modèle des automates finis, mais qu'en vue de traiter certaines opérations classiques des automates peuvent, avec des ajustements, être considérées comme des symboles d'entrée dépourvus de sens [2].

Finalement, bien que les grammaires locales sont restreintes à décrire des langages non contextuels (cf. section 2.4), la possibilité d'ajouter des sorties et de contraintes leur donne la capacité de reconnaître des langages contextuels (cf. section 3.5.8) et de réaliser des opérations transformationnelles.

Malgré la puissance inhérente aux automates finis (FSAs), ceux-ci ne possèdent pas de mécanisme leur permettant faire appel à d'autres FSAs. Pour palier ce problème, tout en conservant certains avantages des modèles d'automates finis classiques, les grammaires locales sont représentées par des RTNs (Woods, 1970, Bates, 1978).

1. L'aplatissement ou la dérécursivation d'un RTN fait référence au processus de remplacement de transitions qui désignent l'appel à un sous-réseau. Afin d'obtenir un automate fini équivalent, il est necessaire de substituer les transitions étiquetées par des symboles non-terminaux, par une copie exclusive du sous-réseau appelé.

2. Par exemple, un registre qui fait référence au nom d'une variable est considéré comme un symbole d'entrée terminale. C'est lors de l'analyse syntaxique que l'algorithme en charge donne le sens au symbole. Par exemple, en associant un registre en mémoire au prochain symbole lu.

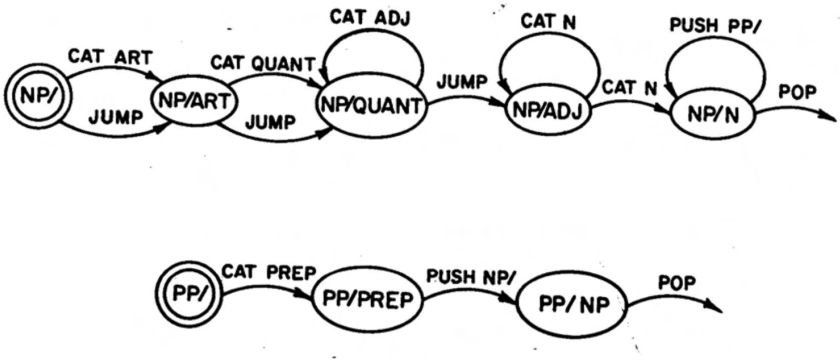


Figure 3.1 – Exemple d'un RTN pour la reconnaissance de phrases nominales (Bates, 1978, p. 193)

Un réseau de transitions récursif est une grammaire non contextuelle où on admet que les membres droits des règles soient des automates finis au lieu de simples séquences de symboles, lorsqu'ils modélisent des grammaires locales ils sont définis comme un graphe de nœuds et d'arêtes (les liens entre les nœuds). Les arêtes sont étiquetées avec des symboles de sortie ; les nœuds, l'alphabet d'entrée, peuvent contenir des mots, dont le mot vide epsilon (ε), des traits lexicaux sur les mots, ou faire appel à un sous-réseau (état non-terminal). Le graphe est lu de gauche à droite, c'est-à-dire, l'un des nœuds est désigné comme nœud initial et un autre comme nœud final. Une chaîne textuelle est reconnue, si un chemin existe allant du nœud initial au nœud final où chacun des nœuds parcourus accepte une sous-partie de la chaîne présentée en entrée. Les arêtes d'un RTN peuvent être connectées, pour former des cycles dans le graphe. Les réseaux sont alors nommés récursifs, une caractéristique, qui leur confère une grande puissance expressive pour caractériser des structures lexicales dans un texte en langue naturelle.

3.3 Modélisation

Formellement, une grammaire locale peut être définie sous la forme d'automate fini comme étant un 7-tuple $G = (Q, \Sigma, \Gamma, I, F, E, s)$, où :

- Q est un ensemble fini d'états
- Σ est l'alphabet d'entrée constitué d'un ensemble fini de symboles
- Γ est l'alphabet de sortie constitué d'un ensemble fini de symboles
- I est l'ensemble sous-initial des états qui étiquettent des transitions (*soit les appels à des sous-graphes*)
- $F \subseteq Q$ est l'ensemble des états accepteurs ($F \neq \varnothing$)

- $E \subseteq Q \times (\Sigma \cup \varepsilon) \times (\Gamma \cup \varepsilon) \times Q$ est un ensemble fini de transitions

- $s \in Q$ est l'unique état initial (*du graphe principal*)

L'automate de la figure 3.2 est équivalent au graphe 3.1, cependant certaines transitions ont été supprimes pour améliorer la clarté.

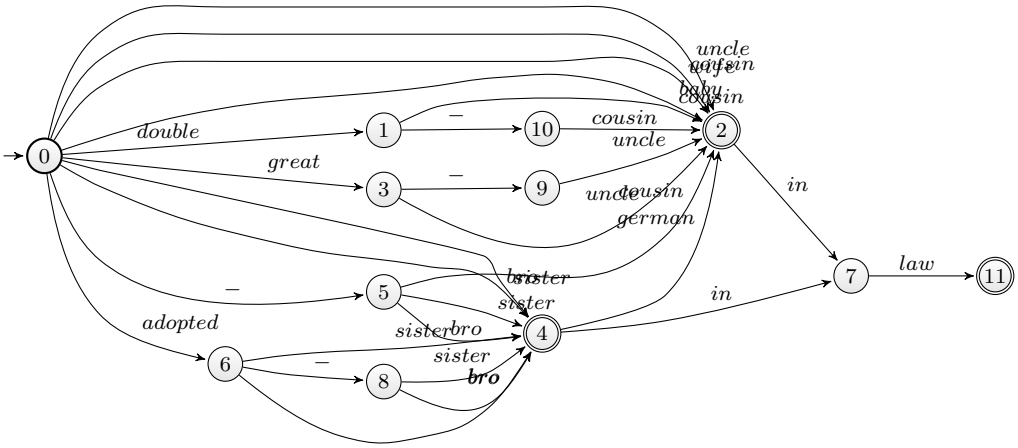


Figure 3.2 – Reconnaissance de relations filiales

Quelques observations

Observons qu'une LG peut être vue comme un RTN avec des sorties et que certains symboles non-terminaux des étiquettes d'entrée d'une LG ont une sémantique spéciale, par exemple, pour faire référence à l'appel à un sous-graphe, d'autres à des masques lexicaux, ceci dit, ces symboles non-terminaux peuvent être considérés comme terminaux, c'est-à-dire, dépourvus de sens, et par ce fait, permettre l'application directe, ou avec peu de changements, des algorithmes classiques des automates finis. Notons aussi :

(i) Une LG sans sorties ($\Gamma = \varnothing$), c'est-à-dire qui produit le mot vide epsilon (ε), est équivalente à un RTN.

(ii) Une LG dérécursivée, c'est-à-dire sans cycles, qui produit comme résultat la concaténation des sorties est équivalente à un FST.

(iii) Une LG dérécursivée et sans sorties est équivalente à un FSA.

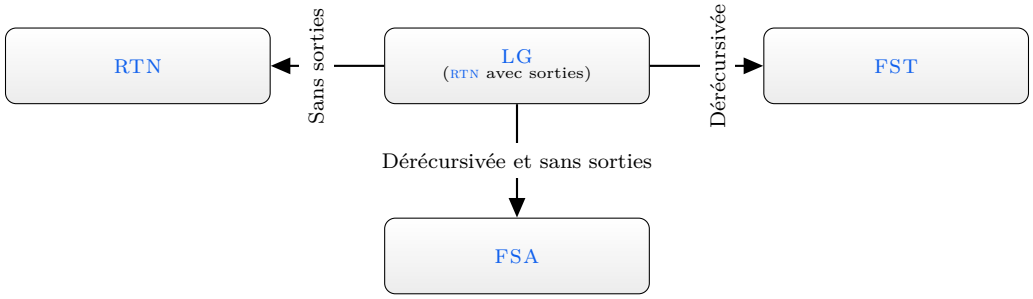


FIGURE 3.3 – Possibles équivalences entre les LGs, RTNs, transducteurs finis (FSTs) et FSAs

3.4 Outils existants

Le formalisme des grammaires locales à été implémenté par différents outils pour traiter des problèmes liés au traitement automatique des langues. Par la suite, nous passons brièvement en revue certains de ces outils.

3.4.1 INTEX

INTEX(Silberztein, 1993, 1994, 2000) est un environnement de développement de dictionnaires électroniques, de grammaires et de lexiques-grammaires pour le traitement de corpus : recherche d'expressions dans des textes, indexation des motifs trouvés, concordances et statistique des résultats. INTEX est à l'origine de plusieurs caractéristiques utilisées par ses successeurs :

- Les opérations de prétraitement du texte dont le découpage en unités lexicales, le découpage en phrases, la normalisation des formes non-ambiguës, et l'application des dictionnaires électroniques du LADL (DELA) (Courtois et Silberztein, 1990, Courtois et al., 1997).

- L'interface graphique pour construire des graphes. Notamment par l'utilisation de boîtes (les nœuds du graphe) pour associer les étiquettes de transition de l'automate sous-jacent. Ceci en reprenant les conventions proposées originalement par Maurice Gross (Gross, 1987).

- L'utilisation de méta-symboles pour reconnaître, ou non, des classes de caractères, par exemple ⟨MOT⟩, ⟨MAJ⟩, ⟨!DIC⟩ ; ainsi que des masques lexicaux afin de faire référence aux informations fournies par les dictionnaires, par exemple, ⟨N⟩, ⟨V⟩, ⟨pomme.N⟩.

- Les politiques pour traiter les sorties de la grammaire : soit par remplacement (mode *replace*) où les sorties remplacent les entrées reconnues, soit par combinaison (mode *merge*) où les sorties produites sont insérées dans le texte. Également les

politiques permettant de sélectionner le mode de reconnaissance en donnant priorité aux longueurs des séquences reconnues : soit les plus courtes, soit les plus longues.

INTEX été développé en C++ et distribué avec une licence propriétaire sans possibilité de connaître les détails sur les algorithmes utilisés.

3.4.2 UNITEX

UNITEX [1] Paumier (2003a), est un environnement logiciel libre dédié à l'analyse du langage naturel. Il a été développé par Sébastien Paumier et l'équipe d'informatique linguistique du Laboratoire d'Informatique Gaspard-Monge (LIGM). À partir de la version 3.0 UNITEX est devenu UNITEX/GRAMLAB et continue à être développé au tour d'une communauté regroupant des linguistes et des développeurs. UNITEX débute comme une alternative libre à INTEX.

3.4.3 NOOJ

NOOJ [2] (Silberztein, 2004, Silberztein et Tutin, 2005) représente l'évolution d'INTEX.

3.4.4 OUTILEX

L'analyseur syntaxique d'OUTILEX [3] (Blanc et al., 2006) est de la famille des analyseurs Earley.

3.4.5 SXPIPE

les grammaires locales du type dag2dag (Sagot et Boullier, 2008) de SXPIPE [4] (Sagot et Boullier, 2005), exprimées dans un langage proche de la BNF et analysées à l'aide du système SYNTAX (Boullier et Deschamp, 1991).

3.4.6 OPENFST

OPENFST [5] (Allauzen et al., 2007) est une bibliothèque logicielle libre et modulaire orientée vers la manipulation des transducteurs pondérés (WFSTs). Elle fournit des algorithmes efficaces en temps et en espace pour la construction, la combinaison, l'optimisation et la recherche sur les automates. Mise à part un cœur d'opérations

1. http://unitexgramlab.org/
2. http://www.nooj-association.org
3. http://igm.univ-mlv.fr/~mconstan/outilex
4. http://alpage.inria.fr/~sagot/sxpipe.html
5. http://www.openfst.org

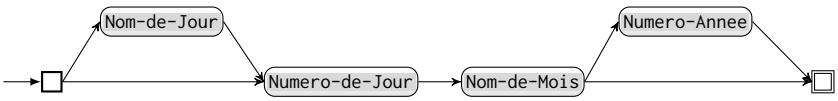
classiques, OpenFst met à disposition des *extensions* avec de nouveaux algorithmes et des automates équivalents aux WFSTs, dans ces derniers, une extension permettant de manipuler des transducteur à pile (PDT) est fournie. Une des opérations disponibles sur les PDTs, est celle de l'aplatissement (Allauzen et Riley, 2012, p. 8) d'un RTN, c'est-à-dire, le remplacement des transitions étiquetées par des symboles non-terminaux qui désignent l'appel à un sous-graphe par une copie exclusive du même graphe, afin d'obtenir un PDTs équivalent.

3.5 Caractéristiques principales

Nous pressentons quelques caractéristiques principales des grammaires locales, non sans rappeler que dans le contexte de notre travail, nous nous concentrons sur les graphes syntaxiques.

3.5.1 Faire appel à des sous-graphes

Les grammaires locales comportent la notion de sous-graphe. Cette notion permet de décomposer une analyse en différentes analyses plus simples qui s'attachent chacune à la reconnaissance d'une sous-partie du motif recherché. Considérons par exemple une version simplifiée de la reconnaissance de dates. Elle peut être exprimée par la reconnaissance des quatre sous-motifs suivants : Nom de Jour, Numéro de Jour, Nom de Mois, Numéro d'Année.



Graphe 3.2 – Appels de sous-graphes pour chaque sous-motif

Le graphe 3.2 fait appel à un sous graphe pour chacun des constituants potentiels d'une date complète. Comme nous le constatons dans la figure, certains d'entre-eux sont optionnels. Les sous graphes peuvent être de simples graphes ou bien, eux-mêmes faire appel à d'autres sous-graphes. Il serait possible par exemple de créer un sous-graphe pour traiter le sous-motif lorsqu'il est écrit en chiffre et un autre pour le traitement de la version en lettres. Cette notion de sous-graphe « en réseau » proche de celle des RTNs rend possible une représentation aisée et structurée de motifs complexes.

3.5.2 Faire référence à des masques lexicaux

Dans une grammaire locale, il est possible d'utiliser des masques lexicaux. Ces masques se présentent sous la forme d'une requête à des dictionnaires entourée de chevrons (<>). Le format de cette requête est conforme au format des informations présentes dans les entrées de dictionnaires. Par exemple, ⟨N+Hum:fp⟩ reconnaît un nom humain féminin

pluriel. Il est également possible d'utiliser le symbole ∼ pour indiquer qu'un trait ne doit pas figurer dans l'entrée. ⟨N∼ Hum⟩ fait référence à un nom non humain.

3.5.3 Faire référence à des méta-symboles

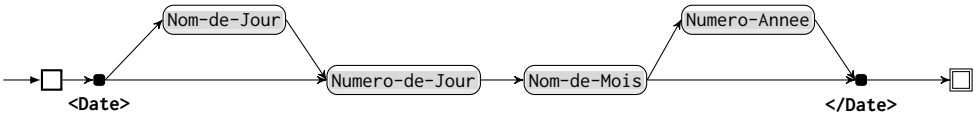
Les grammaires locales disposent de méta-symboles capables de représenter certaines séquences de caractères selon leur structure ou leur appartenance ou non à un dictionnaire. La liste de méta-symboles est disponible au tableau II.

En combinant un masque lexical ⟨N+Given⟩ et un méta-symbole ⟨FIRST⟩ (reconnaît un mot qui commence par une majuscule) le graphe 3.3 constitue une grammaire basique capable de reconnaître des noms de personnes.

Graphe 3.3 – Graphe basique de reconnaissance de noms de personne

3.5.4 Gérer des sorties

Les grammaires locales ne se limitent pas à pouvoir lire le texte en entrée. Elles permettent également d'écrire dans le texte traité et de produire un nouveau texte enrichi. Cette fonctionnalité est très utilisée pour annoter un texte.

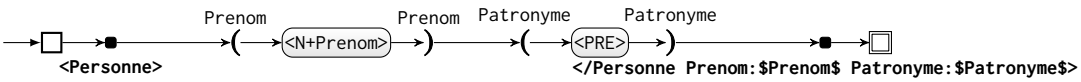


Graphe 3.4 – Écriture de sorties :balisage de dates.

Le graphe 3.4 reprend celui présenté au graphe 3.2. Il comporte en plus deux sorties ⟨E⟩/⟨Date⟩ et ⟨E⟩/⟨/Date⟩ qui entourent la date reconnue. L'application de cette grammaire ajoute donc des balises autour de toutes les occurrences de date trouvées dans le texte. Un nouveau texte annoté peut être ainsi généré.

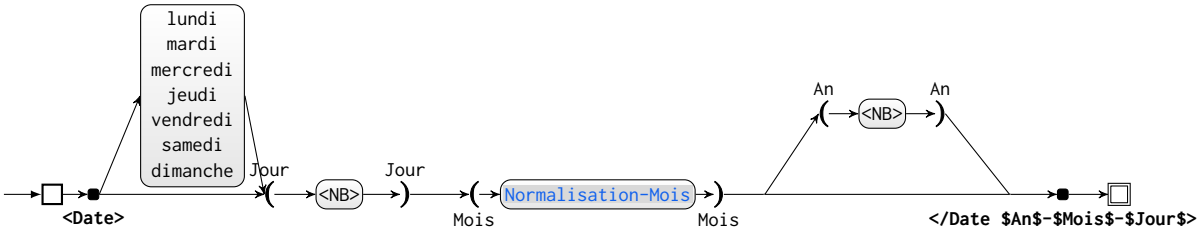
3.5.5 Stocker des variables locales

Dans certaines applications il est souhaitable de produire une annotation qui ne se limite pas à une annotation globale du motif trouvé. Le graphe 3.3 de reconnaissance de nom de personne peut être enrichi en ce sens afin de produire une annotation qui distingue les deux constituants : prénom, patronyme. Dans ce but, nous utilisons deux variables qui captent chacun d'eux.



Graphe 3.5 – Annotation de noms de personne

Le graphe 3.5 stocke le patronyme et le prénom dans des variables et délimite le nom complet reconnu au moyen de balises d'étiquette « Personne ». La balise fermante comporte la séquence `Prénom:$Prenom$ Patronyme:$Patronyme$` dans laquelle les symboles $ sont utilisés pour indiquer que nous voulons afficher le contenu de la variable qu'ils entourent. Ainsi, si dans un texte nous rencontrons le nom *Emmanuel Macron* le graphe produit : `<Personne>Emmanuel Macron</Personne Prénom:Emmanuel Patronyme:Macron>`. Dans ce graphe les variables mémorisent des éléments issus du texte analysé en entrée. Il est également possible de stocker des données issues des sorties d'une grammaire.



Graphe 3.6 – Normalisation de dates

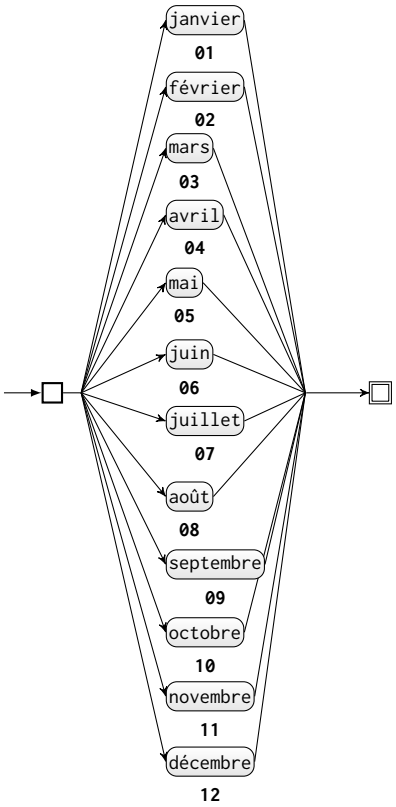
Le graphe 3.6 reconnaît une date comportant au moins le numéro de jour et le mois : la reconnaissance des noms de jour et de l'année sont optionnels. La date reconnue est balisée ; la balise fermante comporte la forme normalisée de la date reconnue. Le graphe utilise le méta-symbole ⟨NB⟩ pour reconnaître le numéro du jour et de l'année qui sont captés dans des variables qui mémorisent des entrées issues du texte (parenthèses rouges). Il fait appel à un sous-graphe *Normalisation_Mois* (graphe 3.7) qui transforme le nom du mois en son numéro. Le résultat de cette transformation est mémorisé dans la variable *Mois* qui stocke une sortie du graphe (parenthèses bleues). Enfin les variables contenant es parties normalisées de la date sont utilisées dans la balise fermante afin de produire la forme normalisée complète souhaitée. Ainsi l'application de la grammaire à la date *mercredi 18 octobre 2017* produit : `<Date>mercredi 18 octobre 2017</Date 2017-10-18>`.

3.5.6 Utiliser des filtres morphologiques

Nous avons évoqué précédemment les notions de masques lexicaux et de méta-symboles. Les recherches effectuées grâce à eux peuvent être affinées (restreintes) en leur adjoignant un filtre morphologique entre double chevrons `<<>>` qui ajoute une contrainte supplémentaire. Il est possible par exemple de rechercher des adverbes qui se terminent par *ment* `<ADV><<ment$>>` (le dollar représente la fin du mot), des verbes qui commencent par l'un des préfixes *re*, *ré*, *auto* `<V><<^(auto|re|ré)>>` (le symbole ^ représente le début du mot et le pipe « | » le *ou* logique) ou bien encore des mots en majuscules qui finissent par *s* `<UPPER><<s$>>`.

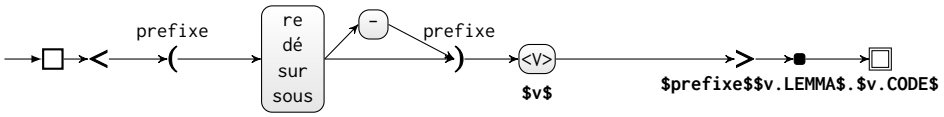
3.5.7 Définir l'unité minimale d'analyse

Pour analyser un texte de manière efficace en terme de temps d'exécution il est habituel de le décomposer en une unité plus grande que le caractère : le token. Un token est



Graphe 3.7 – Sous-graphe de normalisation de mois

soit une suite de lettres d'une longueur supérieure ou égale à 1, soit un caractère seul. Le texte est alors vu comme une suite de numéros de tokens. Les tokens proprement dits sont listés dans un lexique construit au cours de la lecture du texte. L'indice d'un token dans cette liste est le numéro utilisé dans la représentation du texte évoquée précédemment. Dans la plupart des analyses l'utilisation du token comme unité minimale est satisfaisante. Cependant, il est utile de pouvoir avoir accès au contenu d'un token pendant l'analyse. Les filtres morphologiques permettent de filtrer les tokens selon leur contenu mais ne permettent pas d'analyser des suites de lettres dynamiquement construites pendant l'analyse. Pour avoir un accès plus fin au contenu d'un token il faut utiliser une analyse caractère par caractère. Ceci est rendu possible au sein des grammaires locales en utilisant le mode morphologique.



Graphe 3.8 – Mode morphologique :reconnaissance et étiquetage de formes verbales préfixées

Le graphe 3.8 permet de détecter dans un texte des verbes ayant comme préfixe *re*, *dé*, *sur*, *sous* à partir de formes non préfixées déjà présentes dans le dictionnaire.

Les chevrons violets indiquent la zone de la grammaire dans laquelle l'analyse est effectuée caractère par caractère. La première boîte reconnaît les préfixes recherchés et mémorise celui effectivement trouvé. La deuxième boîte qui reconnaît un verbe (<V>) comporte en dessous la variable v (variable morphologique). Cette dernière est utilisée afin d'étiqueter entre accolades la forme verbale reconnue. Cette étiquette est construite d'une part, grâce à l'adjonction du préfixe au lemme de la forme verbale non préfixée, d'autre part grâce à la récupération des informations syntaxico-sémantique de cette même forme dont hérite ainsi la forme verbale préfixée. Si par exemple, nous rencontrons dans un texte les formes verbales *sousalimentait* ou *sous-alimentait* et si le verbe alimenté est présent dans le dictionnaire utilisé, la grammaire étudiée produit respectivement les annotations *sousalimentait{sousalimenter.V+z1 :I3s}*, *sous-alimentait{sous-alimenter.V+z1 :I3s}*.

3.5.8 Capacité à reconnaître des langages contextuels

Le principe pour concevoir des grammaires locales capables de reconnaître des langages contextuels (cf. section 2.5), repose sur le fait, tel que décrit par Silberztein (2015), que *tout langage est inclus dans un langage rationnel*. Pour illustrer ceci, nous reprenons l'exemple de deux langages $\mathcal{L}_1$ et $\mathcal{L}_2$ disponible dans Donabédian et al. (2013, p. 8) et Silberztein (2015, p. 229).

Le langage $\mathcal{L}_1$ est égal à $\{a^n b^n c^n \mid n > 0\}$, autrement dit, celui des mots qui sont formés par une suite de n lettres a, suivi par n lettres b, suivi par n lettres c. $\mathcal{L}_1$ est alors contextuel (cf. section 2.5), les mots qui appartiennent à ce langage sont $\{abc, aabbcc, aaabbbccc, aaaabbbbcccc, ...\}$.

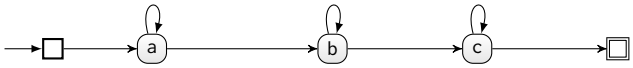
Le langage $\mathcal{L}_2$ est égal à $\{a^+b^+c^+\}$, c'est-à-dire, celui des mots qui sont formés par une suite arbitraire de lettres a, suivi par une suite arbitraire de lettres b, suivi par une suite arbitraire de lettres c. $\mathcal{L}_2$ est régulier et les mots qui appartiennent au langage sont $\{a, b, ab, abc, aa, ab, bb, aab,$ abc $, aabb, \ldots,$ aabbcc $, \ldots,$ aaabbbccc $, \ldots\}$.

Il n'est pas difficile de constater que les mots engendrés par le langage contextuel $\mathcal{L}_1$, sont aussi des mots du langage régulier $\mathcal{L}_2$. Nous pouvons généraliser ce fait en prenant en compte que pour un alphabet Σ, les mots engendrés par n'importe quel type de langage L dans l'hiérarchie de Chomsky : régulier (type 3), non contextuel (type 2), contextuel (type 1) ou sans restriction (type 0), sont inclus dans Σ^*. Du fait que Σ^* est le langage rationnel formé par l'ensemble de tous les mots sur Σ, nous avons alors que $|L \cap \Sigma^*| = |L|$.

Étant donné que tout langage est inclus dans un langage rationnel, la stratégie mise en place dans une grammaire locale pour reconnaître $\mathcal{L}_1$, le langage contextuel, est tout d'abord de reconnaître $\mathcal{L}_2$, le langage régulier, et par la suite d'exclure les séquences qui n'appartiennent pas à $\mathcal{L}_1$ [1]. Pour la reconnaissance de $\mathcal{L}_2 = \{a^+b^+c^+\}$, la première

1. Un théorème connexe abordé dans Grune (2014) et prouvé par Ginsburg et al. (1967) est que *tout langage sans restriction (type 0), peut être engendré en intersectant deux langages non contextuels (type 2) et en appliquant un homomorphisme d'effacement au résultat :* $\mathcal{L}_0 = h_e(\mathcal{L}_2 \cap \mathcal{L}_2)$

partie, il est possible d'utiliser une grammaire locale comme celle du graphe 3.9, dont l'automate équivalent est celui de la figure 3.4.



Graphe 3.9 – Grammaire locale qui reconnaît $\mathcal{L}_2 = \{a^+b^+c^+\}$

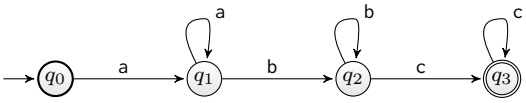
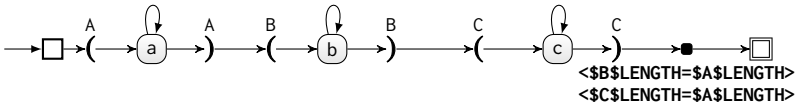


Figure 3.4 – Automate équivalent au graphe 3.9

Une fois une séquence $w \in \mathcal{L}_2$ reconnue, il est requis de définir la façon de vérifier si w appartient également à $\mathcal{L}_1 = \{a^n b^n c^n \mid n > 0\}$. Dans cet exemple, il est donc nécessaire de vérifier si w est bien formé par une suite de n lettres a, suivi par n lettres b, suivi par n lettres c. Pour réaliser une telle opération, les grammaires locales doivent être dotées de variables d'entrée, des registres qui stockent des sous-séquences d'une séquence en entrée, ainsi que de contraintes, sous la forme d'opérateurs, permettant de vérifier si deux variables d'entrée ont la même longueur. Par exemple, dans NooJ, la longueur d'une variable d'entrée peut être obtenue à l'aide de l'opérateur LENGTH et leur équivalence avec l'opérateur égal (=) (Silberztein, 2015, p. 230). La grammaire locale NooJ du graphe 3.10, dont l'automate équivalent est celui de la figure 3.5 [1], illustre comment le graphe 3.9 peut être doté de variables et de contraintes afin de vérifier si $w \in \mathcal{L}_1$.



Graphe 3.10 – Grammaire NooJ qui reconnaît $\mathcal{L}_1 = \{a^n b^n c^n \mid n > 0\}$

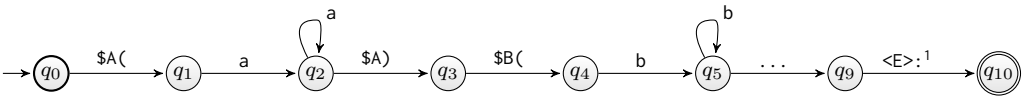


Figure 3.5 – Automate équivalent au graphe 3.10

Le graphe 3.10 constitue un des exemples de la façon dont une grammaire locale peut reconnaître un langage contextuel (ici $\mathcal{L}_2$) tout en reconnaissant un langage régulier (ici $\mathcal{L}_1$) et en excluant les séquences qui n'appartiennent pas au langage contextuel. En général, cette approche peut s'étendre en gardant le principe d'utiliser des grammaires

1. En raison de l'espace réduit, la transition $q_9 \xrightarrow{\text{<E> :<\$B\$LENGTH=\$A\$LENGTH><\$C\$LENGTH=\$A\$LENGTH>}} q_{10}$ est désignée de la forme $q_9 \xrightarrow{\text{<E>}:1} q_{10}$.

qui peuvent être analysées plus efficacement, telles que les grammaires régulières et non contextuelles, munies des contraintes pour exclure les séquences qui n'appartiennent pas à un langage défini par une grammaire contextuelle.

3.6 Grammaires locales et extraction de l'information

Les approches d'extraction automatique d'information basées sur la construction de grammaires locales modélisent le problème sous forme ascendante, c'est-à-dire en tenant compte d'abord les dépendances entre groupes voisins de mots et ensuite entre les groupes plus distants. En effet, il est souhaitable d'analyser tout d'abord les contextes immédiats des mots afin d'identifier les éléments d'indice qui peuvent, ou non, déclencher l'identification des motifs recherchés pour étudier ensuite les contextes plus éloignés ou complexes.

Les contextes trouvés peuvent alors se diviser en deux groupes : **internes** ou **externes**. Les contextes internes font partie des motifs recherchés et constituent alors une *preuve interne* pour l'identification, en revanche, les contextes externes n'appartient pas aux motifs recherchés et représentent alors une *preuve externe* pour l'identification.

Une fois les contextes caractérisés, ils sont utilisés pour construire des graphes simples au niveau des mots, aussi nommés graphes de premier niveau, qui éventuellement sont appelés par des graphes de niveau supérieur, capables d'identifier des séquences de plus en plus complexes.

Cette approche est utilisée dans les travaux de Sætre (2003) pour le développement d'un système d'extraction automatique d'information à partir de textes biomédicaux. En partant d'un ensemble de graphes qui reconnaissent des motifs simples (par exemple, des noms de gènes), des graphes d'un niveau supérieur sont construits pour décrire des séquences de plus en plus complexes (par exemple, des interactions entre les gènes).

Les grammaires locales sont aussi exploitées dans le cadre d'un système d'indexation de noms de personnalités développé par Fairon et Watrin (2003). À l'opposé d'autres travaux, le corpus à analyser n'est pas prédéfini, mais il s'agit d'un flux continu de textes issue du web qui est généré par un service tiers [1].

Les grammaires développées utilisent un dictionnaire de noms de personnes et autre de noms de professions et décrivent des motifs qui se trouvent autour du nom d'une personnalité. Ces motifs doivent nécessairement [2] être accompagnés par des informations bibliographiques comme l'age ou la profession, par exemple, comme dans : *Mr. X, director of Y – The director of Y, Mr. X*, ou X est un nom de personne et Y le nom d'une organisation. À part la tâche d'identification, les graphes sont utilisés pour annoter les entités reconnues à l'aide de balises. Une fois le texte annoté, une

1. GlossaNet, Fairon (1998) : http://glossa.fltr.ucl.ac.be
2. La démarche choisie par les auteurs et mise en œuvre par le système est alors de privilégier la précision en donnant moins d'importance au taux de rappel

dernière phase, basée sur l'analyse des collocations des motifs extraits, est entamée afin de synthétiser et d'identifier les clés d'indexation les plus pertinentes.

Les travaux de Aubin et Barbier (2003) autour de l'extraction d'information à partir de documents contractuels font appel à l'utilisation des grammaires locales dans le but de remplir une base de données. Ces grammaires servent à identifier les dates ainsi que les noms de personnes ou d'organismes qui sont cités dans des contrats passés entre un centre de recherche public français et ses partenaires. Le processus de construction de grammaires commence par mettre en évidence des motifs lexicaux qui sont fréquemment associés aux entités recherchées. Il est suivi par la constitution de dictionnaires électroniques spécialisés qui sont ultérieurement utilisés par des graphes nécessaires à l'extraction des motifs recherchés. Même si l'étude ne révèle pas de résultats quantitatifs concernant les performances, les auteurs constatent des *taux de bruit relativement faibles* ce qui s'avère pertinent pour la tâche requise.

Poibeau et al. (2003) utilisent les grammaires locales comme partie d'un système de reconnaissance d'entités nommées dans des documents multilingues. Pour cela, des dictionnaires de noms propres et des grammaires locales sont conçues spécifiquement pour chacune des langues visées, les graphes sont développés en prenant en compte les mots et les contextes qui constituent des déclencheurs pour l'identification d'une entité. Une fois un texte analysé par les grammaires, le traitement se poursuit par un module destiné à améliorer les performances globales par la reconnaissance de nouvelles entités jusqu'alors inconnues et négligées.

La technique pour étendre l'ensemble des expressions identifiées est fondée sur l'apprentissage supervisé (Cornuéjols et Miclet, 2011, p. 41). Tandis que la connaissance du système est représentée par les lexiques et les grammaires, l'ensemble d'exemples d'apprentissage est constitué par les entités, inconnues au départ, qui ont été positivement identifiées par l'application des grammaires locales. Par exemple, si *Newton* est un mot inconnu au départ et si lors du passage des grammaires sur l'expression *Sir Isaac Newton*, il est reconnu comme étant un nom de personne, alors cette information est memorisée. La méthode d'apprentissage utilisera cette dernière pour construire une *fonction de prédiction* permettant de déterminer que les occurrences isolées du mot *Newton* peuvent également être étiquetées comme un nom de personne.

Suite à l'application des grammaires locales et à l'identification supplémentaire des entités qui ont été négligées, un dernier module est utilisé pour, à partir d'un contexte donné, résoudre les conflits d'étiquetage. En suivant l'exemple précédent, les occurrences isolées du mot *Newton* qui sont étiquetées erronément comme un nom de personne dans la phrase *The Second Law of Newton* sont corrigées grâce au mot *Law*.

Dans Saidi (2004), les grammaires locales sont utilisées pour l'identification et l'extraction de motifs dans des annonces de séminaires, tels que le lieu, l'adresse, la date ou l'heure.

Nakamura (2004) propose d'analyser automatiquement des bulletins boursiers au moyen de grammaires locales. L'approche utilisée consiste à faire une caractérisation linguistique fine (*analyse linguistique, syntaxico-sémantique et lexicale*) des schémas de phrase pour ensuite transposer chaque composant de cette description sous la forme

d'une grammaire locale.

Dister et al. (2004) décrivent une technique de repérage et d'extraction de mots inconnus permettant leur indexation dans un dictionnaire électronique. La méthode est développée dans le cadre du GlossaNet (Fairon, 1998), un service proposant de créer des corpus dynamiques à partir du web [1]. En premier lieu, une grammaire permettant d'extraire les formes commencent par une majuscule et qui ne sont pas reconnues par un dictionnaire des mots simples permet de repérer les mots à analyser. Ensuite, une caractérisation et classification manuelle des types de mots inconnus permet la création de nouveaux dictionnaires électroniques ou la mise à jour des dictionnaires déjà existants.

Ranchhod et al. (2004) présentent des méthodes appliquées pour développer des ressources linguistiques en portugais pour le traitement automatique du langage. Les ressources sont divisées en dictionnaires et grammaires. D'une part, des dictionnaires généraux de mots simples et de mots composés ont été créés, ainsi que des dictionnaires spécialisés de noms propres (organisations, toponymes, et acronymes). D'autre part, en utilisant de l'information linguistique contenue dans les dictionnaires, des grammaires de désambiguïsation (Laporte et Monceaux, 1999) et des graphes syntaxiques ont été développés. Tandis que les grammaires de désambiguïsation visent à lever l'ambiguïté entre des noms et des adjectifs qui sont homographes, les grammaires locales sont utilisées pour décrire des dates, des expressions temporelles, des pourcentages et des structures prédicatives plus complexes.

Afin d'extraire des données biographiques contenues dans des dépêches de presse et de les organiser dans une base de connaissances, Kevers (2006) propose premièrement de créer une définition fonctionnelle des évènements biographiques (par exemple : naissance, décès, profession, etc.) sous forme de triplets ($\{sujet, relation, objet\}$) permettant d'exprimer les relations entre entités (par exemple : X est né à L, ou X est un patronyme et L un lieu). Une fois la liste des relations constituée, l'information est utilisée pour développer des cascades de grammaires (Friburger et Maurel, 2004) capables de décrire et aussi d'annoter les données biographiques.

3.7 Chaîne de traitement pour l'extraction de l'information à l'aide des grammaires locales

Étant donné un texte en entrée, nous énumérons une suite d'étapes (c.f. figure 3.6) afin d'appliquer une grammaire locale (possiblement en cascade) en vu d'effectuer une tâche d'extraction de l'information. Ces étapes, dont certaines facultatives, sont associées à plusieurs sous-problèmes traditionnels dans le traitement du langage naturel. Pour chaque problème, nous exposons également les approchés les plus répandues qui permettent leur résolution.

1. Cette approche est à l'opposé de la constitution des corpus statiques provenant, par exemple, des banques de données

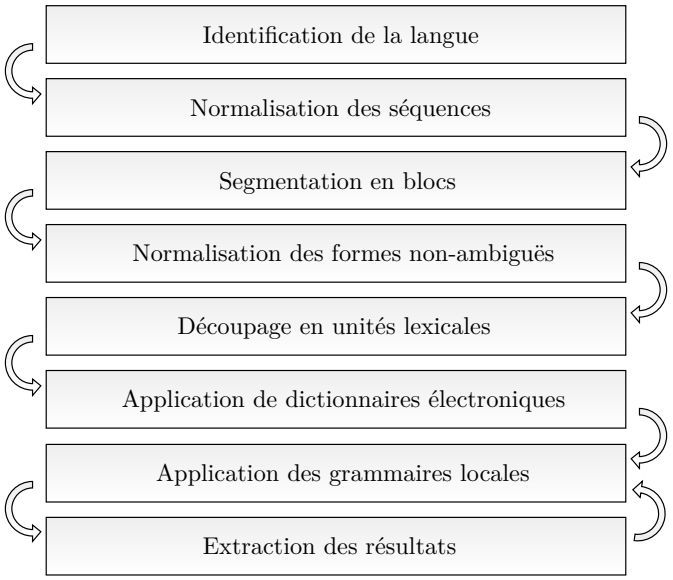


FIGURE 3.6 – Chaîne de traitement pour l'extraction d'information à l'aide des grammaires locales

a) **Identification de la langue** : correspond à la tâche de déterminer la langue principale du texte : anglais, français, espagnol, grecque, etc. Plusieurs techniques peuvent être mise en œuvre : classification fondée sur les n-grammes de caractères (Cavnar et al., 1994), prédiction par reconnaissance partielle (PPM, *Prediction by Partial Matching*, Teahan et Harper, 2003), apprentissage profond par réseaux de neurones (*Deep Neural Networks*, Lopez-Moreno et al., 2014), etc.

b) **Normalisation des séquences** : elle mobilise des règles d'insertion, de remplacement ou d'identification qui ont pour fonction, d'une part, d'insérer ou de remplacer des séquences de caractères par d'autres séquences et, d'autre part, de délimiter les séquences du texte qui ne seront pas prises en compte lors des analyses. Prenons comme exemple, les séparateurs (tabulations, sauts de ligne, etc.) ou les balises des langages de définition ou de description de documents (TEI, HTML, XML, etc.). Cette étape de normalisation comprend l'utilisation d'approches pour remplacer efficacement des motifs ou pour dépouiller la mise en forme d'un texte : expressions rationnelles Karttunen et al. (1996), translitération par des transducteurs finis Irvine et al. (2010), détection de la mise en forme en utilisant des caractéristiques textuelles peu profondes Kohlschütter et al. (2010), etc.

c) **Segmentation en blocs** : elle consiste à découper le texte en blocs tels que des phrases en utilisant, par exemple, des modèles fondés sur des transducteurs finis (Friburger et al., 2000), ou sur l'entropie maximale (*MaxEnt Classifier*, Agarwal et al., 2005), ou encore sur l'apprentissage non-supervisé (Kiss et Strunk, 2006), etc.

d) **Normalisation des formes non-ambiguës** : elle vise la normalisation des formes lexicales non-ambiguës. Cela consiste à remplacer les séquences du texte qui peuvent

être normalisées sans ambiguïté, par exemple, en anglais, « *she's* » par « *she is* » ou « *won't* » par « *will not* ». Cette étape peut mobiliser toute technique permettant de remplacer efficacement des séquences d'un texte, telles que celles fondées sur l'utilisation des transducteurs finis (Hassan et al., 2008) ou les techniques citées dans l'étape b).

e) **Découpage en unités lexicales** : elle consiste à séparer et regrouper dans un index les séquences de lettres contiguës qui ont une signification conjointe. Cette opération est effectuée en utilisant les séparateurs d'unités lexicales adaptés à la langue du texte, par exemple, des espaces pour les langues indo-européennes ou d'autres critères pour traiter des langues agglutinantes. Cette étape peut faire appel à toute technique pour découper un texte en unités lexicales : découpage par des transducteurs finis (Paumier, 2003b), par construction d'arbres binaires complets (PAT-*tries*) au niveau des caractères Carpenter (2005), ou par des modèles statistiques Darwish et al. (2014), etc.

f) **Application de dictionnaires électroniques** : elle consiste à rechercher l'association entre les unités lexicales du texte et les entrées lexicales des dictionnaires électroniques. Les dictionnaires contiennent des mots simples ou composés, associant à chaque entrée un lemme et des codes grammaticaux, sémantiques, flexionnels, ou n'importe quel autre code, relatif ou non à l'entrée. Cette étape peut mobiliser des approches permettant de stocker, charger, rechercher des entrées et appliquer efficacement un dictionnaire électronique à un texte : par exemple, via la construction d'un ou plusieurs automates acycliques minimaux (ADFA, *acyclic deterministic finite automaton*, Revuz, 1992, Watson, 2003), la construction d'index inversés Heinz et Zobel (2003), la construction de tables de hachage Lam et Huo (2005) ou encore la construction des RTN d'affixes (Berlocher et al., 2006) pour traiter les langues agglutinantes.

g) **Application des grammaires locales** : correspond à l'application au texte d'une ou de plusieurs grammaires locales, possiblement en cascade. Le cœur de cette tâche est prise en charge par un analyseur syntaxique. Ce dernier permet non seulement de vérifier que le texte en entrée correspond aux règles définies par la grammaire locale, mais aussi de stocker de manière indexée les motifs repérés. Les algorithmes d'analyse syntaxique les plus utilisés dans cette étape sont ceux fondés sur la première recherche en profondeur (*top-down depth-first*) comme celui d'UNITEX (Paumier, 2003a) ou bien du type Earley utilisé, par exemple, dans OUTILEX (Blanc et al., 2006).

h) **Extraction des résultats** : une fois les motifs repérés et indexés, il ne reste qu'à produire un résultat en sortie. En suivant la finalité pour laquelle la grammaire a été développée, la sortie peut, par exemple, prendre la forme de concordances, d'arbres d'analyse syntaxique, du texte en entrée muni des balises correspondantes aux motifs repérés, ou même de fichiers répertoriant les positions et, alternativement, les noms balises associés aux motifs.

Chacune des étapes listées précédemment doit être considérée d'un point de vue

général toute en prenant en compte la nature des textes à analyser. De ce fait, certaines sont facultatives, d'autres peuvent disparaître ou être fusionnées, ainsi que de nouvelles étapes, non énumérées, peuvent apparaître.

Par exemple, par rapport l'étape a) d'*identification de la langue*, elle est normalement facultative, les grammaires locales étant développées pour décrire un sous-langage appartenant à une langue ciblée, il est inhabituel de passer par une telle étape. Ceci étant dit, dans une tâche d'extraction d'information orientée à traiter plusieurs langues (une à la fois), des grammaires locales peuvent être conçues pour chaque une d'entre elles et alors reconnaître à l'avance la langue visée s'avère indispensable.

Pour ce qui est une éventuelle fusion d'étapes, nous pouvons citer le cas du traitement de l'arabe ou des langues agglutinantes. En effet, les étapes e) du *découpage du texte en unités lexicales* et g) *d'application de dictionnaires électroniques* sont parfois fusionnées car interdépendantes : les limites de chaque unité lexicale peuvent dépendre des limites des unités voisines, mais aussi des informations trouvées dans le dictionnaire.

Finalement, concernant des étapes qui peuvent apparaître, nous pouvons citer par exemple la *désambiguïsation des catégories grammaticales*[1] : elle consiste à sélectionner une seule catégorie grammaticale parmi celles associées à une unité lexicale par l'application de dictionnaires (après l'étape f). La désambiguïsation des catégories grammaticales peut utiliser toute technique permettant d'effectuer l'étiquetage morpho-syntaxique d'un texte : utilisation des grammaires par contraintes (Karlsson, 1990), utilisation de modèles de Markov cachés (HMMs, *Hidden Markov Models*, Church, 1988), utilisation des champs aléatoires conditionnels (CRFs, *Conditional Random Fields*, Lafferty et al., 2001), utilisation des machines à vecteurs de support (SVMs, *Support Vector Machines*, Nietzio, 2002), utilisation des approches fondées sur l'apprentissage profond par réseaux de neurones (Collobert et Weston, 2008, Andor et al., 2016) ou bien des approches hybrides combinant des approches symboliques et fondées sur l'apprentissage automatique (Sigogne, 2010).

3.8 Autres travaux autour des grammaires locales

Dans Calberg (2003) un modèle de génération morphologique à deux niveaux fondé sur des grammaires locales est utilisé pour le traitement du finnois. Au premier niveau, les morphèmes sont décomposés par des graphes indépendants permettant de découper les entrées du lexique en unités de base et de leur associer des informations sémantiques. Au deuxième niveau, des graphes dits de *linéarisation* sont appliqués sur les résultats précédents afin de produire des unités lexicales valides.

1. Observons que pour l'application des grammaires locales, que la désambiguïsation des catégories grammaticales est normalement absente, autrement dit, les grammaires sont souvent appliquées à des textes dont presque aucune ambiguïté lexicale n'a été levée. En effet, une des particularités de l'approche fondée sur l'utilisation des grammaires locales est de permettre un traitement du texte même en présence d'ambiguïtés lexicales.

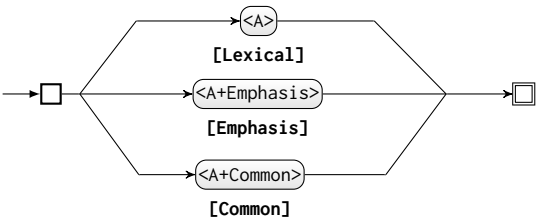
Dans Monnier et al. (2003), les grammaires locales sont utilisées pour introduire dans les textes des marqueurs destinés à l'étude de la prosodie de la langue anglaise. Le travail consiste d'abord à classifier la prosodie en 4 niveaux : l'accent tonique (ou *stress* en anglais), l'intonation, les frontières prosodiques (Bolinger, 1986, citée par Monnier et al.) et l'intensification énonciative. Ensuite, des dictionnaires électroniques et des grammaires locales sont créés pour identifier et marquer les structures prosodiques selon les niveaux précités. Finalement, à l'aide d'un script de post-traitement, une mise en correspondance entre les marqueurs trouvés et des paramètres acoustiques est effectuée pour générer un texte susceptible d'être interprété par un logiciel de synthèse vocale nommé KALI[1].

Concernant les dictionnaires, il s'agit des dictionnaires électroniques du LADL pour l'anglais (Klarsfled et Carthy, 1991, Monceaux, 1995, Chrobot et al., 1999) qui ont été enrichis par l'ajout de noms propres, de néologismes et de toute autre nouvelle entrée porteuse d'informations prosodiques. En outre, les anciennes entrées comme les nouvelles ont été munies de traits supplémentaires. Nous présentons ci-après des exemples d'entrées enrichies :

(10) **carnival**,.N+1:s

(11) **extremely**,.ADV+Emphasis

Dans l'exemple 10, le trait +1 est créé pour indiquer que dans *carnival* ['kɑrnəvəl] la syllabe qui porte l'accent tonique est la première. Dans l'exemple 11, le trait +Emphasis est utilisé comme un *marqueur d'intensité*. D'autres traits sont également créés pour indiquer des caractéristiques prosodiques accidentels.



Graphe 3.11 – Grammaire (simplifiée) pour ajouter des marqueurs prosodiques aux adjectifs (Monnier et al., 2003, p. 1139)

En ce qui concerne les grammaires locales, il est tout d'abord question de construire un graphe pour chaque catégorie syntaxique, par exemple comme le graphe 3.11 pour les adjectifs. En faisant référence aux entrées des dictionnaires, ces graphes sont utilisés pour étiqueter, au niveau des mots, toute occurrence d'une catégorie syntaxique. Ensuite, au niveau des phrases, des structures plus complexes comme les propositions incises sont décrites à l'aide de graphes d'un niveau supérieur, c'est-à-dire des graphes

1. KALI (Morel et Lacheret-Dujour, 2001) est un logiciel payant de synthèse vocale développé par le Laboratoire Crisco de l'Université de Caen.

composées de plusieurs sous-graphes, qui serviront à ajouter davantage de marqueurs prosodiques dans le texte.

Dans le cadre d'une étude linguistique des emplois des verbes *donner* et *passer* dans un corpus portant sur le football, Gasiglia (2004) fait appel à Cordial et Unitex. Les deux outils ont été utilisés conjointement pour extraire les occurrences des verbes et ainsi permettre d'analyser leurs structure syntaxiques et la nature de leurs arguments.

Dans Blanc et Dister (2004) est présenté une description des automates lexicaux avec structures de traits, c'est-à-dire des automates sur les mots dans lesquels les transitions peuvent être étiquetées par des masques lexicaux. Dans cette étude, une nouvelle représentation des masques lexicaux est utilisée afin d'adapter les algorithmes généraux sur les automates finis aux automates lexicaux. La démarche entreprise permet alors d'effectuer efficacement des opérations classiques sur les automates telles que la déterminisation, la minimisation, l'intersection ou le complément sur les automates lexicaux. À partir d'un état qui comporte une transition sortante étiquetée par un masque lexical, la technique qui permet l'ajustement des algorithmes réside à *découper les transitions de telle sorte que les ensembles décrits soient deux à deux disjoints ou égaux*. En suivant cette stratégie, le nombre d'opérations de comparaison et de découpage à réaliser peut être, dans le pire des cas, exponentiel par rapport au nombre total de transitions. Ceci est à prendre en compte lors de la construction des automates, mais pas lors de leur application.

3.9 Défis liés à la gestion et construction des grammaires locales

3.9.1 Gestion des grammaires

Afin de gérer le volume important des grammaires locales nécessaires pour les applications du TAL et de faciliter leur diffusion, accessibilité et partage, Constant (2004) proposait la création d'un outil de gestion de graphes sous la forme d'une bibliothèque en-ligne. Cette plateforme visait d'une part, l'hébergement des catalogues des grammaires documentées, d'autre part, la possibilité de rechercher ces grammaires en utilisant des filtres sur les composants ou sur les métadonnées indexées.

Dans Constant (2007), un tel système de partage de grammaires locales nommé Graal est présenté. Il est constitué par un ensemble de dépôts décentralisés, composés de paquets comprenant des métadonnées et une collection de grammaires. Une interface visuelle en-ligne appelée GraalWeb, permet d'avoir un accès centralisé aux informations ainsi qu'à des fonctions d'exploration et de recherche des grammaires.

Bien que l'initiative de *Graal* avait une visée essentiellement pratique, dans les faits, peu d'utilisateurs ont partagé leurs grammaires au moyen du système, moins encore de dépôts de grammaires ont vu le jour. De surcroît, la décision des principaux navigateurs web (Firefox version 52, Chrome version 35, Opera version 37, entre autres) de limiter la technologie pour exécuter des applets Java, rendra inaccessible l'accès à

GraalWeb [1], tout au moins dans son implémentation actuelle.

3.9.2 Génération automatique de grammaires locales

Traboulsi (2005) décrit le prototype d'un système appelé *Local Grammar Finder* (LGFinder) pour la construction semi-automatique de grammaires locales visant l'identification d'entités nommées dans un corpus de textes d'actualité financière en anglais. À partir d'un corpus de texte non étiqueté, la méthode est divisée en en trois étapes (voir figure 3.7) : 1. Identifier la fréquence de mots lexicaux (noms, verbes, adjectifs, adverbes) ; 2. Analyser les collocations dans le texte afin de trouver les plus fréquentes et 3. Générer des grammaires locales à partir du traitement des concordances de chacune des collocations trouvées.

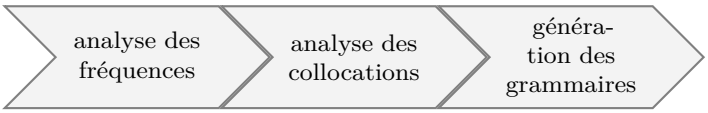


Figure 3.7 – Méthode LGFinder pour la génération de grammaires locales

Après l'identification des collocations les plus fréquentes, le cœur de la méthode repose tout d'abord dans la génération d'une concordance à partir de chaque collocation. Ensuite, les mots de chaque concordance sont collectés et représentés sous la forme de vecteurs géométriques ($\mathbb{R}^n$, avec $n = 14$). Les 7 premières coordonnées des vecteurs sont respectivement codés selon un ensemble prédéfini de caractéristiques lexicales : mots des catégories ouvertes (*open-class words* – noms, verbes lexicaux, adjectifs et adverbes), mots de la classe fermée (*close-class words* – pronoms, déterminants, conjonctions et prépositions), mots hors dictionnaire, nombres et signes de ponctuation. Les 7 autres servent à encoder les positions (normalisées) de chacune des caractéristiques lexicales mentionnées ci-dessus. Une fois constitués, les vecteurs sont ordonnés selon leur fréquence et comparés en utilisant la distance euclidienne afin de les rapprocher, c'est-à-dire de regrouper les motifs linguistiques qui se ressemblent le plus. En dernier lieu, les groupes des motifs linguistiques les plus fréquentes servent à construire les grammaires locales à l'aide d'Unitex.

En ce qui concerne la démarche implémentée par LGFinder, nous pouvons remarquer deux limitations : En premier lieu, l'algorithme présenté pour regrouper les vecteurs (Traboulsi, p. 16), recherche les vecteurs les plus proches de chaque vecteur et a une complexité en temps quadratique $\mathcal{O}(n^2)$. En effet, étant donné n vecteurs, on a $n \cdot \frac{(n-1)}{2}$ pairs de vecteurs et autant de distances euclidiennes à calculer. Enfin, lors de la dernière étape, une intervention manuelle est requise pour définir les catégories sémantiques correctes (patronymes, organisations, toponymes, etc) des motifs linguistiques codés par les vecteurs. En somme, ces deux aspects font obstacle

1. *GraalWeb* est disponible sur http://igm.univ-mlv.fr/~mconstan/library/index_graalweb.html

à l'analyse de grands volumes de textes et à la mise en pratique d'un algorithme de génération complètement automatique de grammaires locales.

3.9.3 Limitations

Toutes les approches qui utilisent les grammaires locales fournissent un taux de précision satisfaisant. Cependant, il reste difficile de construire des grammaires locales capables de capter toutes les combinaisons possibles et de faire face à la complexité du traitement des variations langagières attendues et inattendues : mots inconnus, omis ou mal orthographiés, violations des règles de grammaire standard.

Plusieurs auteurs ont évoqué les limites d'utilisation, en termes d'ergonomie, performance et qualité, des grammaires locales.

Selon Friburger (2002), la principale difficulté découle de l'incomplétude des grammaires locales, des ambiguïtés et de l'absence éventuelle de contextes dans les phrases. L'incomplétude des grammaires est relevée aussi par Nakamura (2004) où il est question d'insertions imprévisibles (de motifs) qui ne sont pas prises en compte dans les graphes.

Dister et al. (2004) fait un même constat relatif aux motifs non reconnus « *tout simplement parce qu'ils sont mal orthographiés...* », ou comme souligné par Fairon et Watrin (2003), dans le cas des patronymes, simplement parce que les motifs à reconnaître présentent patronymes beaucoup de variations.

Le problème de contextes est également soulevé dans les travaux de Sætre (2003) relatifs au langage microbiologique, où il est complexe de décrire de motifs contenant de longues séquences. Pour sa part, Kevers (2006) se questionne sur la possibilité d'utiliser des contextes porteurs d'informations mais qui sont éparpillées dans toute une phrase ou paragraphe.

Constant (2003) souligne les difficultés rencontrées pour construire et maintenir des grammaires précises et complètes. La maintenance des graphes est aussi évoquée dans Pirovani et Silva de Oliveira (2015), qui se questionne sur la façon de créer, adapter et transformer des règles dans des graphes afin d'augmenter la performance de la reconnaissance. Par ailleurs, Ezzat (2014) insiste sur le coût de développement des ressources linguistiques nécessaires pour obtenir une couverture satisfaisante.

En outre, pour Gasiglia (2004) le problème réside en « *l'incapacité de discriminer les occurrences verbales des occurrences nominales* », autrement dit, de traiter les ambiguïtés des catégories morphosyntaxiques des entrées lexicales. Selon Fairon (1998), Calberg (2003), Poibeau et al. (2003) il est aussi question des tâches de prétraitement, par exemple, il est indispensable de convertir du HTML en TXT ou prétraiter les textes pour avoir un résultat exploitable à travers les grammaires locales.

Dans le même sens, certaines tâches exigent des modules de post-traitement des résultats, dans Monnier et al. (2003), pour remplacer des marqueurs insérés par les grammaires avec des paramètres acoustiques définis dans un tableau. Dans Kogkitsidou et Antoniadis (2016) pour réaliser des transformations au niveau des caractères sur des

marqueurs insérés dans des messages SMS.

Le recours à des modules supplémentaires fait aussi partie des approches *hybrides* ou la sortie de la grammaire représente l'entrée d'un module fondé sur l'apprentissage automatique : Watrin et al. (2014), Boujelben et al. (2014), Hkiri et al. (2016)

Comme conséquence directe ou indirecte de ces limites, les tâches d'extraction de l'information à l'aide des grammaires locales, telle que la reconnaissance d'entités nommées, bute sur un plafond de performance. Il est devenu coûteux d'améliorer le taux de rappel, autrement dit, le pourcentage de motifs retrouvés au regard des motifs pertinents, des systèmes existants en ajoutant des règles supplémentaires aux grammaires locales ou en enrichissant les ressources linguistiques qu'elles mobilisent.

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 des variations attendues et inattendues dans un texte. Ceci résulte du fait de l'apparition des mots inconnus, ou des mots composés, pas nécessairement formées par de mots inconnus (Maurel, 2004), ou mal orthographiés ou encore dérogeant aux règles syntaxiques établies.