UNIVERSITE DE FRANCHE-COMTE

ECOLE DOCTORALE «LANGAGES, ESPACES, TEMPS, SOCIETES»

Thèse en vue de l'obtention du titre de docteur en
INFORMATIQUE

ANALYSE MORPHO-SYNTAXIQUE AUTOMATIQUE ET RECONNAISSANCE DES ENTITES NOMMEES EN ARABE STANDARD

Présentée et soutenue publiquement par

Slim MESFAR

Le 24 novembre 2008

Sous la direction de M. le Professeur Max SILBERTZTEIN

Membres du jury :

Abdelmajid BEN HAMADOU, Professeur à l'université de Sfax, Rapporteur
Jean-Patrick GUILLAUME, Professeur à l'université Paris III, Nouvelle-Sorbonne
Denis MAUREL, Professeur à l'université de Tours, Rapporteur
Odile PITON, Maître de Conférences à l'université Paris I, Panthéon–La Sorbonne
Paul SABATIER, Directeur de recherche au CNRS, Marseille
Max SILBERZTEIN, Professeur à l'université de Franche-Comté
Jean-Marie VIPREY, Professeur à l'université de Franche-Comté

Remerciements

Table des matières

Liste des Figures

Liste des tableaux

Introduction générale

Contexte général

Cette thèse s'inscrit dans le cadre général du Traitement Automatique des Langues naturelles (TALN). Cette discipline se situe à la frontière de la linguistique, de l'informatique et de l'intelligence artificielle ; elle concerne la conception et le développement de programmes et techniques informatiques capables de traiter de façon automatique des données exprimées dans une langue. Les applications liées au TALN ont fait l'objet d'une attention toute particulière depuis plusieurs décennies. Bien que certaines langues ont été réputées privilégiées comme le français et l'anglais, d'autres voient se poursuivre les recherches et les travaux afin de proposer des outils robustes de traitement.

Nos recherches portent sur l'étude de la langue arabe qui, malgré sa position de cinquième langue au monde[1] avec plus de 50 000 sites arabes sur le web et plus de 320 millions locuteurs, n'a encore aucun analyseur capable de traiter de façon robuste la totalité de ses phénomènes morpho-syntaxiques. Par ailleurs, nous assistons à un accroissement des contenus textuels en arabe, surtout en ligne. A ce jour, le traitement et l'exploitation de ces ressources documentaires présentent encore un défi pour les chercheurs dans le domaine du traitement automatique des langues naturelles.

C'est dans ce contexte général que le problème d'une analyse morpho-syntaxique automatique robuste de l'arabe a été abordé. Nous avons entrepris la construction d'un ensemble de ressources linguistiques pour l'arabe afin d'implémenter une composante d'analyse morpho-syntaxique automatique et reconnaître les entités nommées dans les textes écrits en arabe standard. Cette composante, qui comporte la description de son vocabulaire et de sa morpho-syntaxe, permet de mieux comprendre la langue.

Plan de la thèse

Le premier chapitre de cette thèse définit les outils de base utilisés pour construire les descriptions formalisées à large couverture des langues naturelles (dictionnaires et grammaires électroniques). Les automates à états finis, déterministes, acycliques et minimaux se sont très largement répandus en TALN pour la représentation et le stockage de gros volumes de données sous forme de dictionnaires électroniques. Dans ce travail, nous nous intéressons à l'étude de la structure interne de tels automates. Plus précisément, nous adoptons une approche de construction séquentielle et nous proposons un nouvel algorithme de compression dynamique de ces automates pouvant, ainsi, répondre aux besoins des langues flexionnelles telle que l'arabe.

Dans le second chapitre, nous proposons une brève étude linguistique de la langue arabe. Nous nous intéressons principalement aux phénomènes morpho-syntaxiques qui lui sont spécifiques et plus particulièrement aux problèmes rencontrés dans l'analyse de l'arabe tel qu'il est écrit couramment, tels que les problèmes de voyellation, de flexion, de dérivation, d'agglutination et d'ambiguïté.

Dans le troisième chapitre, nous discutons les différentes approches pour la représentation des données et la conception de l'analyse morpho-syntaxique. Nous présentons un panorama des travaux déjà réalisés en pointant leurs insuffisances ; ils servent de repères de validation pour l'analyseur que nous décrirons plus loin. Nous mentionnons que pour notre part, nous nous orientons vers une approche modulaire qui nous semble la plus adaptée aux objectifs fixés. Enfin, nous présentons l'architecture générale et les différents points qui permettent de valider notre choix

[1] Source : Ecole d'été en Linguistique par le biais de l'encyclopédie Encarta en ligne (consultée le 3 juin 2007).

de la plateforme de développement linguistique NooJ comme outil de développement de notre analyseur.

Au quatrième chapitre, nous commençons par une description de la macrostructure et la microstructure des dictionnaires électroniques que nous construirons. Cette partie est consacrée à la présentation de l'organisation et du contenu du dictionnaire de la langue que nous appellons « **El-DicAr** » (**El**ectronic **Dic**tionary for **Ar**abic). Ensuite, nous décrivons l'interaction entre les entrées lexicales et les descriptions formalisées des règles de flexion et d'agglutination afin de pouvoir générer automatiquement toutes les formes fléchies potentielles. L'ensemble de ces formes est stocké dans des machines à états finis à l'aide du compilateur de dictionnaires décrit dans le premier chapitre. La version compilée du dictionnaire est utilisée comme moteur linguistique pour les étapes ultérieures d'analyse morpho-syntaxique.

Avec le cinquième chapitre, nous entamons la série des applications informatiques que nous développons. Nous commençons par une composante d'analyse morphologique pour arriver à annoter automatiquement les textes et évaluer la couverture des ressources linguistiques mises en œuvre. En effet, l'arabe est une langue fortement agglutinante ; son analyse ne nécessite pas seulement la vérification de l'appartenance de chaque mot du texte au lexique « **El-DicAr** » et à la liste de formes fléchies qui en découle, mais aussi de donner tous les découpages potentiels en morphèmes. Chacun de ces morphèmes est associé aux informations morpho-syntaxiques qui s'y rattachent en se basant sur un ensemble de contraintes lexicales pour en vérifier la validité. Par ailleurs, nous proposons un nouvel algorithme d'analyse pour le moteur lexical de NooJ. Cet algorithme suggère une nouvelle routine de parcours des transducteurs à états finis représentant le lexique de la langue afin de pouvoir traiter aussi bien les mots voyellés et les non voyellés que ceux qui le sont partiellement. Nous décrivons la mise en place d'un ensemble d'heuristiques pour la correction des erreurs typographiques les plus fréquentes de la langue. Enfin, nous procédons à une évaluation des ressources linguistiques et de l'analyseur morpho-lexical. Cette évaluation est réalisée sur un corpus de textes bruts qui inclut environ 130 000 formes différentes.

Dans le sixième chapitre nous nous orientons vers l'analyse syntaxique locale de l'arabe pour traiter la question de reconnaissance automatique des entités nommées. Nous décrivons un système de reconnaissance des noms propres, expressions temporelles et numériques. Ce système se base sur une combinaison des résultats obtenus par le biais de l'analyseur morphologique, ci-dessus décrit, et de grammaires locales représentant des règles d'identification écrites manuellement.

Le dernier volet de cette thèse est consacré à la conception et la réalisation d'un service de concordances en ligne, baptisé **NooJ4Web** (NooJ for the Web - NooJ pour le web). Nous présentons l'architecture globale de ce concordancier, les étapes de traitement, la construction des corpus de recherche et la formulation des requêtes. Cette application permet la validation des différentes ressources développées à travers une mise en ligne gratuite de l'application au profit des utilisateurs potentiels et le recueil de leurs réactions et suggestions. Finalement, nous décrivons les différentes utilisations de cette application illustrées par le déroulement d'un exemple (captures d'écran à l'appui).

Chapitre 1

Technologie à états finis pour le traitement des langues naturelles

La technologie à états finis, désignant l'utilisation de machines à états finis comme les automates et les transducteurs, est en étroite connexion avec une grande variété de domaines tels que l'algorithmique, la logique, les compilateurs informatiques, le traitement de textes, l'identification des chaines d'ADN, les systèmes de contrôle de protocoles, le traitement de l'information, le traitement des langues naturelles, etc. Ce dernier domaine représente notre terrain d'application.
L'utilisation de la technologie à états finis a été le sujet d'étude dans plusieurs travaux de recherche tels que [Revuz, 1991], [Daciuk, 1998] et [Tounsi, 2007]. Elle a été privilégié par différentes équipes de recherche, tels que les Centres de recherche de Xerox et Bells Labs Research Center, pour créer des applications capables d'aborder les différents niveaux d'une analyse linguistique (niveau lexical, morphologique, syntaxique, etc.).

1.1Principales motivations

Depuis le début des travaux qui ont démontré la possibilité d'appliquer cette technologie à la représentation linguistique [Koskenniemi, 1983], les machines à états finis ont été jugées comme les plus adéquates pour les descriptions morphologiques et phonologiques de toutes les langues, indépendamment de leurs natures (romanes, sémitiques, germaniques, etc.) [Beesley et Karttunen, 2003]. Il existe un certain nombre d'avantages qui rendent la technologie à états finis particulièrement attractive dans le domaine du traitement automatique des langues naturelles, TALN. Parmi ces avantages, nous citons :

- **Représentativité** : A partir des premiers travaux de Johnson [Johnson, 1972], les recherches se sont poursuivies pour démontrer que les règles morphologiques et phonologiques, couramment utilisées dans les théories de la linguistique, peuvent être directement mises en œuvre sous forme de machines à états finis [Karttunen, 1995] ;
- **Modularité** : Divers moyens de combinaison des machines à états finis sont proposés pour supporter les diverses opérations sur les langages représentées par ces machines. Par exemple, les opérations d'union favorisent le développement modulaire. En l'occurrence, si nous développons différents fragments de la grammaire pour que chacun traite un sujet bien particulier, ceux-ci peuvent ensuite être directement combinés en une seule machine. Ainsi, une collaboration interdisciplinaire est possible : un linguiste peut être chargé de caractériser les phénomènes linguistiques (description des phénomènes sous formes de règles ou expressions rationnelles) et un programmeur peut se concentrer sur la mise en œuvre effective. L'avantage en est que les expressions rationnelles utilisées par les machines à états finis sont assez proches des notations standard du domaine de la linguistique [Yona et Wintner, 2005], ces règles sont assez lisibles et compréhensibles ;
- **Représentation compacte** : Les automates à état finis peuvent être minimisés, en garantissant que, pour un langage donné, un automate avec un nombre minimal d'états peut toujours être généré (cf. section 1.3). L'algorithmique décrite dans ce premier chapitre de la thèse montre la diversité des techniques développées afin d'apporter des améliorations en terme de performance de stockage ;
- **Rapidité et efficacité des traitements** : Lorsqu'un automate est déterministe, sa procédure de reconnaissance pourrait être optimale (linéaire en fonction de la longueur de la chaîne à reconnaître). Les automates peuvent représenter de très gros automates décrivant des lexiques ainsi que leurs inflexions. La compilation de larges automates à états finis qui comprennent plusieurs millions d'entrées ne demanderait que quelques secondes. Ces machines peuvent facilement être réunies pour former de plus larges automates. Les automates pouvent toujours être déterministes (voir section 1.3.1), les outils développés tirent profit pour apporter des améliorations en terme de performances de temps d'accès ;

« *Les automates acycliques sont une structure de données bien adaptée à la représentation de lexiques de langues naturelles, … La représentation par automates acycliques est économique en espace, elle permet de réaliser une compression importante des données. Elle fournit également des fonctions d'accès et de recherche de motifs, des plus rapides* », [Revuz, 1991]

- **Réversibilité** : Les automates et les transducteurs à états finis sont, par nature, déclaratifs et entièrement réversibles : ils peuvent servir autant pour l'analyse que la génération. En particulier, les transducteurs peuvent être utilisés pour représenter différents types de chaînes sans avoir à modifier la structure interne de la machine à états finis ;
- **Maniement des langues concaténatives aussi bien que celles non concaténatives** [Beesley, 1998] : La technologie à états finis permet d'effectuer des traitements linguistiques comme la dérivation, l'agglutination ou la morphologie à base de racines et de schèmes. En outre, généralement les applications utilisant des automates à états finis sont développées et tirent profits des architectures orientées-objets. Par exemple, le module qui fléchit les noms et les adjectifs hérite du même module utilisé pour toutes les formes nominales. Ce module hérite, à son tour, d'un module général de règles de flexion ;
- **Support de l'Unicode** : Cette capacité permet aux développeurs de travailler dans leurs langues maternelles dont les scripts utilisent des alphabets non latins.

Suite à ce bref exposé des principales motivations pour l'emploi des machines à états finis dans le domaine du traitement automatique des langues naturelles, nous procédons à une description des principaux outils formels utilisés.

1.2 Outils formels de base : théorie

Dans la littérature, nous trouvons de multiples définitions des machines à états finis. Les différences entre celles-ci consistent essentiellement dans le choix des symboles décrivant les composants dont les transitions et les états. Sur le plan théorique, ces définitions sont équivalentes ; elles contribuent à la simplification de la description et l'explication des applications qui y sont liées. Dans ce mémoire, nous reprenons celles utilisées dans [Daciuk, 2000], [Watson, 2003] et [Tounsi, 2007].

1.2.1 Expressions rationnelles

Définition 1 (Alphabet, mot, langage) :

Soit Σ un ensemble fini non vide de symboles appelé alphabet. Les symboles de Σ sont dénommés lettres ou caractères. Un mot ω sur Σ est une suite finie x_1, .., x_n d'éléments de Σ. ω est aussi appelé séquence et sa longueur est notée $|\omega|$. On note Σ^* l'ensemble des mots définis sur Σ. Un langage est un sous-ensemble de Σ^*.

L'ensemble des langages réguliers sur Σ peut-être défini récursivement par les règles suivantes :

- le langage vide Ø est un langage régulier ;
- le langage {ε} est un langage régulier ;
- pour tout x de Σ, {x} est un langage régulier ;
- si L1 et L2 sont des langages réguliers, alors nous pouvons construire les langages réguliers :
  - ✓ $L1 \cup L2$: l'union ensembliste des deux langages ;
  - ✓ $L1 \cdot L2$: la concaténation des deux langages ;
  - ✓ $L1^*$: la fermeture de Kleene[2] ;

[2] La fermeture de Kleene du langage régulier L1, notée $L1^*$, désigne l'ensemble de toutes les chaînes de caractères qui peuvent être formées en concaténant zéro, une ou plusieurs chaînes de L1.

Notons que certains langages ne sont pas réguliers, comme par exemple le langage $\{a^n.b^n \mid n > 0\}$ défini sur le vocabulaire {a, b}. En particulier, la plupart des langages de programmation ne sont pas des langages réguliers.

Définition 2 (Expression rationnelle) :

Tout langage régulier sur *L* peut être décrit par une expression rationnelle, c'est-à-dire un terme construit sur l'alphabet $\Sigma \cup \{+, ., *, \underline{\varepsilon}, \underline{\emptyset}\}$ de la manière suivante :

- $\underline{\emptyset}$ décrit le langage vide $\emptyset$;
- $\underline{\varepsilon}$ décrit le langage $\{\varepsilon\}$;
- pour tout x appartenant à Σ, x décrit le langage {x} ;
- si e1 et e2 sont des expressions rationnelles sur Σ décrivant respectivement les langages L1 et L2 alors :
  - ✓ e1 + e2 décrit le langage $L1 \cup L2$
  - ✓ e1.e2 décrit le langage $L1 \cdot L2$
  - ✓ $e1^*$ décrit le langage $L1^*$

Par convention, nous considérons que l'opérateur unaire « * » est plus prioritaire que l'opérateur « . », lui-même plus prioritaire que l'opérateur « + ».

Nous donnons, ci-dessous, quelques exemples d'expressions rationnelles définies sur le vocabulaire {a, b, c}, ainsi que les langages (réguliers) correspondants :

Expression rationnelle	Eléments du langage
a + b.c	{a, bc}
$(ab)^*$	{ε, ab, abab, ababab, …}
$ab + c^*$	{ε, ab, c, cc, ccc, …}

Notons que, les langages réguliers sont les langages reconnus formellement par des automates finis.

1.2.2 Automates à états finis

Un automate est un modèle abstrait de machine qui peut être vu comme une représentation graphique des données à modéliser. Il possède un alphabet, un ensemble d'états et une relation de transition. Une transition, δ (p, a)= q, indique que l'état q est atteint à partir de l'état p en lisant le symbole a de l'alphabet.
On formalise les notions liées aux automates par les définitions suivantes :

Définition 3 (Automate fini non déterministe) :

Un automate fini non déterministe est un quintuplet A = (Q, Σ, δ , I, F), où :

- Q est un ensemble fini non vide d'états ;
- Σ est l'alphabet ;
- δ : Q x Σ → 2^Q est une fonction de transition. Quand la transition δ (q,a) est non définie, nous notons que δ (q,a) = ⊥ . Nous étendons la fonction δ à la fonction de transition de mots δ^* : 2^Q x Σ^* → 2^Q définie comme suit : (où a ∈ Σ, x ∈ Σ^*) :

δ (q, ε) = q

$$\delta^*(q, ax) = \begin{cases} \delta^*(\delta(q, a), x) & \text{si } \delta(q,a) \neq \perp \\ \perp & \text{sinon} \end{cases}$$

- I ⊆ Q est l'ensemble non vide des états initiaux ;
- F ⊆ Q est l'ensemble non vide des états finaux.

Nous définissons aussi :

- |A| : la taille de l'automate qui est égale au nombre d'états dans Q, soit |Q|.
- $L(A)$: le langage associé à l'automate A formé par l'ensemble des mots reconnus par cet automate : $L(A) = \{ \omega \in \Sigma^* \mid \delta^*(I, \omega) \in F , \delta^*(I, \omega) \cap F \neq \emptyset \}$
- $L(q)$: l'ensemble de toutes les chaînes formées partant de l'état q pour atteindre l'un des états terminaux de l'automate A : $L(q) = \{ \omega \in \Sigma^* \mid \delta^*(q, \omega) \in F \}$

  Notons que, $L(A) = \sum_{q_i \in I} L(q_i)$:

  L'ensemble $L(q)$ peut aussi être défini récursivement comme étant :

  $$L(q) = \{a \in L(\delta(q, a)) \mid a \in \Sigma \wedge \delta(q, a) \neq \perp \} \cup \begin{cases} \{\varepsilon\} \text{ si } q \in F \\ \phi \text{ sinon} \end{cases}$$

Définition 4 (Automate fini déterministe) :

Contrairement au cas de non déterminisme, un automate fini déterministe ne comporte qu'un seul état initial q_0, c'est-à-dire $I = \{q_0\}$ et $|I|=1$. De plus, pour chaque couple (q, a), où $q \in Q$ et $a \in \Sigma$, la fonction $\delta(q, a)$ détermine une valeur unique ; sa fonction de transition est définie comme étant

$\delta : Q \times \Sigma \rightarrow Q$.

Le déterminisme d'un automate à états finis nous garantit une séquentialité de l'analyse d'un mot.

Définition 5 (Chemin et acceptation dans un automate déterministe A) :

Un mot est reconnu par un automate si et seulement si il est l'étiquette d'un chemin qui part de l'état initial et arrive à un état terminal.

La reconnaissance d'un mot au moyen d'un automate à états finis se fait pas à pas. A chaque pas, une transition est effectuée, elle correspond au couple formé par l'état en cours et un symbole, appartenant à l'alphabet Σ, lu à partir du mot à reconnaître. Le mot est dit reconnu s'il existe une séquence de transitions dans l'automate qui permet de lire tous les symboles du mot et d'aboutir à l'un des états finaux (signalés par un double cercle). Par contre, si le système est dans un état autre qu'un état terminal après avoir lu la phrase, ou si le système bloque – c'est-à-dire se trouve dans une configuration dans laquelle aucune transition n'est applicable – le mot est rejeté.

Soit un mot $\omega = x_1, .., x_{|w|}$, le chemin de ω dans M est une séquence de $|\omega| + 1$ états $(q_0,\ldots, q_{|w|})$ tel que : $q_{i+1} = \delta(q_i, x_{i+1}), \forall i \in [0, |\omega| - 1]$

Un mot $\omega = x_1, .., x_{|w|}$ est accepté par l'automate M si le chemin de ω dans M relie l'état initial q_0 à un état final $q_{|w|}$.

Théorème de Myhill, 1956 :

Etant donné un automate non déterministe A, il existe un automate déterministe A' qui reconnaît le même langage.

Ci-dessous, nous proposons deux automates à états finis équivalents : le premier est non-déterministe - le deuxième est déterministe, reconnaissant le langage : $a^n b^m$, avec $n, m \geq 1$. Ils permettent, alors, de reconnaître des séquences de caractères telles que : ab, aab, aabb, abbb, etc.

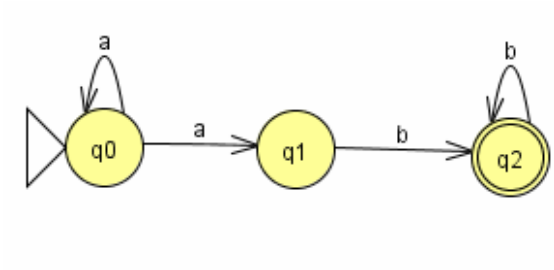


Figure 1: Automate à états finis non déterministe (pour le langage $L=a^n b^m$, avec n, m ≥ 1)

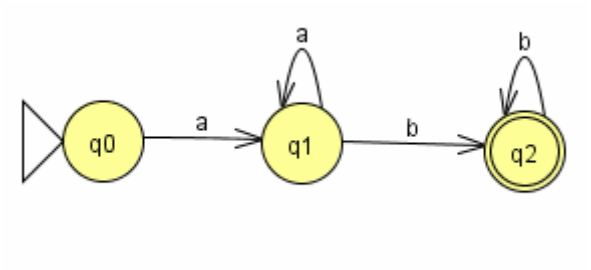


Figure 2 : Automate à états finis déterministe (pour le langage $L=a^n b^m$, avec n, m ≥ 1)

Le non déterminisme du premier automate provient du fait que deux transitions portant la même étiquette 'a' sont spécifiées à l'état q_0, ceci se note formellement : $\delta(q_0, a) = \{q_0, q_1\}$. Par contre, le deuxième automate est déterministe puisqu'au couple (q_0, a) correspond la seule valeur q_1.

Dans notre exemple, une simple modification de l'automate non déterministe (cf. Figure 1) permettait d'obtenir l'automate déterministe équivalent (cf. Figure 2), puisqu'il a suffit de déplacer la transition 'a' de l'état q_0 à l'état q_1. Cependant, il serait faux de croire que la déterminisation d'un automate est toujours aussi simple. En général, cette conversion se fait au prix d'une augmentation parfois considérable du nombre d'états de l'automate.

Définition 6 (Automate fini acyclique déterministe) :

Un automate fini acyclique déterministe ne comporte pas de cycles ou de suite de transitions qui permettent de passer deux fois par le même état. Ceci se note formellement :
$\delta(q, a) \neq q$ et $\delta^*(q, ax) \neq q$, pour tout $a \in \Sigma$, $x \in \Sigma^*$ et $q \in Q$.

Les automates acycliques sont plus spécialement adaptés au problème des lexiques. Ils ont en effet la particularité de reconnaître des ensembles finis, et réciproquement tout ensemble fini est reconnaissable par un automate acyclique.

1.2.3 Transducteurs à états finis

Les transducteurs sont des automates avec deux alphabets pour étiqueter les transitions. Le deuxième jeu de symboles est appelé l'alphabet d'émission. Tandis qu'un automate à états finis se contente de reconnaître les éléments d'un langage, un transducteur est un automate "étendu" permettant, pour chaque étape de transition, d'également produire un symbole en sortie. Il existe plusieurs formalismes pour décrire un transducteur à états finis, prenons celui de la machine de Mealy :

Définition 7 (Transducteur à états finis) :

Un transducteur à états finis est un quintuplet $T = (Q, \Sigma, \delta, q_0, F)$, où:

- Q est un ensemble fini non vide d'états ;
- Σ est un alphabet de symboles complexes. Chaque symbole est constitué d'une paire (e,s), où e $\in$ à un alphabet d'entrée E, et s $\in$ à un alphabet de sortie S. $\Sigma \subseteq E \times S$;
- $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble non vide des états finaux.

Les transducteurs ainsi définis sont dits sous-séquentiels ; une information est émise par leurs états terminaux. Un mot est reconnu par un transducteur de la même façon que par un automate, mais de plus une sortie est produite. Ce type de représentation est à priori idéal pour décrire des

dictionnaires électroniques comme le DELAP[3], où l'on veut émettre la phonémisation d'un mot au fur et à mesure que l'on vérifie sa présence dans le dictionnaire.

Sur la Figure 3, ci-dessous, le transducteur représenté permet de reconnaître les deux chaînes de caractères : “ab” et “ac”, il produit, respectivement, les deux sorties : “XY” et “ZX”.

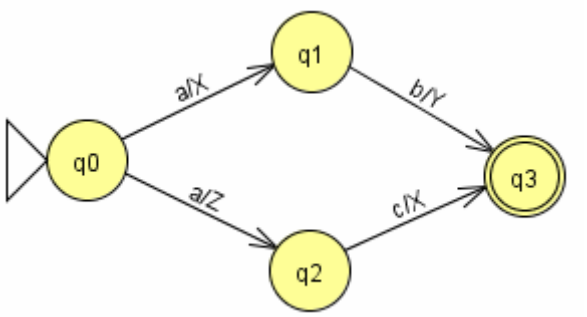


Figure 3 : Transducteur à états finis

Comme pour les automates à états finis, nos recherches se limitent à des transducteurs acycliques qui reconnaissent des ensembles finis. Cette acyclicité offre des possibilités algorithmiques qui n'existent pas sur les transducteurs standards.

Tandis qu’expérimentalement les automates et les transducteurs à états finis permettent de représenter adéquatement de nombreux phénomènes linguistiques (la flexion, les variantes lexicales, les ambiguïtés, etc.), la récursivité du langage démontrée par Chomsky dans ses travaux sur la grammaire générative reste non résolue par les deux modèles déjà définis. Des automates à pile et des réseaux de transitions récursifs (RTNs : Recursive Transition Networks) sont alors mis en œuvre pour résoudre de tels problèmes.

1.2.4 Automates à pile

Un automate à pile est semblable à un automate fini non-déterministe mais il dispose également d'une pile qui peut être utilisée pour stocker des informations pertinentes. La puissance de calcul des automates à piles correspond aux langages non-contextuels pouvant être décrits par une grammaire hors-contexte.

Définition 8 (Automate à pile) :

Un automate à pile est un 7-uplet $M = (Q, \Sigma, \Gamma, \delta, z_0, q_0, F)$ tel que :

- Q est un ensemble fini d’états ;
- Σ est l’alphabet d’entrée ;
- Γ est l’alphabet de pile ;
- $\delta \subseteq ((Q \times \Sigma^* \times \Gamma) \times (Q \times \Gamma^*))$ est la fonction de transition ;
- $z_0 \in \Gamma$ est le symbole initial de pile ;
- $q_0 \in Q$ est l’état initial ;
- $F \subseteq Q$ est l’ensemble non vide des états finaux.

Un automate à pile permet de reconnaître un ensemble de mots de Γ^*, le langage accepté ou reconnu par l’automate. A chaque transition $\delta((q, u, \beta), (q_0, \gamma))$, l’automate passe de l’état q à l’état q_0 en lisant le préfixe u sur la chaîne d’entrée et en remplaçant le sommet de pile β par γ.

[3] DELAP : Dictionnaire Electronique du LADL pour la Phonémique.

Les automates à pile utilisent une zone mémoire organisée en pile qui permet de sauver des informations :

- Le choix d'une transition peut dépendre de la valeur au sommet de la pile (pop) ;
- Une transition peut entraîner un ou plusieurs empilements (push).

Nous notons, aussi, les propriétés suivantes :

- Le langage reconnu par un automate à pile est un langage hors-contexte ;
- Pour tout langage hors-contexte, il existe un automate à pile qui le reconnaît ;
- Contrairement aux automates à états finis, il n'y a pas d'équivalence entre automates à piles déterministes et non-déterministes : il existe des langages hors-contexte non reconnaissables par un automate à pile déterministe.

1.2.5 RTNs / ATNs

Un réseau de transitions récursif (Recursive Transition Network : RTN) est défini par ensemble de graphes semblables à ceux d'un automate fini dans lequel chaque transition permet d'atteindre un état terminal ou non-terminal. La différence par rapport à un automate à états finis se situe au niveau du traitement des états non-terminaux : le RTN traite chaque état non-terminal comme un éventuel appel à d'autres réseaux (y compris le RTN lui-même)

Définition 9 (Réseau de Transition Récursif - RTN) :

Un RTN peut être défini formellement par un 6-uplet $M = (Q, I, \Sigma, \delta, q_0, F)$, où :

- Q est un ensemble fini non vide d'états ;
- I est l'ensemble des états sous-initiaux (états qui étiquettent au moins une transition du transducteur RTN, et représentent donc un appel récursif au sous-RTN) ;
- Σ est un alphabet de symboles complexes. Chaque symbole est constitué d'une paire (e,s) avec e $\in$ à un alphabet d'entrée E, et s $\in$ à un alphabet de sortie S. $\Sigma \subseteq E \times S$;
- $\delta : Q \times (\Sigma \cup E \cup \{\varepsilon\}) \rightarrow Q$ est la fonction de transition ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble non vide des états finaux.

Un réseau de transitions récursif (RTN) peut être étendu pour donner un réseau de transitions augmenté (ATN : Augmented Transition Network). Un ATN est un RTN auquel s'ajoutent certaines extensions qui lui donnent un pouvoir descriptif supérieur à celui d'une grammaire non-contextuelle. Trois extensions sont apportées, il s'agit notamment :

- d'ajouter des registres aux réseaux de transitions ;
- d'imposer des conditions sur les transitions ;
- d'associer des actions aux transitions effectuées.

1.2.6 Les machines à états finis dans NooJ

La plateforme linguistique NooJ fait usage des différentes machines à états finis citées ci-dessus afin de représenter des données, formaliser des phénomènes linguistiques et analyser des textes :

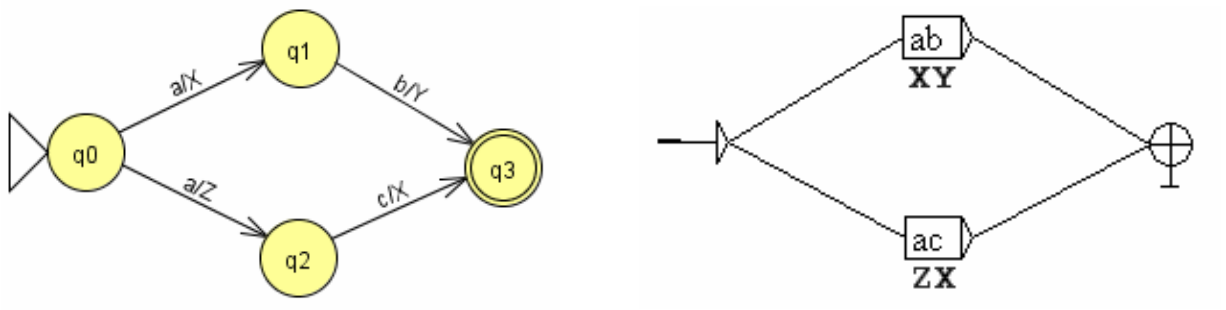
- Les expressions rationnelles représentent un moyen rapide pour les requêtes simples. Par exemple, lorsque la séquence recherchée consiste en quelques mots, il est possible d'énumérer ces mots directement dans une expression rationnelle ;
- Les transducteurs à états finis peuvent servir à décrire divers phénomènes linguistiques; notamment pour associer chaque patron retenu à un résultat d'analyse. Ils sont aussi utilisés pour stocker des données lexicalisées ainsi que toutes les informations morpho-syntaxiques qui s'y rattachent ;

- Les automates à états finis ne sont qu'un cas particulier des transducteurs : ils produisent le mot vide. Ils servent à localiser des phénomènes morpho-syntaxiques dans un corpus, extraire les séquences reconnues, construire des tables de concordances, etc. ;
- Les RTNs se présentent sous la forme d'ensembles organisés de graphes. Ces graphes peuvent eux-mêmes être des automates, des transducteurs à états finis ou des RTNs. Dans la pratique, les RTNs sont utilisés pour construire des bibliothèques de graphes facilement réutilisables ;
- Les ATNs sont des RTNs qui contiennent des variables et produisent des contraintes ou des productions complexes. Les variables permettent de stocker les séquences reconnues ; leurs contenus sont ensuite utilisés pour effectuer des transformations (par exemple : passer de la voix active à la voix passive).

Bien que théoriquement équivalentes aux définitions formelles ci-dessus citées, NooJ en utilise d'autres pour décrire les différentes machines à états qu'il met en œuvre. En l'occurrence, un transducteur à états finis est défini comme étant un quintuplet $T = (N, \Sigma, C, N_0, N_T)$, où:

- N est un ensemble fini non vide de nœuds ;
- Σ est un alphabet de symboles complexes. Chaque symbole est constitué d'une paire (e, s), où e $\in$ à un alphabet d'entrée E, et s $\in$ à un alphabet de sortie S. $\Sigma \subseteq$ E x S ;
- C est un ensemble fini non-vide de connexions entre les nœuds de N ;
- $N_0 \in$ N est le nœud initial ;
- $N_T \in$ N est le nœud terminal;

Ci-dessous, nous reprenons le transducteur de la Figure 3 et nous représentons son équivalent selon le formalisme utilisé dans NooJ. Les deux transducteurs reconnaissent les mêmes séquences « ab » et « ac » pour produire respectivement « XY » et « ZX ».



Représentation formelle (cf. Figure 3) Représentation dans NooJ

Figure 4 : Transducteur à états finis dans NooJ

Nous notons aussi que NooJ propose des façons équivalentes pour construire des machines à états finis. Nous pouvons citer :

- expressions rationnelles équivalentes aux automates ;
- expressions rationnelles à production équivalentes aux transducteurs ;
- grammaires de réécriture équivalentes aux RTNs
- grammaires avec variables équivalentes aux ATNs.

1.3 Construction et minimisation des automates : algorithmique

En se basant sur les définitions et propriétés citées plus haut, tout langage formé par l'ensemble des mots construits à partir des symboles d'un alphabet donné, peut être décrit sous la forme d'un automate à états finis, acyclique et déterministe. Le langage à reconnaître pourrait être assez large. Plusieurs travaux se sont dirigés vers l'optimisation de cette représentation par le biais d'une minimisation, voire une compression de l'automate construit.

1.3.1 Préliminaires mathématiques

Étant donné un automate déterministe A, la minimisation consiste à calculer un autre automate A' équivalent à A dont le nombre d'états est minimal. Cet automate est unique à un isomorphisme d'automate près. Les deux automates A et A' reconnaissent le même langage.

Soit A = (Q, Σ, δ, q_0, F) un automate à états finis déterministe. On se propose de réduire A et de construire l'automate minimal équivalent à l'aide des définitions suivantes:

Définition 10 (Automates équivalents) :

Deux automates A et A' sont dits équivalents s'ils acceptent le même langage, c'est-à-dire si L(A) = L(A').

Théorème de Moore, 1956 :

Etant donné un automate déterministe A, il existe un et un seul automate déterministe minimal A' qui reconnaît le même langage à une renumérotation des états près.

Définition 11 (Etat accessible) :

Un état q ∈ Q est dit accessible s'il existe un chemin (un mot) qui part de l'état initial et mène à q, on note : $q \text{ est accessible si } \exists w \in \Sigma^* (\delta^*(q_0, w) = q)$

Un état q ∈ Q est dit coaccessible s'il existe un chemin qui part de l'état q et arrive jusqu'à un état final, on note : $q \text{ est coaccessible si } \exists w \in \Sigma^* : \delta^*(q_0, w) = q', q' \in F$.

Un état est inaccessible lorsqu'aucune transition n'arrive dessus. Un automate où les états sont tous accessibles et coaccessibles est dit accessible, on note :
$Accessible(A) \text{ ssi } \forall q \in Q, \exists x \in \Sigma^* (\delta^*(q_0, x) = q)$

Dans l'exemple ci-dessous (voir Figure 5) : q_0 est l'état initial, q_1 et q_2 sont deux états terminaux. En partant de l'état initial q_0, l'état q_1 ne peut jamais être atteint, il est dit donc inaccessible.

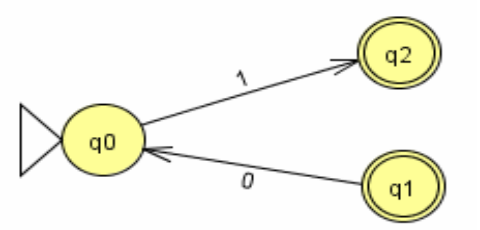


Figure 5 : Automate fini avec un état inaccessible

Définition 12 (Etats équivalents) :

Deux états p et q sont dits équivalents, notés p ≡ q, si on peut atteindre un état final en partant de p ou en partant de q en utilisant le même chemin. On note :
$p \equiv q \text{ ssi } \forall w \in \Sigma^* : \delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F$

Notons aussi que deux états sont équivalents s'ils vérifient les conditions suivantes :

- $p \equiv q \Leftrightarrow \forall x \in \Sigma : \delta(p, x) \; R \; \delta(q, x)$
- $p \equiv q \Rightarrow p \in F \Leftrightarrow q \in F$

Minimiser un automate consiste à diminuer le nombre d'états de telle sorte qu'il n'existe pas deux états équivalents. La minimisation d'un automate déterministe *A* se fait en deux temps :

- ✓ La première étape consiste à supprimer tous les états inaccessibles dans *A* ;
- ✓ La seconde étape repose sur le regroupement des états équivalents.

L'automate acyclique déterministe minimal obtenu est A' = (Q', Σ', δ ', q_0', F') tel que :

- l'alphabet Σ' = Σ ;
- l'ensemble des états Q' formé par des états tous distincts (ou non équivalents) ;
- la fonction de transition δ ': Q'* Σ' →Q' ;
- le nouvel état initial q_0' représente l'état initial q_0 ;
- les nouveaux états finaux représentent les états finaux de départ.

Formellement, nous notons que :
$A \text{ est Minimal ssi } \forall q, q' \in Q, (q \neq q' \Rightarrow L(q) \neq L(q')) \wedge Accessible(A)$

1.3.2 Algorithmes de construction et de minimisation

Dans cette section, nous présentons une classification chronologique des divers algorithmes de construction d'automates finis, déterministes et minimaux. Cette taxonomie a été, en partie, décrite dans [Watson, 2001] et [Watson, 2003]:

- Depuis les années cinquante, les premières recherches sur la minimisation ont été abordées par Huffman [Huffman, 1954] et Moore [Moore, 1956], elles ont permis le calcul d'un automate minimal. Le principe de leur algorithme de minimisation consiste à partir d'une partition grossière de l'ensemble des états de l'automate puis, de raffiner cette partition jusqu'à obtenir une partition stable par la fonction de transition ;
- Au début des années soixante, Brzozowski [Brzozowski, 1962] utilise une autre stratégie qui permet de calculer, en temps exponentiel, l'automate minimal d'un langage à partir d'un automate non déterministe spécifiant ce langage ;
- En 1971, Hopcroft [Hopcroft, 1971] propose un nouvel algorithme de minimisation qui a l'avantage, par rapport à celui de Moore, d'éviter des insertions d'éléments lorsqu'elles sont inutiles ; il améliore donc le temps de calcul. Ces travaux ont été repris et plusieurs algorithmes ont en découlé ;
- Vers la fin des années 1980, Maurice Gross a conçu les dictionnaires électroniques de type DELA en utilisant des automates à états finis. Ses algorithmes ont été mis en œuvre par Max Silberztein au sein de la plateforme de développement INTEX, l'une des premières applications linguistiques utilisant la technologie à états finis pour stocker de gros volumes de données [Silberztein, 1993]. L'algorithme de minimisation proposé repose sur deux étapes :
  i) construire un arbre lexicographique ;
  ii) minimiser l'arbre construit par fusion des états équivalents à l'aide d'un parcours ascendants de l'arbre.

  Cet algorithme a été hérité d'INTEX pour être employé au sein du moteur linguistique de son successeur NooJ dans ses différentes versions v1.x ;
- Au début des années 1990, Dominique Revuz propose le premier algorithme de minimisation linéaire des automates acycliques. Cet algorithme, basé sur une optimisation des travaux de Hopcroft, procède en trois étapes :
  i) construire un arbre lexicographique ;
  ii) trier les états de l'arbre par hauteur ;
  iii) itérativement sur la hauteur, fusionner les états de même hauteur, même contenu et mêmes transitions sortantes.

  Cette méthode de minimisation est rapide mais gourmande en mémoire [Revuz, 1991] et [Revuz, 1992];

- Plus tard, entre 1995 et 1998, plusieurs équipes de recherche ont travaillé indépendamment sur plusieurs variantes algorithmiques autour de la minimisation incrémentale dont la plupart sont identiques ou très similaires. A ce propos, nous citons les travaux de Watson, Daciuk et Mihov qui ont conçu un algorithme de minimisation incrémentale permettant l'ajout de mots triés dans l'ordre lexicographique. Ensuite, plusieurs variantes d'algorithmes ont été implémentées pour donner plus de facilités de maintenance telle que la suppression des mots du langage accepté par l'automate, tout en maintenant la minimalité de celui-ci [Daciuk, 1998 ; Watson, 1998 ; Mihov, 1999 ; Mihov et Maurel, 2000];
- En 2000, Daciuk a proposé un algorithme généralisé de minimisation incrémentale utilisant une nouvelle stratégie [Daciuk et al., 2000]. Cet algorithme permet l'insertion des mots du langage disposés dans un ordre lexicographique quelconque. Il se base sur l'ajout des mots les uns après les autres dans un automate acyclique déterministe en maintenant la minimalité de celui-ci après chaque ajout. Cet algorithme est économe en mémoire, mais plus lent en temps de calcul ;
- En 2001, Graña et al. proposent un récapitulatif des résultats obtenus par les algorithmes ci-dessus ainsi que leurs principales variantes améliorées [Graña et al., 2001] ;
- Ensuite, en 2002, Carrasco et Forcada ont dérivé une généralisation de l'algorithme de Daciuk de manière à manipuler des automates cycliques [Carrasco et Forcada, 2002] ;
- Dans [Watson, 2003], B.W. Watson a proposé une nouvelle variante récursive basée sur l'algorithme de minimisation Brzozowski.

Dans ce qui suit, nous détaillerons la description des trois principales routines de mise en œuvre des machines à états finis représentant des dictionnaires électroniques :

- L'algorithme utilisé dans NooJ, dans sa première version V1.x, ainsi que son prédécesseur INTEX : la construction à deux niveaux ;
- L'algorithme de construction linéaire proposé par Dominique Revuz ;
- L'algorithme de construction séquentielle proposé par Jan Daciuk.

1.3.2.1 Construction à deux niveaux

NooJ est la plateforme linguistique de développement qui sera utilisée tout au long de ce mémoire comme étant notre outil de travail. Le moteur lexical de sa première version, NooJ V1.x, héritait de son prédécesseur INTEX une méthode traditionnelle pour la construction de ces dictionnaires électroniques sous la forme d'automates à états finis, déterministes et minimaux. Cet algorithme de construction reposait sur deux étapes :

- construction d'un arbre lexicographique ou TRIE (prefix-tree). Cet arbre pourrait être considéré comme un automate fini déterministe, non minimal représentant l'ensemble des mots du langage ;
- minimisation linéaire de l'arbre construit en vue de réduire au minimum l'automate construit. Cette minimisation repose sur une fusion des états équivalents à l'aide d'un parcours ascendants de l'arbre.

L'exemple, ci-dessous, nous servira d'illustration de la démarche de construction des automates de dictionnaires électroniques dans NooJ. Considérons le langage réduit constitué des trois entrées : chats, chiens, rats.

La première étape de construction de l'automate reconnaissant les mots de ce langage donne un arbre lexicographique. Cette construction utilise la notion du plus long préfixe commun entre un mot et un ensemble de mots. Le calcul du plus long préfixe commun est utile pour ne pas créer deux fois un chemin commun à deux mots. Par exemple, les deux mots 'chats' et 'chiens' admettent la séquence 'ch' comme préfixe commun : l'arbre résultant utilise les mêmes états représentants ce préfixe, insérés lors de l'ajout du mot 'chat', au moment de l'ajout du deuxième mot 'chiens'.

L'arbre produit est le suivant :

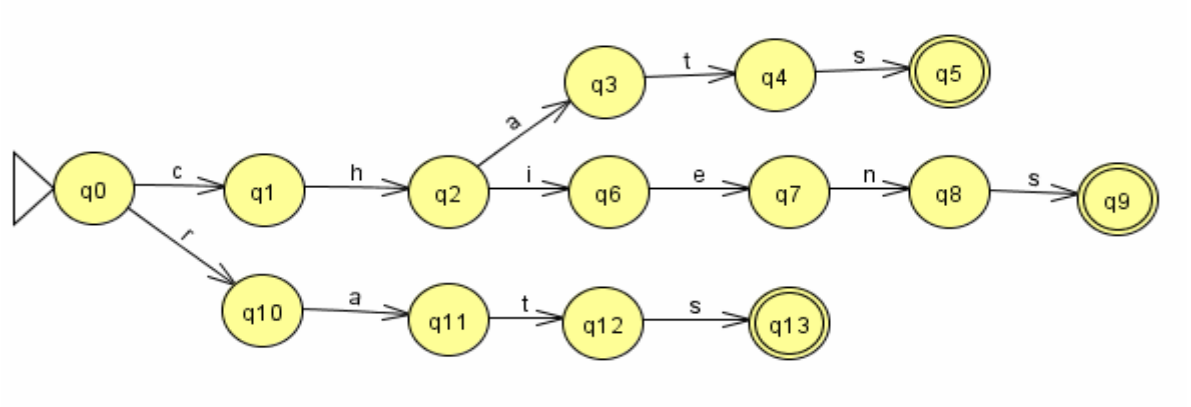


Figure 6 : Construction d'un arbre lexicographique

La deuxième étape de l'algorithme de construction consiste en la minimisation linéaire de l'arbre lexicographique par le biais d'un parcours ascendant. Le résultat de cette minimisation produit :

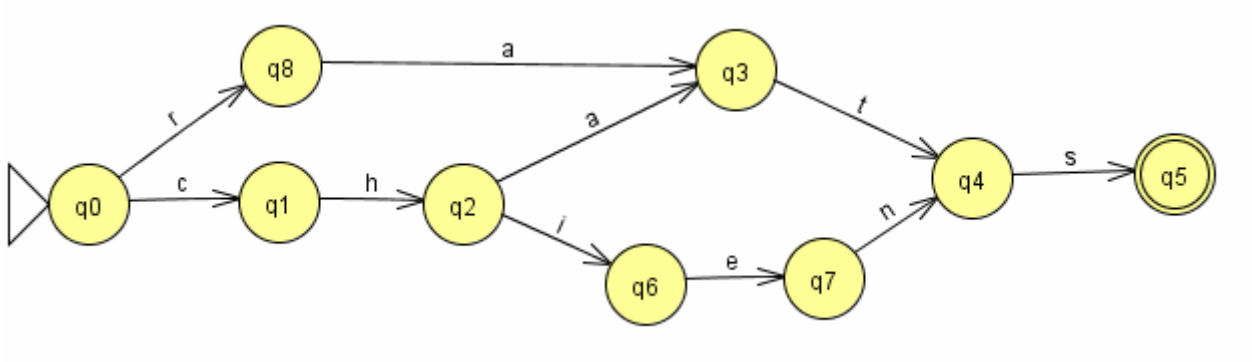


Figure 7 : Minimisation de l'arbre lexicographique

Nous signalons que cette méthode est très gourmande en mémoire principalement lors de la première étape à cause de l'insertion de plusieurs états inutiles dans l'automate initial. Cet arbre lexicographique pourrait, dans des cas réels, dépasser la mémoire physique disponible. Dans ce cas, malgré la linéarité garantie par l'algorithme de minimisation, celle-ci reste inefficace et la construction devient impossible. L'exemple de la construction du dictionnaire du hongrois peut nous fournir un ordre de grandeur réel : ce dictionnaire contient plus de 50 000 entrées associées à plus que 130 million de formes fléchies, l'arbre lexicographique construit contiendrait plus de 100 millions d'états. Avec une moyenne de 10 octets nécessaires pour le stockage de chaque état, la structure de données ainsi construite devient non gérable par les environnements de développements actuels, y compris ceux les plus performants comme l'environnement de développement de Visual Studio.

1.3.2.2 Construction linéaire

Au début des années 1990, Dominique Revuz part du constat qu'il était possible d'éviter le passage par deux étapes pour la construction d'un automate déterministe minimal à partir d'une liste de mots formant un langage particulier. Il a mis au point deux algorithmes :

- Le premier algorithme construit, pour le langage L, un automate pseudo-minimal en un temps proportionnel à la somme des longueurs des mots de L. Cet algorithme procède à une première minimisation, en comprimant certains suffixes communs à plusieurs mots ;

- Un second algorithme de minimisation des automates acycliques transforme ensuite l'automate pseudo-minimal en l'automate minimal.

L'algorithme de pseudo-minimisation utilise les suffixes communs des mots de notre langage pour réduire le nombre d'états dans l'automate généré. La pseudo-minimisation permet donc, de réduire le coût de la minimisation dont la complexité dépend du nombre d'états et de transitions. Paradoxalement à la première méthode de construction qui utilise la notion du plus long préfixe commun, cet algorithme utilise la notion du plus long suffixe commun entre un mot et un ensemble de mots. Ce plus long suffixe commun est utile pour ne pas créer deux fois un chemin commun à deux mots. Notons que, le plus long suffixe commun entre un mot et un ensemble de mots est le plus long suffixe du mot qui est aussi suffixe d'un mot de l'ensemble. Ce plus long suffixe commun, noté **lsc**(*w*,*X*), permet de réutiliser le chemin déjà créé dans l'automate.

Pour simplifier la recherche du plus long suffixe commun, les mots sont ajoutés à l'automate dans l'ordre lexicographique inverse (les comparaisons lettre à lettre se faisant de droite à gauche), ce qui permet d'entrer consécutivement le mot ayant le plus long suffixe commun avec le dernier mot inséré.

Lors de l'ajout des mots, dans l'automate, dans un ordre lexicographique inverse, l'algorithme itératif de pseudo-minimisation a le comportement suivant:

- Tant qu'une transition étiquetée par la lettre courante existe et que l'état atteint par cette transition n'a qu'un prédécesseur, on avance sur cet état et sur la lettre suivante ;
- Tant que la transition existe mais l'état atteint a plus d'un prédécesseur, on duplique l'état atteint par la transition que l'on redirige vers ce dupliqué. Ensuite, on avance sur le dupliqué et sur la lettre suivante ;
- Tant que le plus long suffixe commun n'est pas atteint, on crée une nouvelle transition étiquetée par la lettre courante et un nouvel état sur lequel on transite et on avance sur la lettre suivante ;

Si le plus long suffixe commun est atteint,

- Tant que l'état du plus long suffixe commun appartient au chemin décrit par le préfixe du mot dans l'automate, on crée une nouvelle transition étiquetée par la lettre courante et un nouvel état sur lequel on avance ;
- Si le plus long suffixe commun ne coïncide pas avec le chemin décrit par le préfixe du mot, alors on crée une transition qui y mène et on le suit jusqu'à la fin du mot.

Quand la fin du mot est atteinte dans l'une des boucles précédentes, soit 'd' le dernier état atteint :
Si 'd' n'est pas terminal, de deux choses l'une :

- Si **d** a plus d'un prédécesseur, on duplique **d** en **d'**, et **d'** devient terminal ;
- Si **d** a un seul prédécesseur, il devient terminal.

Cet algorithme a une complexité linéaire en $O(T)$, où T représente la taille du lexique ; Il permet de construire un arbre lexicographique. L'étape suivante sert à trier les états de cet arbre par hauteur ; la hauteur H d'un état q est égale à la longueur du plus long chemin partant de cet état **q**, et aboutissant à un état terminal.

Enfin, itérativement sur les hauteurs calculées, un algorithme de minimisation fusionne les états équivalents et élimine les états redondants.

Cette méthode de minimisation est relativement rapide par rapport à la première méthode de construction à deux niveaux. Toutefois, elle reste assez gourmande en mémoire.

1.3.2.3 Construction séquentielle

Contrairement aux deux méthodes décrites ci-dessus, l'algorithme de minimisation proposé par Jan Daciuk procède à l'ajout des mots les uns après les autres à l'automate en maintenant la

minimalité de celui-ci après chaque ajout [Daciuk et al., 2000]. Il s'agit alors d'une minimisation incrémentale ne nécessitant aucune intervention supplémentaire à la fin de l'ajout. En effet, avant d'insérer un mot, l'algorithme recherche les plus longs préfixes et suffixes communs entre le mot et l'automate qui ne se chevauchent pas, ensuite, il ajoute ce qui reste du mot entre la fin du préfixe et le début du suffixe. La différence majeure avec les algorithmes précédents réside dans la possibilité d'obtenir un automate partiellement minimisé en interrompant l'algorithme.

La première version de cet algorithme a été restreinte à l'ajout de mots déjà triés dans l'ordre lexicographique. Cependant, à défaut d'espace disque pour le stockage et le tri des données ou à cause de la provenance de ces données à partir d'un autre programme ou une source physique, il est parfois difficile ou même impossible de trier à l'avance les données du langage d'entrée avant de procéder à l'ajout des celles-ci dans un automate.

Dès lors, une deuxième variante a été proposée par Daciuk pour permettre la prise en compte de données disposées dans un ordre aléatoire. Néanmoins, l'ajout de mots non ordonnés en maintenant la minimalité de l'automate intermédiaire pourrait conduire à des modifications des parties de l'automate à chaque insertion. Une liste d'opérations doit être, alors, entreprise à chaque fois qu'un mot est ajouté :

- Rechercher le plus long préfixe commun entre le mot et l'automate déjà construit ;
- Suivre le préfixe commun à moins qu'un état de confluence[4] soit rencontré ;
- Enlever le dernier état rencontré du registre R[5] ;
- Cloner tous les états dans le préfixe à partir du premier état de confluence rencontré ;
- Créer un chemin pour le suffixe et marquer l'état final ;
- A partir de l'état final vers l'état de départ, i .e. premier état de confluence rencontré, soit ajouter les états du chemin créé dans le registre R, soit les remplacer par des états qui y sont équivalents ;
- Arrêter le processus d'ajout ou de remplacement d'états lorsqu'il n'y a plus de changements qui surviennent.

Notons que:

- En cas d'absence d'états de confluence, il suffit d'ajouter le mot suffixe au dernier état du suffixe commun ;
- En cas de présence d'états de confluence au niveau du préfixe commun, le simple ajout d'une transition à partir du dernier du chemin commun ajouterait, accidentellement, plus de mots que désirés (voir Figure 9) ;
- Les états antérieurs dans le chemin du préfixe commun peuvent nécessiter d'être changé à cause des changements qui auraient été engendrés au niveau de l'ensemble des mots acceptés à partir de cet état, *L(q)*. Un recalcul des équivalences des états recalculer la relation d'équivalence pour tous les états du chemin du nouveau mot ;
- Pour vérifier si deux états q et q' sont équivalents, nous avons seulement à vérifier s'ils sont terminaux ou non, et si leurs suites de transitions sont identiques (les mêmes numéros, étiquettes et états cibles).

L'exemple ci-dessous, va nous servir à illustrer la méthode de construction incrémentale des automates. Nous reprenons le même langage réduit constitué des trois entrées : chats, rats, chiens.
Après ajout des deux premiers mots du langage (« chats » et « rats »), nous obtenons l'automate minimal suivant :

[4] Un état de confluence est un état ayant plus qu'une seule transition entrante.
[5] Le registre R contient tous les états q_i pour lesquels l'ensemble $L(q_i)$ est restreint à un seul mot.

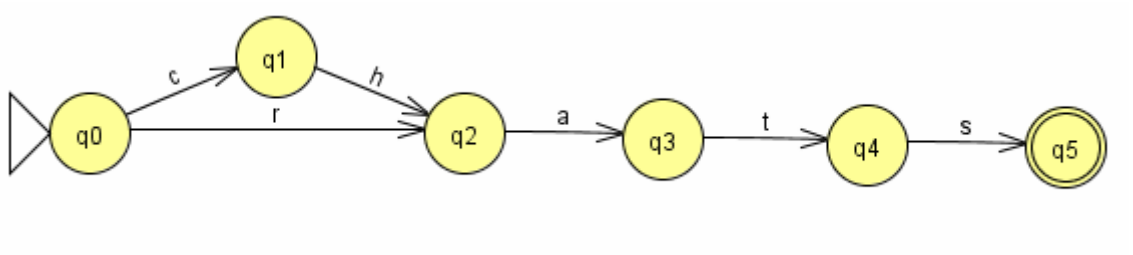


Figure 8 : Insertion des deux entrées « chats » et « rats » dans un automate

Le résultat de l'ajout direct du mot « chiens » à ce dictionnaire minimisé qui contient les mots « rats » et « chats » produit le dictionnaire, ci-dessous, qui reconnait, accidentellement, le mot « riens ».

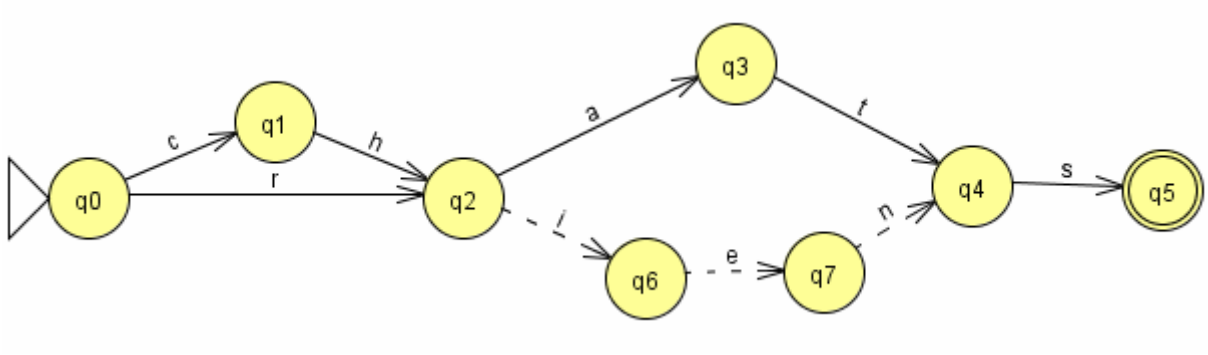


Figure 9 : Ajout de l'entrée « chiens » aveuglément

Nous observons que l'état q_2, dernier état du chemin du plus long préfixe commun soit la séquence 'ch', doit être cloné avant l'ajout du mot qui reste à insérer. L'état ainsi construit est l'état q_8.

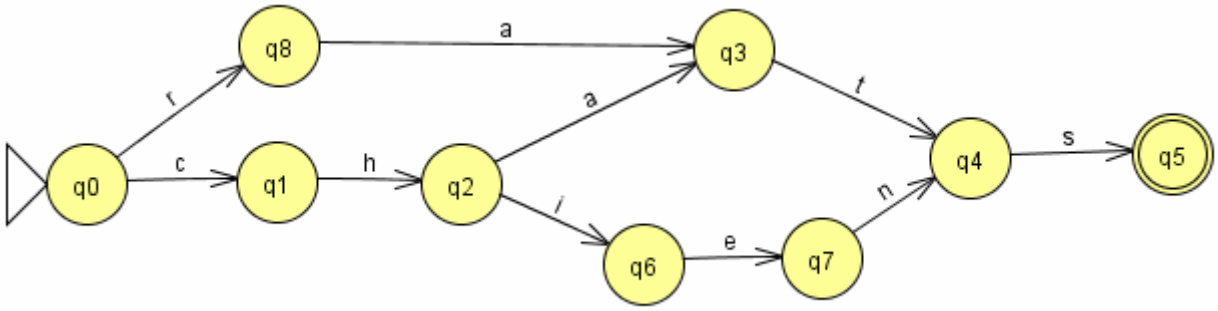


Figure 10 : Construction séquentielle de l'automate

Nous remarquons, alors, que la minimisation de l'automate à la volée peut changer les classes d'équivalence de certains états à chaque fois qu'un mot est ajouté. Avant de réellement construire un nouvel état dans l'automate, il faut d'abord vérifier s'il pourrait être équivalent à un état déjà existant. Cela conduit à un algorithme de construction incrémentale dont l'automate résultant est minimal à l'issue de l'insertion de chaque mot du langage.

Dans un premier lieu, nous avons développé et testé cette méthode sur des langages à grandeurs réelles. Dans un second lieu, nous l'avons inclus dans le moteur lexical de NooJ dans sa version v2.0.

Une comparaison entre la Figure 7 et la Figure 10 permet d'illustrer qu'indépendamment de la méthode de minimisation utilisée, l'automate fini, déterministe, acyclique et minimal représentant un langage donné est le même à une numération d'états près.

Cependant, lorsqu'il s'agit de données à grandeurs réelles, même les algorithmes de minimisation les plus optimaux mènent à des automates dont la taille est très importante. A titre d'exemple, nous citons le dictionnaire DELAF[6] de la langue française qui contient environ huit cent mille mots [Courtois et Silberztein, 1990], son automate fini, déterministe et minimal compte environ 80 000 états et 177 465 transitions. Les recherches se sont alors orientées vers l'exploration d'autres pistes de réduction de tel automate, notamment par le biais de méthodes de compression.

1.4 Compression et optimisation des automates

Dans la littérature, nous retrouvons plusieurs techniques de compression. Lors de la manipulation de données des langues naturelles, contrairement aux cas de traitement d'images et de sons, la compression doit obligatoirement s'effectuer sans perte. En effet, une fois décompressées, toutes les données manipulées devraient être restituées à l'identique.

La compression d'automates à états finis et minimaux repose sur 3 principales techniques :

- Codage des entrées de l'automate ;
- Réduction de la taille de certains éléments de l'automate ;
- Partage de l'espace occupé par de certaines parties redondantes de l'automate.

1.4.1 Codage des entrées

Comme déjà signalé, dans ce mémoire, les automates à états finis sont utilisés pour stocker de gros volumes de données tels que les dictionnaires électroniques représentant le lexique d'une langue naturelle. Pour répondre aux attentes des différentes applications en TALN, la consultation d'un dictionnaire ne doit pas se restreindre à la vérification de la reconnaissance d'un mot en entrée. Cette consultation doit fournir deux informations :

- La reconnaissance ou non du mot en entrée ;
- L'ensemble des informations qui peuvent y être associées.

Pour fournir de telles informations lors de la consultation d'un dictionnaire électronique, NooJ, notre plateforme linguistique de développement, n'utilise pas de dictionnaires à deux niveaux par les transducteurs ; il exprime cette structure à l'intérieur du mot de la langue acceptée par l'automate. L'entrée du dictionnaire est donc divisée en deux parties : le mot et l'ensemble des annotations associées à ce mot. Ces annotations serviront pour répondre aux différents types de requêtes susceptibles d'être soumises à l'application chargée de la consultation de la base lexicale. Nous avons alors besoin d'associer chaque mot avec son lemme, sa catégorie grammaticale (nom, verbe, etc), ses caractéristiques morphologiques (Genre, Nombre,…), ses informations syntaxiques et sémantiques tels que le nombre et le type des compléments d'un verbe, la liste des verbes supports associée à un nom prédicatif, les traductions potentielles, etc.

En effet, la simple consultation d'un dictionnaire électronique dans NooJ permet d'associer à chaque forme reconnue la liste des informations qui y sont associées. Cette liste contient principalement : son lemme, sa catégorie grammaticale (ex. : Nom), les codes flexionnels (ex. : masculin, pluriel, participe passé,…), les codes syntaxiques (ex. : "verbe transitif"), code distributionnel (ex. : "Humain") et domaine sémantique (ex. : "terme médical").

[6] DELAF : Dictionnaire Electronique du LADL pour le Français

L'exemple, ci-dessus, reprend les mêmes entrées utilisées lors de la description des algorithmes de minimisation. Les exemples donnés représentent trois analyses typiques des trois formes :

```
rats,rat,N+Animal+p
chats,chat,N+Animal+p
chiens,chien,N+Animal+p
```

Notez que ces trois formes fléchies sont associées à la même chaîne de l'information (N : nom, Animal, p : pluriel), mais les lemmes sont différents : "rat" pour "rats", "chat" pour "chats" et "chien" pour "chiens". L'ajout de ces mots associés à leur information dans un automate déterministe minimal pourrait générer le résultat suivant qui ressemble à un arbre lexicographique (TRIE) puisque ces différentes analyses conduiraient à différents états terminaux :

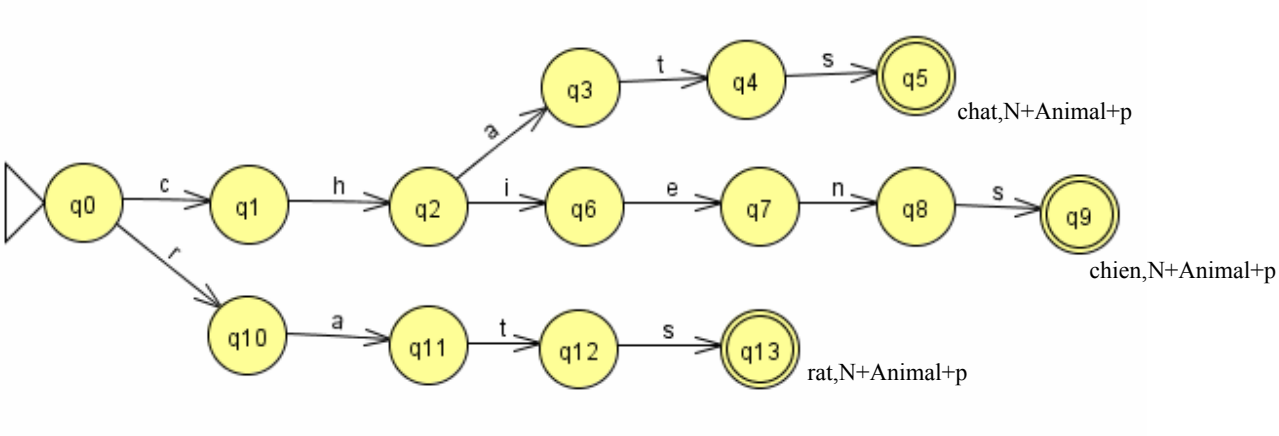


Figure 11 : Construction d'un automate et insertion des informations associées aux entrées

À mesure que nous associons des informations de plus en plus précises pour chaque entrée du dictionnaire, même les techniques de minimisation les plus optimales restent insuffisantes. L'étape de compilation nécessiterait d'avantage de mémoire pour le stockage, et donc la minimisation de l'automate est de moins en moins efficace puisque le résultat tend à ressembler à un TRIE (cf. Figure 11). Ceci entraîne une augmentation considérable de la taille du dictionnaire résultant parfois assez importante pour être traité par les PC actuels notamment pour les langues à morphologie complexe tels que l'arabe ou le hongrois. Par conséquent, une technique de compression est nécessaire pour la réduction de cette taille surtout si l'information devient de plus en plus précise.

Afin d'unifier les trois analyses, NooJ remplace chaque lemme par une commande basée sur l'utilisation d'un opérateur de suppression, soit l'opérateur morphologique <B> (B pour "BackSpace - retour arrière"). Cet opérateur permettra le calcul du lemme à partir de sa forme fléchie. En l'occurrence, pour calculer le lemme "chat" de l'entrée "chats", on supprime la dernière lettre de celle-ci ; cette suppression correspond à la commande "<B>".

A l'aide de ce codage, nous associons les trois mots "chats", "rats" et "chiens" à une analyse unique :

```
Chats,<B>,N+Animal+p
Rats,<B>,N+Animal+p
Chiens,<B>,N+Animal+p
```

Grâce à ce codage, un grand nombre d'entrées lexicales peut être associé à la même analyse. Ces analyses sont stockées dans une table de hachage. La taille de la table de hachage qui en résulte est réduite de façon significative, et, surtout, qu'un grand nombre d'entrées comme "chats" et "rats"

conduira à un petit nombre d'états terminaux. Ceci engendre un processus de minimisation très efficace (voir Figure 12):

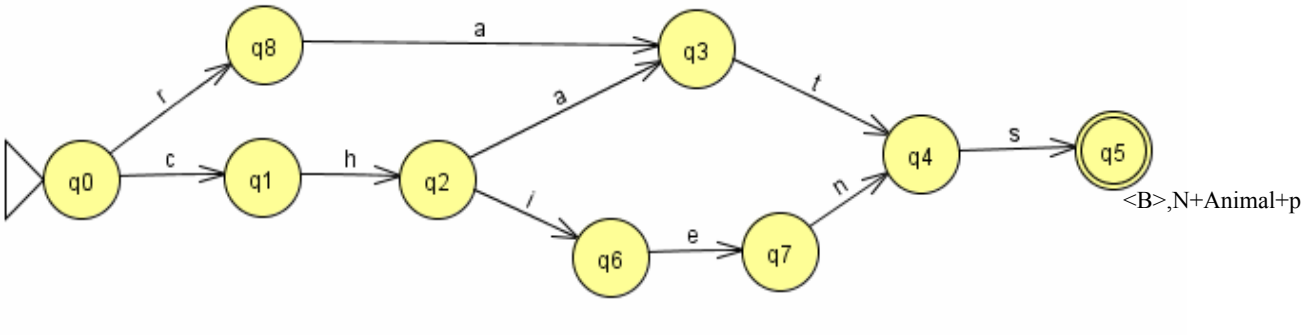


Figure 12 : Construction d'un automate et factorisation des informations associées aux entrées

En réalité, ce codage est formé par deux informations :

- le nombre de caractères à supprimer à partir de l'entrée du dictionnaire pour que le reste coïncide avec le début du lemme ;
- la chaîne de caractères qui devrait être ajoutée au résultat de la première opération de suppression pour former le lemme correspondant à la forme fléchie en question.

Par exemple, parmi les paradigmes flexionnels qui peuvent être traités avec cette méthode nous citons :

Entrées de dictionnaire	**Analyses**	**Opérations morphologiques**
aiderons,aider,V+F+1+p *aimerons,aimer,V+F+1+p*	**<B3>**	Supprimer les 3 dernières lettres de la forme fléchie : aider**ons** → aider
aidons,aider,V+P+1+p *aimons,aimer,V+P+1+p*	**<B3>er**	Supprimer les 3 dernières lettres de la forme fléchie (aid**ons** → aid), puis ajouter "er" (aid → aider)

Cette technique de compression fonctionne très bien lorsque la différence entre les mots et leurs lemmes se situe uniquement au niveau des suffixes, ce qui est le cas pour une majorité de formes fléchies pour le français et plusieurs autres langues romanes. Sauf que NooJ peut également être utilisé pour formaliser des préfixations. Par exemple, un dictionnaire NooJ peut être utilisé pour produire les analyses suivantes :

```
remonterons,monter,V+F+1+p
remangerons,manger,V+F+3+s
```

Si nous utilisons la même technique de compression pour calculer les lemmes (respectivement monter et manger) des formes fléchies en entrée (remonterons et remangerons), nous obtenons le codage suivant :

```
remonterons,<B11>monter,V+F+1+p
remangerons,<B11>manger,V+F+1+p
```

En conséquence, les deux formes fléchies "remonterons" et "remangerons" (ainsi qu'un grand nombre de formes, tels que «rechercherons», «renoterons», etc) s'analysent différemment, même si, logiquement, leurs analyses devraient être similaires.

L'automate déterministe minimal représentant le langage réduit aux deux formes ressemble à nouveau à un arbre lexicographique. Nous obtenons, alors :

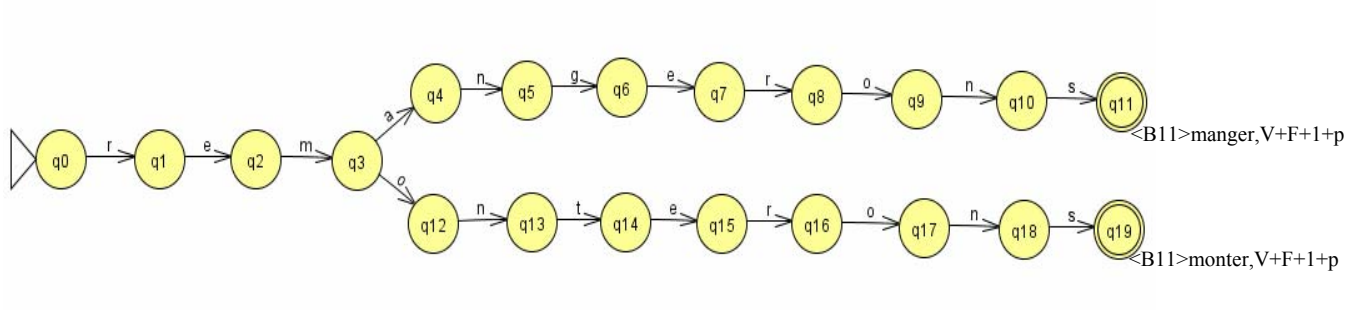


Figure 13 : Automate résultant de l'ajout des mots « remontera » et « remangera »

Cette technique de codage utilise uniquement l'opérateur <B> qui désigne le nombre de caractères devant être supprimés à partir de la fin de la forme fléchie afin que le reste puisse correspondre au début de la base de la forme, et fournit la séquence de lettres qui doit être annexée à la chaîne de caractères résultante de l'opération précédente pour former le lemme. Une telle solution pourrait être plus ou moins appropriée pour les langues qui n'utilisent pas de préfixes et infixes dans leurs morphologies flexionnelles, par exemple, le français et l'anglais. Pour d'autres langues, telles que les langues sémitiques, comme l'arabe ou l'hébreu, les dictionnaires sont représentés par automates qui ressemblent à de très larges TRIEs : le processus de minimisation ne pouvait pas empêcher l'automate de prendre une taille hors de contrôle. Une nouvelle méthode est alors requise à fin de lier les formes à leurs lemmes.

1.4.2 Réduction de la taille des constituants d'un automate

Après le développement de son algorithme de construction incrémentale, Daciuk a proposé les expérimentations de diverses méthodes de compression pour réduire des automates finis, déterministes, acycliques et minimaux représentant des dictionnaires de langues naturelles.
Initialement, il décrit un automate par l'ensemble des transitions sortantes de chaque état telle que chaque transition comporte quatre informations :

1) E : une étiquette ;
2) F : un booléen pour indiquer la nature de l'état d'arrivée (final ou pas) ;
3) # : le nombre de transitions sortantes de son état d'arrivée ;
4) → : un pointeur vers son état d'arrivée.

En se basant sur cette liste d'informations, J. Daciuk reproduit les automates représentant des dictionnaires électroniques sous forme d'une liste d'états, tel est le cas de l'automate de la Figure 14 que nous listons dans le Tableau 1. Cet automate reconnaît les deux formes « stay » et « say ».

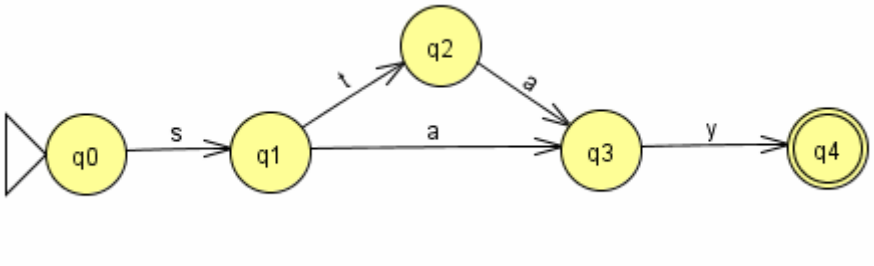


Figure 14 : Automate reconnaissant les formes : « say » et « stay »

	E	F	#	→
1	s		2	2
2	t		1	4
3	a		1	5
4	a		1	5
5	y	✓	0	0

Tableau 1 : Série des transitions de l'automate de la Figure 13

Ces méthodes de compression se basent sur la réduction de la taille de certains éléments de l'automate, elles peuvent être décrites comme suit :

- **Compression 1 : Elimination des compteurs de transitions :**
  Pour gagner en espace mémoire, cette méthode élimine le compteur de transitions (2) et le remplace par un booléen pour indiquer si la transition est la dernière de l'état courant. Cette méthode considère que chaque état comme une liste de transitions plutôt qu'un ensemble de transitions, voir Tableau 1. Au lieu de stocker le compteur des transitions, il ajoute, pour chaque état, un drapeau à 1-bit indiquant si cette transition est la dernière appartenant à un état en particulier. En pratique, ce drapeau qui figure dans le Tableau 2 dans la colonne S, peut être stocké dans le même espace mémoire que le pointeur pour la cible de l'état en cours.

	E	F	S	→
1	s		✓	2
2	t			4
3	a		✓	5
4	a		✓	5
5	y	✓	✓	0

Tableau 2 : Liste des transitions après élimination des compteurs de transitions

- **Compression 2 : Partage des transitions :**
  Pour économiser de l'espace, cette méthode représente une seule fois les transitions portant les mêmes étiquettes et aboutissant au même état d'arrivée telles que les transitions 3 et 4 du Tableau1. Ce procédé permet, donc, la réduction de l'ensemble des transitions de l'automate. Ainsi, nous obtenons la configuration suivante :

	E	F	#	→
1	s		2	2
2	t		1	3
3	a		1	4
4	y	✓	0	0

Tableau 3 : Série des transitions après partage des de transitions

  Toutefois, nous notons que cette compression n'est possible qu'en présence des compteurs de transitions ; elle est donc incompatible avec la première méthode de compression.

- **Compression 3 : Omission des pointeurs vers les états suivants :**
  Pour réduire la taille occupée par l'automate, J. Daciuk s'appuie sur les recherches de Kowaltowski dans [Kowaltowski et al., 1993] pour réorganiser l'automate et stocker les transitions formant une série les unes derrière les autres. Cette réorganisation permet de remplacer les pointeurs vers les états qui suivent directement dans la série par un booléen (c'est-à-dire, un drapeau à 1-bit). Ce drapeau est représenté sur la colonne D dans le tableau suivant :

	E	F	D	#	→
1	S		✓	2	
2	T			1	3
3	A		✓	1	
4	Y	✓		0	0

Tableau 4 : Série des transitions après omission des pointeurs vers les états suivants

- **Compression 4 : Partage des transitions finales** :
  Partant de la représentation obtenue par la 1ère compression qui permet de réduire une partie des états de l'automate (les états de départ des transitions aboutissant au même état d'arrivée et portant les mêmes étiquettes), la compression 4 traite à nouveau ces états lorsqu'ils possèdent encore des transitions sortantes qu'ils ne partagent pas. Elle réduit les transitions restantes en utilisant un booléen et un pointeur.

- **Compression 5 : Changement de l'ordre des transitions :**
  La 1ère et la 4ème compressions allègent l'automate en supprimant certaines transitions. Cependant, cette suppression est conditionnée par la structure de l'automate. En effet, l'ordre dans lequel sont stockées les transitions peut influencer sur la consommation mémoire. D'après l'étude menée par Kowaltowski [Kowaltowski et al., 1993], disposer les transitions sur chaque état par ordre décroissant selon la fréquence de leurs étiquettes permet une meilleure compression.

Ces méthodes ont toutes été testées sur les automates représentant différentes langues naturelles et les résultats obtenus montrent qu'il n'y a pas de compression optimale pour traiter tous les dictionnaires. Selon le dictionnaire, certaines méthodes peuvent être plus efficaces que d'autres. Les compressions les moins performantes sont les compressions 3 et 4.

1.4.3 Recherche des sous-automates redondants

La troisième technique de compression consiste à partager l'espace occupé par de certaines parties redondantes de l'automate. Une étude de la structure interne d'un automate a été menée par Lamia Tounsi [Tounsi, 2007]. Cette étude a soulevé, d'une part, la possibilité d'indexer des automates et, d'autre part, la possibilité d'utiliser des algorithmes dédiés initialement à l'indexation des textes pour le faire. Cette indexation a utilisé des solutions existantes pour le calcul de redondance dans un texte comme dans l'étude menée par Crochemore sur la recherche de motifs *(pattern matching)* [Crochemore et Rancart, 1997]. Par ailleurs, la recherche de sous-structures dans un automate rappelle le problème de recherche de composants dans un graphe.
L'étude de la structure interne d'un automate acyclique donné *A* a été effectué à partir de la recherche de tous ses sous-automates.

Un sous-automate est défini comme un automate isolé possible à extraire de *A*. La méthode de compression proposée considère trois types de sous-automates :

- sous-automate parallèle ;
- sous-automate série ;
- sous-automate minimal qui n'est ni parallèle, ni série.

La recherche des sous-automates redondants est basée sur un algorithme itératif qui calcule et factorise par une seule transition les transitions en parallèle et en série jusqu'à ce que l'automate en soit totalement dénué. Ensuite, il calcule et factorise par une seule transition les sous-automates minimaux, et ainsi de suite, jusqu'à ce que l'automate donné soit réduit à deux états reliés par une seule transition. Dans la Figure 15, l'automate initial contient plusieurs transitions en parallèle, par exemple celles entre l'état 5 et l'état 8 ou celles qui relient l'état 7 à l'état 10. La première étape permet d'aplatir directement toutes ces transitions pour obtenir un automate dénué de parallèle tel

est le cas de l'automate A2. La seconde étape concatène toutes les séries en une unique transition et donne l'automate A3. En aplatissant de nouveau les parallèles, nous obtenons l'automate A4 et en aplatissant ensuite les séries nous obtenons l'automate A5. L'algorithme se termine donc sur l'automate A5 qui ne présente plus aucune série ni aucun parallèle. Notons que, l'automate A5 reconnaît le même dictionnaire que l'automate A1, mais ses transitions se réfèrent soit à des lettres, soit à des sous-automates de A1 ; un algorithme d'indexation automatique a été développé à cet effet [Tounsi, 2007].

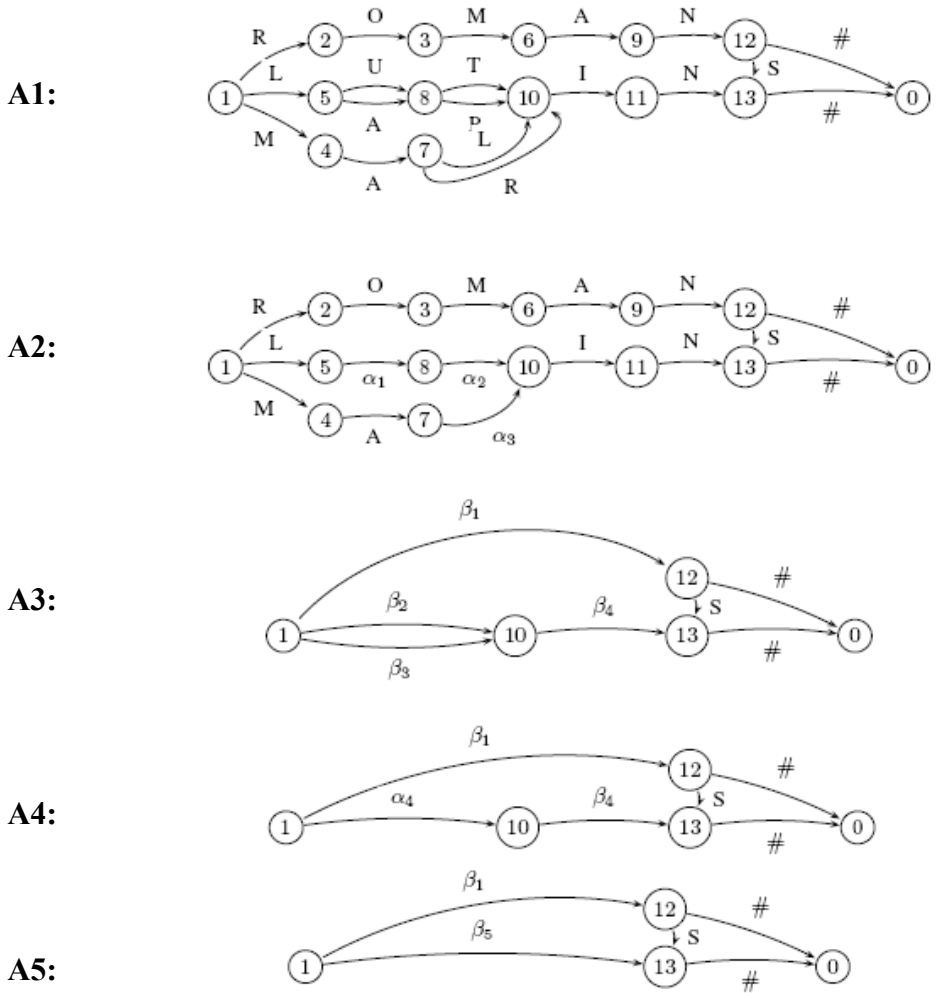


Figure 15 : Recherche de sous automates séries-parallèles

La complexité théorique de recherche des sous-automates séries et parallèles est de $O(m^2)$, où m représente le nombre de transitions dans *A* et dans le pire des cas, la complexité de l'algorithme de recherche des sous-automates minimaux qui ne sont ni parallèles ni séries est de $O(n^3)$ où *n* représente le nombre d'états de *A*.

Au niveau du traitement automatique des langues, cette méthode a été appliquée à plusieurs automates déterministes, acycliques et minimaux représentant des dictionnaires électroniques de différentes langues ; les résultats obtenus montrent que ces automates contiennent un nombre considérable de séries, de parallèles et dans une moindre mesure, des sous-automates issus de la réduction des parallèles et séries. De plus, certains de ces sous-automates possèdent plusieurs niveaux d'imbrications. Par la suite, un index est automatiquement construit pour énumérer la liste de tous les sous-automates ci-dessus recherchés. Cet index est partiel car il est construit en fonction des sous-automates séries et parallèles : il n'inclut pas les sous-automates minimaux.

1.5 Un nouvel algorithme de compression

Le compilateur de dictionnaires, intégré dans NooJ V1.x, ci-dessus décrit, présente quelques insuffisances qui se sont manifestées plus particulièrement lors de la manipulation de langues à morphologie complexe telle que l'arabe, l'hébreu ou le hongrois. Pour y remédier, nous avons implémenté une variante de l'algorithme de construction incrémentale proposé par J. Daciuk (voir paragraphe 1.3.2.3). Cet algorithme permet d'éviter la construction d'un TRIE intermédiaire pouvant, comme dans le cas du dictionnaire complet de l'arabe ou du hongrois, dépasser plusieurs millions d'états dont une grande partie serait redondante. Les performances du nouvel algorithme de minimisation incrémentale ont été, considérablement, améliorées par une nouvelle méthode de compression dynamique permettant la prise en compte des constructions complexes de la langue par suffixations, préfixations et/ou infixations.

1.5.1 Compression dynamique de l'automate d'un dictionnaire électronique

NooJ formalise les paradigmes flexionnels et dérivationnels au moyen de transducteurs à états finis. Lesquels transducteurs sont décrits à l'aide de l'éditeur de graphes dans NooJ ou par l'intermédiaire d'expressions rationnelles. L'entrée des transducteurs est une chaîne de lettres et d'opérateurs morphologiques : ils produisent l'analyse linguistique du mot formé. NooJ prévoit une douzaine d'opérateurs morphologiques autres que <B>, y compris les suivantes :

- <L> : passer à gauche ;
- <R> : passer à droite ;
- <S> : supprimer la lettre courante ;
- <D> : dupliquer la lettre ;
- <A> : supprimer l'accent ;
- Etc.

Le moteur morphologique de NooJ utilise ces opérateurs pour effectuer des transformations au sein des chaînes en entrée. Elles peuvent être associées à deux types d'arguments :

- un nombre : par exemple <L3>: aller à gauche 3 fois ;
- un «W» : par exemple <LW>: aller au début du mot.

Ces commandes fonctionnent sur une pile : chacune d'entre elles opère en temps constant. Ainsi, ils peuvent lier un lemme à toutes ses formes fléchies dans un temps linéaire.
Si nous reprenons l'exemple traité au paragraphe 1.4.1, nous considérons les deux entrées lexicales :

```
remonterons,monter,V+F+1+p
remangerons,manger,V+F+1+p
```

L'utilisation de l'algorithme traditionnel pour calculer les lemmes (monter et manger) à partir des formes fléchies en entrée (remonterons et remangerons) nous donne le codage suivant :

```
remonterons,<B11>monter,V+F+1+p
Remangerons,<B11>manger,V+F+1+p
```

Formes fléchies	Analyses	Opérations morphologiques
remonterons,	**<B11>monter**	Supprimer les 11 caractères qui forment la forme initiale, insérer la séquence des lettres "monter" → (monter)
remangerons	**<B11>manger**	Supprimer les 11 caractères qui forment la forme initiale, insérer la séquence des lettres "manger" → (manger)

Tableau 5 : Exemple de deux analyses morphologiques différentes

Pour remédier à un tel problème, une nouvelle technique de calcul de la correspondance entre la forme fléchie et le lemme qui s'y rattache a été implémentée. L'idée principale consiste à calculer, récursivement, tous les affixes communs entre l'entrée du dictionnaire et son lemme. Cette compression utilise les quatre opérateurs morphologiques suivants : <L>, <R>, <S> et <B> (et non plus seulement <B>).

Les formes fléchies du tableau précédent sont ainsi analysées de façon unifiée :

Formes fléchies	**Analyses**	**Opérations morphologiques**
remonterons, remangerons	**<B3><LW><S2>**	Supprimer les 3 derniers caractères (→ remonter), passer au début du mot, ensuite, supprimer les deux lettres suivantes (monter)

Tableau 6 : Exemple d'unification de l'analyse morphologique

L'automate obtenu ressemble à :

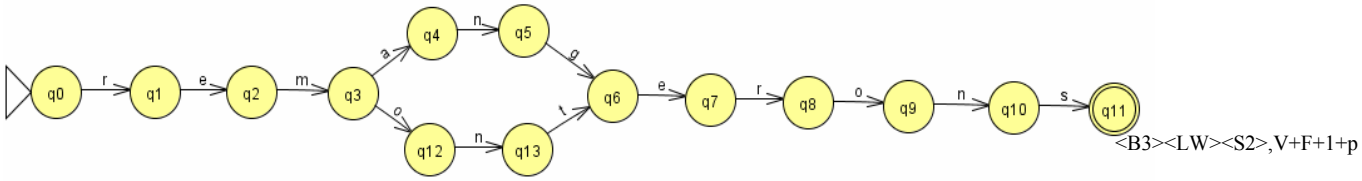


Figure 16 : Automate résultant de l'unification des analyses morphologiques

La nouvelle analyse peut être partagée par de nombreuses formes fléchies, ainsi ces formes conduiront à un état terminal commun dans l'automate représentant le dictionnaire, ce qui, à son tour, déclenche un processus de minimisation beaucoup plus efficace. En outre, la table de hachage qui stocke les différentes analyses est réduite en taille.

Cette méthode de compression est décrite dans l'algorithme CompressLemma ci-après :

Algorithme 1 : **CompressLemma : algorithme de compression**
Entrées : Entrée, Lemme
Sortie : commande de correspondance

```
string CompressLemma (string entrée, string lemme)
{
        if l'entrée est un préfixe du lemme
            return suffixe du lemme; // mange,manger → "r"
        else if le lemme un préfixe de l'entrée
            return "<B" + (longueur du suffixe de l'entrée)+">";
            // mangerons,manger → "<B3>"
        else
            return RecCompression("<LW>",entrée,lemme);
          // remangerons,manger → "<B3><LW><S2>"
}
```

Les deux premiers cas sont assez simples à résoudre ; ils représentent les cas où la forme fléchie est un préfixe du lemme et inversement. Le cas échéant, la résolution repose sur une localisation récursive de la plus longue sous-chaîne commune (Longest Common Substring – LCS) entre les deux chaînes en entrées.

A l'issue du troisième cas, le résultat de la fonction RecCompression est simplifié. A l'instar de l'exemple précédent, nous remplaçons la commande initialement calculée, "<LW><S3><R6><S3>", par la commande "<B3><LW><S3>". Les deux commandes sont équivalentes.

Algorithme 2 : **RecCompression : algorithme récursif de compression**
Entrées : commande, entrée, lemme
Sortie : commande de correspondance

```
string RecCompression (string commande, string entrée, string lemme)
{
    PlusLongueSsChaine = LongestCommonSubstring (entrée,lemme);
    Ind_Entrée = index de PlusLongueSsChaine dans l'entrée
    Ind_Lemme = index de PlusLongueSsChaine dans le lemme
    if ind_Entrée != 0 and ind_Lemme != 0
    {
       Gauche_Entrée = le contexte gauche de PlusLongueSsChaine dans entrée
       Gauche_Lemme = le contexte gauche de PlusLongueSsChaine dans lemme
       commande = RecCompression (commande, Gauche_Entrée, Gauche_Lemme);
       commande += "<R"+ taille de PlusLongueSsChaine + ">";
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans entrée
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans lemme
       commande = RecCompression (commande, Droite_Entrée, Droite_Entrée);
    }
    else if Ind_Entrée!= 0 and Ind_Lemme == 0
    {
       commande += "<S" + Ind_Entrée + ">";
       commande +="<R"+ taille de PlusLongueSsChaine + ">";
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans entrée
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans lemme
       commande = RecCompression (commande, Droite_Entrée, Droite_Entrée);
    }
    else if Ind_Entrée == 0 and Ind_Lemme != 0
    {
       commande += lemme.sous-chaine(0, Ind_Lemme);;
       commande +="<R"+ taille de PlusLongueSsChaine + ">";
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans entrée
       Droite_Entrée = le contexte droit de PlusLongueSsChaine dans lemme
       commande = RecCompression (commande, Droite_Entrée, Droite_Entrée);
    }
    return commande;
}
```

La reconnaissance de la plus longue sous-chaîne commune, LCS, pourrait être coûteuse en terme d'espace mémoire alloué. La méthode traditionnelle consiste à parcourir les deux chaînes et effectuer une comparaison pour chacun des m points de départ du lemme, pour la plus longue chaîne de caractères à partir de chacun des n points de départ de l'entrée. Dans ce cas la complexité est en $O(m^2 \cdot n)$ où n est la taille de la forme fléchie en entrée. Cette méthode serait très coûteuse en espace et temps de construction des dictionnaires, essentiellement lorsqu'il s'agit de plusieurs milliers voire des millions d'entrées lexicales. Pour réduire ce besoin, nous utiliserons une technique de programmation dynamique avec une complexité en $O(m \cdot n)$, où n et m sont respectivement la longueur de la forme fléchie et du lemme. L'algorithme de compression a alors une complexité totale en $O(m \cdot n \cdot \log(n))$.

La technique de programmation dynamique utilisée pour le calcul de la plus longue sous-chaîne commune a été, à l'origine, introduite dans les années 1940 par Richard Bellman. Il s'agit d'une méthode d'optimisation opérant par phases (ou séquences). L'efficacité de cette méthode repose sur le principe d'optimalité de Bellman : "Toute politique optimale est composée de sous-politiques optimales". La programmation dynamique repose alors sur la décomposition du problème initial à un ensemble de sous-problèmes qui lui sont équivalents. Dans notre cas, la recherche de la sous-chaîne la plus longue entre deux chaînes de caractères a été ramenée à une simple comparaison des différents couples de lettres constituant les chaînes de départ.

Nous définissons la matrice L, où $L_{i,j}$ correspond à la longueur maximale de la chaîne de caractères commune prenant fin à Lemme [i] et Entrée[j].

Ensuite,

si Lemme[i] = Entrée[j]
$L_{i,j} = 1 + L_{i-1,j-1}$
alors //si Lemme[i] != Entrée[j]
$L_{i,j} = 0$

Nous illustrons cette technique par le traitement de l'entrée lexicale :

`remonterons: monter,V+F+3+s`

Forme fléchie

Lemme		*r*	*e*	*m*	*o*	*n*	*t*	*e*	*r*	*o*	*n*	*s*
	m	0	0	1	0	0	0	0	0	0	0	0
	o	0	0	0	2	0	0	0	0	1	0	0
	n	0	0	0	0	3	0	0	0	0	2	0
	t	0	0	0	0	0	4	0	0	0	0	0
	e	0	1	0	0	0	0	5	0	0	0	0
	r	1	0	0	0	0	0	0	6	0	0	0

Ensuite, LCS (Lemme, Entrée) = $\max_{1 \le i \le m, 1 \le j \le n} L_{i,j}$

En effet, la valeur LCS (Lemme, Entrée) retenue comme étant la valeur maximale dans cette matrice inclut les informations suivantes :

- la taille de la sous-chaîne commune la plus longue, soit 6 ;
- l'indice de départ de cette sous-chaîne au niveau de l'entrée lexicale, soit 2 ;
- l'indice de départ de cette sous-chaîne au niveau du lemme, soit 0.

Cet algorithme a une complexité en $O(m \cdot n)$ en terme de temps d'exécution et de mémoire. Pour réduire l'utilisation de la mémoire, nous gardons seulement la ligne courante et celle qui la précède de la table $L_{i,j}$. Ainsi, nous passons d'une utilisation mémoire de $O(m \cdot n)$ à $O(\min(m,n))$. La version optimisée de cet algorithme de recherche de la sous-chaine la plus longue est présentée ci-dessous :

Algorithme 3 : **LongestCommonSubstring : algorithme de calcul de la sous-chaine la plus longue entre le lemme et la forme fléchie**
Entrées : entrée, lemme
Sortie : la sous-chaine la plus longue entre la forme en entrée le lemme

```
string LongestCommonSubstring (string entrée,string lemme)
{
   n = taille de l'entrée
   m = taille du lemme
   TailleTableau = Minimum(n,m)
   Colonne1 = Tableau(0.. TailleTableau);
   Colonne2 = Tableau(0.. TailleTableau);
   TailleMax = 0
   LCS = "" // LCS = Longest Common Substring
   for i de 0 à n
   {
    for j de 0 à m
        {
      if entrée[i] == lemme[j]
      {
                Colonne2[k] = Colonne1[k-1] + 1
            if Colonne2[k] > TailleMax
                TailleMax = Colonne2[k]
                LCS = ""
            if Colonne2[k] == TailleMax
                LCS += entrée[i-TailleMax +1..i]
          }
          else // entrée[i] != lemme[j]
                Colonne2[k] = 0
       }
       Colonne1 = Colonne2
   }
   return LCS;
}
```

Bien que l'algorithme CompressLemma ait été, au départ, conçu pour résoudre le problème de la préfixation, il s'avère très bien adapté au problème d'infixation telle que la substitution de lettres. Par exemple, en français, lors de la conjugaison du verbe «semer» au présent de l'indicatif, à la 3ème

personne du singulier, la première voyelle «e» est remplacée par la lettre accentuée "è" pour produire la forme fléchie "sème". L'ancien algorithme de NooJ analyse cette forme comme étant:

`sème [<B3> emer]` → `semer`

La nouvelle routine de calcul de correspondance produit l'analyse suivante :

`sème [r <LW><R><S> e]` → `semer`

Cette commande morphologique « r <LW> <R> <S> e » calcule la correspondance entre la forme fléchie (sème) et son lemme (semer) en procédant par les opérations suivantes :

- r : insérer la lettre 'r' à la fin du mot : sème| + r → sèmer| ;
- <LW> : pointer au début de la forme produite : sèmer| → |sèmer ;
- <R> : déplacer le pointeur d'une position vers la droite : |sèmer → s|èmer ;
- <S> : supprimer la lettre courante, soit la lettre accentuée 'è' : s|èmer → s|mer ;
- e : insérer la lettre 'e' au niveau du pointeur de lecture : s|mer → se|mer → semer.

Bien que plus complexe à lire, la nouvelle commande unifie un grand nombre de paradigmes de conjugaison de verbes tels que : (gèle => geler), (lève => lever), (mène => mener), (pèse => peser), (sème => semer), etc.

La méthode ainsi décrite permet de traiter aussi bien des préfixations et des suffixations que des infixations. Elle permet aussi le traitement de la combinaison de celles-ci dans la même forme. Pour illustrer ce propos, nous détaillons, ci-dessous, l'analyse de l'entrée lexicale :

`relèvent,lever,V+P+3+m+s`

Après avoir vérifié que la forme fléchie « relèvent » n'est ni un préfixe, ni un suffixe du lemme « lever », nous procédons à l'application de la fonction récursive **RecCompression.** Au départ, cette fonction prend comme arguments : la forme fléchie "relèvent" et le lemme "lever". Elle commence par la appeler la fonction ***LongestCommonSubstring***("relèvent","lever") afin de déterminer la plus longue sous-chaine commune entre les deux chaines en entrée. Cette fonction repose sur une méthode de programmation dynamique améliorée, elle nécessite un $O(m \cdot n)$ en temps d'exécution et un $O(\min(m,n))$ en espace mémoire requis. En l'occurrence, comme montré ci-dessous, la plus longue sous-chaine commune determinée est « ve »

		Forme fléchie							
		r	*e*	*l*	*è*	*v*	*e*	*n*	*t*
Lemme	*l*	0	0	1	0	0	0	0	0
	e	0	0	0	0	0	0	0	0
	v	0	0	0	0	1	0	0	0
	e	0	1	0	0	0	2	0	0
	r	0	1	0	0	0	0	0	0

En outre, cette matrice inclut les informations suivantes :

- la taille de la sous-chaîne commune la plus longue = 2 ;
- l'indice de départ de cette sous-chaîne au niveau de l'entrée lexicale = 4 ;
- l'indice de départ de cette sous-chaîne au niveau du lemme = 2.

Nous pouvons en conclure que :

- La plus longue sous-chaine commune = **LCS** = « ve »
- Le contexte gauche de cette sous-chaîne dans l'entrée = **Ctxt_Av_Entrée** = « relè » ;
- Le contexte gauche de cette sous-chaîne dans le lemme = **Ctxt_Av_Lemme** = « le » ;
- Le contexte droit de cette sous-chaîne dans l'entrée = **Ctxt_Ap_Entrée** = « nt » ;
- Le contexte droit de cette sous-chaîne dans l'entrée = **Ctxt_Ap_Lemme** = « r » ;

Ensuite, la fonction **RecCompression** est réitérée, pour traiter, de façon récursive, les contextes gauches de la forme fléchie et du lemme et les contextes droits, respectifs. Dans le tableau, ci-après, nous détaillons le cheminement pour le calcul de la commande de correspondance entre la forme fléchie "relèvent" et le lemme "lever".

<table>
<tr><td colspan="7">Entrée = "relèvent" ET Lemme = "lever"</td></tr>
<tr><td colspan="4">Contextes gauches</td><td rowspan="2">Plus longue sous-chaine commune = LCS</td><td colspan="2">Contextes droits</td></tr>
<tr><td colspan="2">Entrée</td><td colspan="2">Lemme</td><td>Entrée</td><td>Lemme</td></tr>
<tr><td colspan="2">"relè"</td><td colspan="2">"le"</td><td rowspan="3">"ve"</td><td>"nt</td><td>"r"</td></tr>
<tr><td colspan="2">Contextes gauches</td><td rowspan="2">LCS</td><td colspan="2">Contextes droits</td><td colspan="2" rowspan="2">LCS</td></tr>
<tr><td>Entrée</td><td>Lemme</td><td>Entrée</td><td>Lemme</td></tr>
<tr><td>"re"</td><td>""</td><td>"l"</td><td>"è"</td><td>"e"</td><td colspan="2">""</td></tr>
<tr><td colspan="7">Tranformations :</td></tr>
<tr><td colspan="2"><S2></td><td><R></td><td colspan="2"><S>e</td><td><R2></td><td><S2>r</td></tr>
<tr><td colspan="7">Transformation finale: <LW><S2><R><S>e<R2><S2>r</td></tr>
</table>

Tableau 7 : Description de la correspondance morphologique entre « relèvent » et « lever »

La commande, ainsi produite, est constituée d'un ensemble de tranformations morphologiques, où :

- <LW> : positionnement du curseur de lecture au début de la forme fléchie : relèvent → |relèvent ;
- <S2> : suppression des deux lettres courantes (supprimer "re") : |relèvent → |lèvent ;
- <R> : déplacement du pointeur de lecture d'une position vers la droite (Right) : |lèvent → l|èvent ;
- <S>e : suppression de la lettre courante ("è") et insertion de la lettre "e" : l|èvent → l|vent → le|vent ;
- <R2> : déplacement du pointeur de lecture de deux positions vers la droite (Right) : le|vent → leve|nt ;
- <S2>r : suppression de deux lettres courantes ("nt") et insertion de la lettre "r" : leve|nt → leve| → lever| ;

Nous notons que, à la fin de ce processus, nous proposons une tranformation équivalente à celle déterminée pour une meilleure lisibilité et réutilisabilité par d'autres entrées lexicales. En l'occurrence, la commande **<LW><S2><R><S>e<R2><S2>r** est transformée en : **<B2>r<LW><S2><R><S>e**, où :

- <B2>r : suppression de deux dernières lettres ("nt"), à partir de la forme fléchie, et insertion de la lettre "r" : relèvent| → relève| → relèver| ;
- <LW> : positionnement du curseur de lecture au début de la forme: relèver| → |relèver ;
- <S2> : suppression des deux lettres courantes (supprimer "re") : |relèver → |lèver ;
- <R> : déplacement du pointeur de lecture d'une position vers la droite (Right) : |lèver → l|èver ;
- <S>e : suppression de la lettre courante ("è") et insertion de la lettre "e" : l|èver → l|ver → le|ver → lever;

Nous remarquons alors que, dans la pratique, même les dictionnaires de langues romanes bénéficient considérablement de la nouvelle méthode de compression.
Cette méthode de codage montre d'avantage de profits quand il s'agit de langues à morphologie complexe telles que les langues sémitiques (tel que l'arabe ou l'hébreu) et les langues germaniques (tel que l'allemand ou le néerlandais).

1.5.2 Evaluation et expérimentation

Le moteur morphologique de NooJ a été amélioré par l'algorithme de minimisation incrémentale décrit ci-dessus et une nouvelle méthode de codage. Ces deux améliorations se sont réunies pour mener à bien certaines mesures du rendement tel que l'espace de stockage de l'ensemble des données lexicales d'un langage donné.
Bien que cette combinaison a été utilisée avec succès pour toutes les langues traitées dans NooJ, les résultats les plus spectaculaires ont été obtenus pour :

- Le hongrois dont le dictionnaire contient plus que 130 millions d'entrées. Ces entrées ont été stockées dans un automate à 25 millions d'états.
- L'arabe qui utilise massivement les préfixes et infixes lors des transformations morphologiques : par exemple, l'entrée lexicale "كَتَبَ" (kataba - à écrire), on obtient 122 mots, y compris les suivantes:
  - ✓ "أَكْتُبُ" (*áaktubu* – j'écris) → <B><LW><S2><R>a<S><R>a<S><R>a
  - ✓ "يَكْتُبَانِ" (*yaktubaāni* – ils écrivent, dual, masculin) → <B3><LW><S2><R>a<S><R>a<S>
  - ✓ "كَتَبَا" (*katabaā* – ils ont écrit, dual, masculin) → <B>

En comparant l'ancien et le nouvel algorithme pour la construction du dictionnaire électronique de l'arabe, on obtient:

	Nb lemmes	Nb formes fléchies	Nb analyses différentes – codage initial	Nb analyses différentes – nouveau codage	Taux de compression
Verbes	10 500	2 734 122	2 311 973	122 821	94.4%
Noms	18 247	280 267	18 869	10 489	44.4%
Adjectifs	4 651	163 715	16 683	2 251	86.5%

Tableau 8 : Evaluation du nouvel algorithme de compression dynamique

Ce taux de compression assez remarquable au niveau des dictionnaires de l'arabe est principalement dû à la complexité de sa morphologie flexionnelle et dérivationnelle. Celle-ci se base essentiellement sur un ensemble d'opérations morphologiques prenant la forme d'infixations pour passer d'un lemme aux formes fléchies qui s'y rattachent. Des transformations inverses, assez complexes notamment pour les verbes, sont, alors, prévues pour calculer la commande de correspondance entre chaque forme fléchie et son lemme. Par ailleurs, nous signalons que les taux de compression des noms et adjectifs sont moins importants que celui affiché pour les verbes pour deux raisons : d'un côté, ceci est dû à l'abondance de préfixes et suffixes ainsi que leurs combinaisons pour passer des lemmes nominaux et adjectivaux aux formes fléchies qui s'y rattachent ; d'un autre côté, une grande partie des infixations qu'ils mettent en jeu sont représentés par des inversions de voyelles à l'intérieur des formes. La complexité de la langue arabe aussi bien au niveau de sa morphologie flexionnelle et dérivationnelle qu'au niveau des phénomènes morpho-syntaxiques spécifiques qu'elle présente sera détaillé dans la suite de ce mémoire.

Le nouveau compilateur de dictionnaires dans NooJ incluant deux nouveaux algorithmes – une minimisation incrémentale et une compression dynamique – a donné naissance à des dictionnaires électroniques beaucoup plus compacts sans avoir d'influence majeure sur le temps de compilation.

Par exemple, la totalité du dictionnaire arabe (3 244 386 entrées) a été compilé en 4 minutes sur un PC Pentium4, 3,2 Ghz, 2 Go de RAM. Ceci est plus que suffisant pour une utilisation typique de NooJ étant donné que les dictionnaires ne sont, habituellement, recompilés qu'une fois toutes les quelques semaines.

1.6 Conclusion

Ce premier chapitre concerne l'étude de l'utilisation de la technologie à états finis pour le traitement automatiques des langues naturelles (analyse automatique, traitement de corpus, détection / correction orthographique, reconnaissance de la parole, etc.). La première partie expose les principales motivations pour l'utilisation d'une telle technologie, les définitions de base de l'outillage formel utilisé et un état de l'art des travaux algorithmiques déjà menés. Simultanément, nous faisons le point sur les méthodes actuelles de construction, de minimisation et de compression des dictionnaires au sein de notre plateforme de travail NooJ. Nous montrons, expériences à l'appui, que NooJ dans sa première version ainsi que son prédécesseur INTEX utilisaient des méthodes classiques qui sont peu efficaces pour comprimer de grands dictionnaires surtout pour des langues à morphologie complexe. La dernière partie décrit le nouveau compilateur de dictionnaires pour NooJ à partir de sa version 2.0 : une minimisation incrémentale (version 2.0 et 2.1) et une compression dynamique (version 2.2) ; cette dernière est une nouvelle approche de compression de dictionnaire de formes qui procède par recherche récursive de la plus longue sous-chaîne commune. Enfin, nous donnons une évaluation des performances de ce nouveau compilateur.

Chapitre 2

Problèmes spécifiques à la morpho-syntaxe arabe

Afin d'atteindre les résultats escomptés, nous avons, tout au long de nos recherches, eu recours à certaines notions fondamentales qui touchent non seulement le cadre général du travail, mais aussi sa réalisation. Dans cette section, nous commençons par la définition de quelques concepts qui s'avèrent nécessaires pour une bonne compréhension du contexte théorique et pratique dans lequel a été réalisé le travail. Cette section a été inspirée des trois références [Achour, 1998], [Neyreneuf et Hakkâk, 1996] et [Kouloughli, 1994].

2.1 Alphabet arabe

La langue arabe est une langue *sémitique* qui s'écrit de droite à gauche et dont l'alphabet est un *abjad*[7]. L'écriture arabe note essentiellement les consonnes, elle note également les voyelles longues "ا" , "ي" et "و". Les deux dernières sont des réalisations contextuelles des glides "وُ" et "يِ". L'écriture arabe comporte également des voyelles qui ne sont pas essentielles à l'écriture ainsi qu'un certain nombre de signes annexes dont l'emploi est facultatif hormis pour le coran servant à noter les trois voyelles brèves ("ـَ" (*a*), "ـُ" (*u*) et "ـِ" (*i*)). Il existe, de plus, une série d'autres diacritiques de syllabation dont les plus courants sont l'indication de l'absence de voyelle "ـْ"*(sukūn)* et la gémination des consonnes "ـّ" *(šadda)*.

Notons aussi que si un mot arabe est indéfini (sans article ni complément du nom), il prend (sauf exceptions) les désinences "ـً" (*an*), "ـٌ" (*un*) ou "ـٍ" (*in*), nommées *nounation* ou *tanwīn*. Celles-ci sont notées par des diacritiques spéciaux marqués par le redoublement du signe de la voyelle qui précède le suffixe « n » attendu en fin de mot.

En outre, il nous importe de signaler que les notions de lettre capitale et lettre minuscule n'existent pas, l'écriture arabe est dite *monocamérale*. En addition, l'arabe est une langue *semi cursive* ; la plupart des lettres s'attachent entre elles, leurs graphies diffèrent selon qu'elles soient précédées et/ou suivies d'autres lettres ou qu'elles soient isolées. Seulement six d'entre-elles ne s'attachent jamais à la lettre suivante, elles sont notées par un astérisque dans le tableau de translittération fourni en Annexe A.

2.2 Composition du lexique arabe : les formes de bases

La grammaire traditionnelle arabe ne connaît que trois sous-ensembles : noms, verbes et particules. Ce classement montre, rapidement, ses limites quand il s'agit d'un traitement informatisé de la langue. En effet, la classe des particules a été étendue pour inclure des morphèmes grammaticaux qui, en réalité, appartiennent à d'autres classes telles que les pronoms démonstratifs ou relatifs qui constituaient des entrées nominales particulières. Cette extension a aboutit à une re-organisation en quatre sous-ensembles du lexique arabe : les verbes, les noms, les pronoms et les mots outils [Kouloughli, 1991] et [Khoja et al., 2001].

2.2.1 Les verbes

Un verbe est une entité exprimant un sens dépendant du temps. La majorité des verbes arabes sont formés sur des radicaux de 3 consonnes tel est le cas du verbe "كَتَبَ" (*kataba* – écrire) et éventuellement 4 consonnes tel est le cas du verbe "دَحْرَجَ" (*daḥrağa* – glisser, faire glisser). Ces racines peuvent donner naissance à plusieurs schèmes ou patrons à la suite d'une ou plusieurs transformations morphologiques (par exemple, le redoublement d'une consonne, l'allongement d'une voyelle, l'adjonction d'un morphème, etc.), il s'agit dans ce cas de racines à schème augmenté.

[7] Un abjad, appelé aussi alphabet consonantique, désigne un alphabet ne notant que des consonnes (ou notant principalement les consonnes), comme en arabe ou en hébreu.

Plusieurs études linguistiques ont été menées sur le système verbal en arabe, voir [Larcher, 2003]. Dans cette section, il nous importe d'introduire un classement des verbes selon leurs radicaux :

- **Les verbes à racine simple :**
  Les verbes à racine simple ont une base à trois consonnes appelées consonnes radicales. Ces verbes sont associés au schème verbal "فَعَلَ" (*fa'ala*). Lorsqu'aucune des consonnes radicales du verbe n'est une voyelle longue, il est dit *sain*. Ces radicaux peuvent comporter de causes de transformation ou défectuosité (عِلّة), nous citons :
  - ✓ La présence d'un glide "أ" (*á – hamzä*), "ي" (*y - yâ'*) ou "و" (*w – wâw*) parmi les consonnes du radical. Selon la position de ce glide, nous distinguons différents types de verbes :
    - Si l'une des consonnes radicales est "أ" (*á – hamzä*), indépendamment de sa position → *verbe hamzé* « مَهْموز» (*mahmuwz*) ;
    - la 1ère consonne radicale est un "و" (*w*) ou "ي" (*y*) → *verbe assimilé* « مِثّال» (*miṯal*) ;
    - la 2ème consonne radicale est un "و" (*w*) ou "ي" (*y*) → *verbe creux* « أجْوَف» (*ağwaf*) ;
    - la 3ème consonne radicale est un "و" (*w*) ou "ي" (*y*) → *verbe défectueux* « نَاقِص» (*naāqiṣ*) ;
  - ✓ La présence de deux consonnes identiques en deuxième et troisième position du radical → *verbe redoublé* « مُضَاعَف» (*muḏaā'af*).

  Notons que, les causes de défectuosité peuvent être combinées pour avoir des transformations doubles ou triples au sein d'une même racine.

- **Les verbes à racine augmentée :**
  Aux schèmes simples, décrits ci-dessus, vient s'ajouter une dizaine de schèmes verbaux augmentés. Ces schèmes sont formés à partir de racines simples par un ensemble d'opérations morphologiques pour donner un sens particulier aux verbes résultats, nous citons :

Schème verbal augmenté	Opérations morphologiques associées
« فَعَّلَ» (*fa''ala*)	redoublement de la deuxième consonne radicale
« فَاعَلَ» (*faā'ala*)	allongement de la première consonne radicale par l'ajout d'un "ا" (*ā – álif)*
« أفْعَلَ» (*áfa'ala*)	adjonction d'une "أ" (*á – hamzä*) au début de la racine
« تَفَعَّلَ» (*tafa''ala*)	adjonction d'une "ت" (*t*) au début de la racine + redoublement de la deuxième consonne radicale
« تَفَاعَلَ» (*tafaā'ala*)	adjonction d'une "ت" (*t*) au début de la racine + allongement de la première consonne radicale par l'ajout d'un "ا" (*ā – álif)*
« اِفْتَعَلَ» (*ìfta'ala*)	adjonction d'un "ا" (*ā – álif*) au début de la racine + insertion d'un morphème mono-consonantique "ت" (*t*) à la suite de la première consonne
« اِنْفَعَلَ» (*ìnfa'ala*)	adjonction d'un morphème bi-consonantique "اِنْ" (*ìn*) au début de la racine
« اِسْتَفْعَلَ» (*ìstaf'ala*)	adjonction d'un morphème tri-consonantique "اِسْتَ" (*ìsta*) au début de la racine

Tableau 9 : Principaux schèmes verbaux augmentés

2.2.2 Les noms

Le système morphologique des noms arabes distingue trois sous-catégories :

- **Les noms primitifs :** ce sont des noms qui ne peuvent pas être rattachés à une racine verbale. Ils paraissent bien constituer le glossaire fondamental de la langue concrète. Exemple : رَأس (*raás* – tête), كُرْسِيّ (*kursiyy* – siège), كَبْش (kabš – bélier) etc. Dans cette catégorie, nous incluons aussi les noms à racine bilitère tels que : دم (*dam* - sang), فم (*fam* - bouche), أب (áab – père), أخ (*áḫ* – frère), etc.
- **Les noms dérivés ou déverbaux :** ce sont les noms qui peuvent être dérivés à partir d'une racine verbale. Le nombre et la nature de ces formes varient selon le statut du verbe auquel ils se rattachent. En tant que noms, ils peuvent recevoir les marques du cas, du genre et de l'indétermination. La description des différents types de déverbaux sera détaillée dans la section 2.4 qui concerne la description de la morphologie dérivationnelle de l'arabe.

- **Les nombres :** cette catégorie de noms est constituée par les numéraux simples représentant les unités : de "صفر" (*ṣifr*- zéro, 0) à "تسعة" (*tis'at* – neuf, 9) ; les dizaines : "عشرة" (*'ašarat* – dix, 10), "عشرون" (*'išruwn* – vingt, 20), …et "تسعون" (*tis'uwn* – quatre-vingt-dix, 90) ; les centaines, etc. ainsi que les numéraux composés tels que les cardinaux de "أحَـد عَشَر" (*áaḥada 'ašara* - onze, 11) à "تسعة عَشَر" (*tis'at 'ašara* - dix-neuf,19).

Dans leur décomposition, les grammairiens arabes ont assimilé les adjectifs à des noms vu qu'ils y prennent presque toutes les formes morphologiques ; ils peuvent, par exemple, être définis ou indéfinis et se fléchir suivant le cas, le nombre et le genre.

2.2.3 Les pronoms

La classe des pronoms a été introduite en tant qu'extension du système de décomposition traditionnel du lexique de la langue arabe. Cette classe regroupe quelques formes qui étaient considérées comme des noms particuliers. Cet ensemble échappe à toute règle de dérivation. Les pronoms forment une liste fermée de mots. Dans cette liste, nous distinguons :

- **Les pronoms démonstratifs** "أسماء إشارة" (*ásmaā' íšaārat*) : ils représentent une sous-catégorie de pronom exprimant une idée de démonstration. Ils permettent d'indiquer que l'objet représenté se trouve, soit dans le texte, soit dans l'espace ou le temps, défini par la situation d'énonciation. Ils existent deux sous-ensembles : les démonstratifs de proximité (par exemple : "هَذا" (*haḏaā* – celui-ci), "هَؤُلاء" (*haŵulaā'* – ceux-ci) et les démonstratifs d'éloignement (par exemple : "ذلِكَ" (*ḏalika* – celui-là), "أولائِكَ" (*ùuwlaāýika* – ceux-là), etc.). Les démonstratifs ne sont déclinables qu'au duel.

- **Les pronoms relatifs** "أسماء موصولة" (*ásmaā' mawsuwlat*) : ils se rapportent au nom ou au pronom personnel qui les précède et que nous désignons par antécédent. Les relatifs s'accordent avec l'antécédent et ne sont déclinables qu'au duel (comme les démonstratifs). Parmi les pronoms relatifs, nous citons : "الَّذي " (*al-laḏiy* - celui), "اللتَيْن " (al-latayni – celles, féminin, duel), "اللذينَ " (al- *laḏayni* – ceux, masculin, pluriel), etc.

- **Les pronoms personnels** "ضمائر منفصلة" (*ḍamaāýir munfasilat*) : ils servent à désigner les trois types de personnes grammaticales :
  - ✓ la première personne, c'est-à-dire, l'énonciateur ou locuteur : "أنَـا" (*ánaā* - je) ou "نَحْنُ" (*naḥnu* – nous) ;
  - ✓ la deuxième personne, c'est-à-dire, le destinataire ou interlocuteur : "أنْـتَ" (*áanta* – tu, masculin), "أنْـتِ" (*áanti* – tu, féminin), "أنـتُمَا" (*áantumaā* – vous, duel), "أنْتُمْ" (*áantum* – vous, masculin, pluriel), "أنْتُنَّ" (*áantunna* – vous, féminin, pluriel);
  - ✓ la troisième personne, c'est-à-dire, la personne absente, celle dont on parle : "هُوَ" (*huwa* – il), "هِيَ" (*hiya* – elle), "هُمَـا" (*humaā* – ils, duel), "هُمْ" (*hum* – ils), "هُـنَّ" (*hunna* – elles) ;

2.2.4 Les mots outils

Les mots outils sont des entités qui servent à situer des faits ou des objets par rapport au temps ou au lieu. Ils jouent également un rôle clé dans la cohérence et l'enchaînement d'un texte. Par exemple, nous avons des particules qui désignent un temps : بعد (*ba'da* – après)، قبل (*qabla* – avant)، منذ (*mundu* – depuis) ou un lieu tel que حيث (*ḥaytu* – où),…. Selon leur sémantique et leur fonction dans la phrase, ils peuvent jouer un rôle important dans l'interprétation d'une phrase en exprimant une introduction, explication, conséquence, etc. [Kadri et Benyamina, 1992]. Les mots outils incluent différentes catégories, nous citons :

- Les prépositions : par exemple, "فِي" (*fiy* – dans) ou "عَلَى" (*'alaq* – sur);
- Les conjonctions de coordination : par exemple, "ثُمَّ" (*tumma* – ensuite, puis) ;
- Les adverbes : par exemple, "أَبَدًا" (*abadã* – jamais) ou "بشكلٍ عَادِيٍّ" (*bišaklĩ 'aādiyyĩ* – normalement, de façon normale) ;
- Les quantificateurs : par exemple, "كُلّ" (*kulla* – tout) ou "بَعْض" (*ba'ḍa* – un peu) ;
- Etc.

Les mots outils se répartissent en sous-groupes : ceux qui sont *variables* (quantificateurs) et ceux qui sont *invariables* (adverbes, prépositions, etc.).

2.3 Morphologie flexionnelle

L'arabe est une langue flexionnelle. Elle emploie, pour la conjugaison du verbe et la déclinaison du nom, des indices d'aspect, de mode, de temps, de personne, de genre, de nombre et de cas, qui sont en général des suffixes et préfixes [Blachère et Gaudefroy, 1975]. Généralement, ces marques flexionnelles permettent de distinguer :

- le mode des verbes : par exemple, pour le verbe "ذَهَبَ" (*dahaba* – aller), les formes à l'accompli sont repérables à l'aide de leurs suffixes tel que "ذَهَبْتُ" (*dahabatu* – je suis allé) ou de leurs préfixations tel que "أَذهَبُ" (*áadhabu* – je vais) ;
- la fonction des noms à l'aide des suffixations tels que "رَجُلانِ" (*rağulaáni* – deux hommes au *nominatif*) ou "رَجُلَيْنِ" (*rağulayni* – deux hommes à *l'accusatif* ou *génitif*).

2.3.1 Flexion des verbes

La conjugaison des verbes décrit la variation de leurs formes en fonction des circonstances. Généralement, la conjugaison regroupe un certain nombre de valeurs dont :

- **La valeur aspectuelle :** L'aspect est un trait grammatical associé, le plus souvent, au verbe pour indiquer la façon dont le procès ou l'état exprimé par le verbe est envisagé du point de vue de son développement (commencement, déroulement, achèvement, évolution globale, etc.), indépendamment du moment où l'on parle ;
- **La valeur modale :** Le mode dénote la manière dont l'action exprimée par le verbe est conçue et présentée. L'action peut être mise en doute, affirmée comme réelle ou éventuelle. Ils se combinent à la sémantique des verbes et par là créent les aspects ;
- **La valeur temporelle** : Le temps est un trait grammatical permettant de situer un fait (qui peut être un état ou une action) dans l'axe du temps de l'énonciation par rapport à trois repères : passé, présent et le futur. Les indications temporelles sont souvent accompagnées d'indications aspectuelles qui lui sont plus ou moins liées.

Ces trois principales valeurs sont étroitement liées [Chairet, 1996]; elles permettent de décrire deux formes fondamentales du verbe :

- **L'accompli** "الماضي" (*al-maáḍi*) : il indique que le déroulement de l'action exprimée par le verbe est achevé, ce qui implique le passé. Il se caractérise par une suffixation des marques de la personne, du genre, du nombre et du mode à la racine verbale. Par exemple, pour le pluriel féminin du verbe "كَتَبَ" (*kataba* – écrire), nous ajoutons le suffixe "نَ ْ" pour avoir la forme "كَتَبْنَ"

(*katabna* - elles ont écrit) et pour le pluriel masculin, nous ajoutons le suffixe "ُوا" pour avoir la forme "كَتَبُوا" (*katabuwá*, *ils ont écrit*) ;

- **L'inaccompli** "المضارع" (*al-muḍaára'*) : il signale un déroulement inachevé, ce qui peut impliquer le présent. Il se caractérise par une préfixation de ses éléments ainsi qu'une ou plusieurs infixations sous forme de duplication de lettres ou de substitution de voyelles. Par exemple, pour le verbe "مَدَّ" (*madda* – tendre), nous pouvons obtenir "أَمُدُّ" (*áamuddu* – je tends) ou "يَمْدُدْنَ" (*yamdudna* – elles tendent).L'inaccompli inclut deux types de flexions modales :
  - ✓ L'inaccompli indicatif de mode réel où le locuteur énonce le caractère réel (rélisé, devant être réalisé, en cours de réalisation, etc.) de l'action ou l'état exprimé par le verbe ;
  - ✓ L'inaccompli subjonctif et apocopé de mode potentiel où le locuteur se contente d'énoncer la nature possible ou virtuelle de l'action ou l'état exprimé par le verbe.

Dans la littérature, on convient d'ajouter un paradigme supplémentaire qui est :

- **L'impératif** : il exprime l'ordre, le commandement, la défense ou l'exhortation et dont les éléments n'existent qu'à la 2ème personne au singulier, féminin duel et pluriel ;

Les formes ainsi obtenues peuvent combiner des valeurs aspectuelles, modales et temporelles bien que, dans l'usage moderne, l'aspect temporel semble être plus saillant.

A l'exception des verbes sains dont la conjugaison est régulière et suit des règles flexionnelles bien définies, tous les autres types de verbes nécessitent un traitement particulier selon le type de défectuosité, voir section 4.3.1. [Larcher, 2003].

2.3.2 Flexion des noms

En arabe, la déclinaison des noms comporte trois cas : "مَرْفُوع" (*marfou'* - nominatif), "منْصُوب" (*manṣuwb* – accusatif) et "مجْرور" (*mağruwr* – génitif). A l'exception de certains cas particuliers, les noms sont "معرّبة" (*mu'arrabat* - déclinables) et se mettent à l'un de ces trois cas suivant leur fonction dans la phrase. Sur le plan de la graphie, le cas ne correspond qu'à un élément graphique adjoint à la fin des formes nominales.

Le système nominal de l'arabe admet différents systèmes de déclinaison suivant la nature de la forme (simple, diptotes, etc.) et le nombre de celle-ci (singulier, duel ou pluriel). Nous pouvons distinguer :

- **Déclinaison du nom au singulier :**
  - ✓ **Déclinaison de base à trois cas :**
    C'est le cas le plus fréquent, il prend la voyelle "ضَمَّة – ُ" (*ḍamm*at – u) comme une marque du nominatif, la "فَتْحَة – َ" (*fatḥ*at – a) à l'accusatif et la "كَسْرَة – ِ" (*kasr*at – i) au génitif. Quand le nom est indéfini, le *tanwîn* apparaît marqué respectivement par les trois signes diacritiques : "ٌ" (*ũ* – un), "ً" (*ã* - an) et "ٍ" (*ĩ* – in). A l'accusatif indéfini, excepté le cas des noms qui se terminent par " ة " (*at*) ou par " اء " (*ā'*), un álif « ا » (*ā*) vient renforcer le tanwîn "ً" (an) : par exemple, à l'accusatif indéfini, le nom "كِتَاب" (*kitaāb* – livre) produit "كِتَابًا" (*kitaābãā* – livre à l'accusatif indéfini) et le nom "جَزِيرَة" (*ğaziyrat* – île) produit "جَزِيرَةً" (*ğaziyratã* – île à l'accusatif indéfini).
  - ✓ **Déclinaison des diptotes :**
    Les diptotes sont les noms qui, indéfinis grammaticalement, n'acceptent pas de *tanwîn* et prennent la même marque à l'accusatif et le génitif, soit la "فَتْحَة – َ" (*fatḥ*at – a). Par contre, quand ils sont définis, ils suivent la déclinaison de base à trois cas. C'est le cas des noms féminins qui se terminent par " اء " (*ā'*) tel que "صحْرَاء" (*saḥraā'* – désert), les adjectifs masculins de couleurs ayant pour schème "أَفْعل" (*af'al*) tel que "أَحْمَر" (*áḥmar* – rouge) et ceux qui sont féminins de schème "فَعْلاء" (*fa'laā'*) tel que "بَيْضَاء" (*bayḍaā'* – blanche)

✓ **Déclinaison des cinq noms :**
Les cinq noms sont :
- les 3 noms : "أبو"(*áabuw* - père), "أخو" (*áḫuw* - frère) et "حمو" (*ḥamuw* - beau-père) ;
- une variante de "فَم" (*fam* – bouche) : "فو", "فَا" et "فِي" ;
- le nom "ذو" (*ḏuw* – possesseur).

Ce sont des noms bilitères qui prolongent leur voyelle finale quand ils sont définis par un complément.

✓ **Déclinaison de déverbaux de racines défectueuses :**
Certains participes actifs et noms verbaux des verbes à racine défectueuse tels que le participe actif "مَاضٍ" (*maāḏī* – passé) et le nom verbal "تَخَلٍّ" (*taḫallī* - abondon) ne prennent la marque du cas qu'à l'accusatif: le "ي" (dernière lettre de la racine - *y*) est remplacé par le tanwîn (in) aux nominatif et génitif indéfinis. Quant aux participes passifs qui se terminent par "ى" ou "ا" tel que "مُعْطًى" (*muʿṭāq* – donné), ils perdent leur flexion casuelle. Un *tanwîn* différencie le nom indéfini du nom défini. A ce niveau, il nous importe de signaler que l'usage de cette règle de déclinaison est abondonné. En effet, dans les textes courants la forme du nom verbal "قَاضٍ" (*qaāḍī* – avocat) est, généralement, altérée en "قَاضِي" (*qaāḍiy* – avocat) par adjonction du glide "ي" (y - *yâ'*) à la fin de la forme initiale.

- **Déclinaison du nom au duel :**
Il existe en arabe "المُثَنَّى" (*al-muṯannaq* – le duel) pour désigner deux choses ou deux personnes. Il prend la place entre le singulier (pour désigner une chose ou une personne) et le pluriel (à partir de trois choses ou trois personnes).
Il s'agit d'une déclinaison avec deux alternatives où la marque du nominatif est le "ا", et celle de l'accusatif et le génitif est le "ي". Pour former le duel d'un nom indéfini ou défini par l'article, nous lui suffixons : "ان" (*āāni*) au nominatif et "يْن " (*ayni*) à l'accusatif et le génitif. Par exemple, la forme duelle du nom "سَيَّارَة" (*sayyaārat* – une voiture) prend la forme "سَيَّارَتَانِ" (*sayyaāratāāni* – deux voitures, au nominatif) ou "سَيَّارَتَيْنِ" (*sayyaāratayni* – deux voitures, accusatif et génitif)
Dans certains cas, notamment pour les mots dont la racine est défectueuse ou qui se terminent par un "ى" (*q*), un "و" (*q*) ou une *hamzä* (ʻ), la terminaison du nom se transforme devant le suffixe du duel. En l'occurrence, la forme "مَقْهَى" (*maqhaq* – un café) a pour forme duelle "مَقْهَيَانِ" (*maqhayaāni* – deux cafés)

- **Déclinaison du nom au pluriel :**
Il existe deux grandes catégories de pluriel en arabe :

✓ **Les pluriels externes ou réguliers :**
Les pluriels externes sont formés par l'ajout d'un suffixe au singulier sans changement de la structure du mot. Nous distinguons :
- **Le pluriel externe masculin** : Pour le pluriel masculin nous rajoutons les deux lettres "ين" (*iyna*) ou "ون" (*uwna*) dépendamment de la position du mot dans la phrase (sujet ou complément d'objet), exemple : "مسلم" (*muslim* - musulman) devient "مسلمون" (*muslimuwna* – musulmans, au nominatif) ou "مسلمين" (*muslimiyna* – musulmans, accusatif ou génitif) ;
- **Le pluriel externe féminin** : De la même manière, nous rajoutons pour le pluriel féminin le morphème " ات " (*āt*), exemple "سَيَّارَة" (*sayyaārat* – une voiture) devient "سَيَّارَات" (*sayyaāraāt* – des voitures).

✓ **Les pluriels internes ou brisés:**
Les pluriels internes sont désignés par pluriels brisés à cause des modifications et infixations qu'ils nécessitent par rapport à la forme du singulier, à la différence de ce qui se passe avec les

pluriels réguliers (masculin et féminin). Les formes du pluriel brisé sont nombreuses et généralement imprévisibles ; elles suivent une diversité de règles complexes et dépendent du nom ; par exemple : le nom "كَاتِب" (*kaātib* – un écrivain) se transforme pour donner les deux formes plurielles "كُتَّاب" (*kutteb* – écrivains) ou "كَتَبَة" (*katabat* – écrivains).
Notons aussi que les grammairiens arabes ont formulé des distinctions entre pluriels de petit nombre et pluriels collectifs ; par exemple : le nom " شَهْر " (*šahr* - mois) admet deux formes plurielles : " أَشْهَر " (á*šhur* - moins de 12 mois) et " شُهُور " (*šuhuwr* - au-delà) ;

Seuls les pluriels externes suivent des déclinaisons propres. Les pluriels internes se rattachent aux déclinaisons du singulier (déclinaisons de base à trois cas et diptotes).

2.3.3 Flexion des mots outils

Lorsqu'il s'agit de la flexion des particules, nous en distinguons deux catégories :

- **Les mots outils non déclinables ou invariables**: leurs formes sont constantes et n'acceptent aucune déclinaison ; par exemple : "عَلَى" (*ʿalaą* – sur), "مُنْذ" (*mun<u>d</u>u* – depuis), etc.
- **Les mots outils déclinables ou variables** : ils suivent le système de déclinaison à trois cas selon leurs fonctions dans la phrase. Par exemple, le quantificateur "كُلّ" (*kull* – tout) peut accepter les trois voyelles casuelles finales pour désigner le nominatif, accusatif ou génitif selon sa fonction dans la phrase.

2.4 Morphologie dérivationnelle

Tout verbe a dans son sillage des formes déverbales qui lui sont associées et avec lesquelles il entretient des relations morphologiques, syntaxiques et sémantiques stables. Le nombre et la nature de ces formes varient selon le statut du verbe.

Dans le cas général, celui des verbes transitifs, les plus importants de ces déverbaux sont :

- **le nom verbal :** est un nom abstrait formé sur la même racine que le verbe auquel il est associé et exprime le même contenu sémantique que lui ; toutefois, il n'implique aucune notion de temps, d'aspect, de modalité, de personne, ni même de voix. Sémantiquement, il exprime une action, un état ou un processus selon le sens du verbe auquel il est associé. Tout verbe, quelque soit son type, a un nom verbal et il arrive qu'il en ait plus d'un. Les verbes augmentés en ont, généralement, un seul. Par contre, ce n'est pas le cas pour les verbes simples qui peuvent avoir jusqu'à 5 noms verbaux. Par exemple, le verbe "وَدَّ" (*wadda* – aimer) admet cinq noms verbaux différents qui expriment une « affection » avec une certaine nuance sémantique : "وَدٌّ" (*waddu*), "وُدٌّ" (*wuddu*), "وِدَادٌ" (*widaādu*), "وِدَادَةٌ" (*widaādat*) et "مَوَدَّة" (*mawaddat*);
- **le participe actif :** est un nom associé à tout verbe d'action (transitif ou intransitif) et qui désigne l'agent du verbe c'est-à-dire celui qui fait l'action, d'où la désignation nom d'agent donnée à ce déverbal. Par exemple, les verbes à racine simple tel que "ضَرَبَ" (ḍ*araba* – frapper) suivent le schème "فَاعل" (*faāʿil*) pour produire le participe actif "ضَارِب" (ḍ*aārib* – celui qui frappe) ;
- **le participe passif :** est un nom associé à tout verbe d'action transitif. Il désigne le patient qui subit l'action ou le résultat de cette action, d'où la désignation de nom de patient donnée à ce déverbal. Par exemple, nous reprenons le cas du verbe à racine simple "ضَرَبَ" (ḍ*araba* – frapper), il suit le schème "مَفعول" (*maf'uwl*) pour produire le participe passif "مَضْرُوب" (*maḍruwb* – frappé)

Ces trois déverbaux sont ceux qui existent pour le plus grand nombre de verbes. Leurs formations obéissent, pour un type donné de verbe, à des règles extrêmement générales. Habituellement, nous assimilons le participe actif au participe présent français et le participe passif au participe passé. Cette assimilation n'occulte pas les propriétés spécifiques que ces déverbaux ont

en arabe. Pour tous les verbes, simples et augmentés, les participes se forment sur des schèmes stables ; ils ont, donc, un comportement morphologique d'une grande régularité. En tant que noms, les participes peuvent recevoir toutes les marques morphologiques de cette classe : genre, déclinaison, nombre et détermination.

Aux formes dérivées citées ci-dessus, s'ajoutent d'autres déverbaux dont le rang d'utilisation est moins important :

- **le nom de lieu (ou de temps) :** Le nom de lieu est un déverbal supposé désigner le lieu où se produit le procès exprimé par le verbe comme le nom "مَدْرَسَة" (*madrasat* – école, lieu où on étudie) qui peut être généré à partir du verbe "دَرَسَ" (*darasa* – étudier). Lorsque le sémantisme du verbe se prête plutôt à une interprétation temporelle, ce nom est dit nom de temps tel est le cas pour "مَغْرِب" (*maġrib* – moment du coucher du soleil) rattaché au verbe "غَرَبَ" (*ġaraba* – se coucher) ;
- **le nom d'instrument :** est un nom qui désigne l'instrument dont on se sert pour exécuter l'action exprimée par le verbe. Par exemple, le nom "مُوَزِّع" (*muwazzi'* – distributeur) peut être rattaché au verbe "وَزَّعَ" (*wazza'a* – distribuer) ;
- **le nom d'une fois :** est un nom qui désigne une occurrence unique de l'action exprimée par le verbe. Il est généralement produit à partir d'une forme au singulier du nom verbal. Par exemple, le nom "ضَرْبَة" (ḍ*arabat* – une frappe) peut être rattaché au verbe "ضَرَبَ" (ḍ*araba* – frapper) ;
- **le nom de manière :** est un nom qui est supposé servir à indiquer la manière dont l'action exprimée par le verbe se réalise. Pour les verbes augmentés, il n'est pas distinct du nom d'une fois. Nous pouvons donner l'exemple de la phrase : "جَلَسَت جِلْسَةَ الأمِيرَة" (*ğalasat ğilsata al-amirat* - elle s'assit à la manière d'une princesse).

Contrairement aux verbes, les noms primitifs échappent au système dérivationnel. Cependant, certaines règles peuvent être mises en place pour décrire les gentilés et ethnonymes qui sont des noms ou adjectifs par lesquels nous désignons des habitants d'un lieu, une nationalité, une identité nationale, etc.

2.5 Phénomènes morpho-syntaxiques

Depuis plusieurs décennies, de nombreux travaux de recherche ont été menés pour étudier les phénomènes linguistiques spécifiques à la langue arabe tels que la non vocalisation et l'agglutination. La difficulté de ces phénomènes est apparue sous forme d'une complexité de mise en place dès qu'il s'agissait d'un passage de l'état d'un prototype théorique ou maquette à celui d'un système réellement utilisable dans des applications à large échelle. Dans la section suivante, nous détaillons les phénomènes morpho-syntaxiques qui nous paraissaient les plus importants à étudier avant d'entamer le développement d'un analyseur automatique de l'arabe.

2.5.1 Problème de la voyellation

Pour décrire le problème de la voyellation, nous reprenons la définition de Joseph Dichy : « *L'écriture arabe courante ne note pas les voyelles brèves, la gémination des consonnes, les marques casuelles composées d'une voyelle brève suivie, pour les noms et les adjectifs indéterminés, d'une consonne "ن" (n - tanwîn), etc. On parle alors d'écriture 'non-voyellée'. Ces signes de voyellation, qui sont réalisés, lorsqu'ils sont notés, sous la forme des signes diacritiques placés au dessus ou au-dessous des lettres, apparaissent dans certains textes religieux (coran ou hadith) ou littéraires (poésie classique, notamment) : on dira qu'ils sont édités en graphie voyellée. On distingue en outre deux pratiques, celle de l'écriture entièrement voyellé et celle qui l'est partiellement. La voyellation partielle répond, dans les éditions soignées, à la levée de certaines ambiguïtés de première lecture […]. On notera que la voyellation partielle ne repose pas sur une*

codification appuyée sur une tradition : elle ne présente donc pas un caractère systématique. » [Dichy, 1997].

L'analyse automatique de l'arabe se heurte au problème redoutable de l'absence des voyelles (ou signes diacritiques) dans les textes. Ce problème entraîne plusieurs cas d'ambiguïté lexicale en arabe compte tenu de la nature polysémique des mots non voyellés. Le problème de la voyellation multiple, bien que beaucoup plus fréquent en arabe, peut être comparé à celui que pose l'accentuation multiple des mots français non accentués.
Pour illustrer ce propos, nous considérons un mot non accentué, en français, comme *peche*. Ce mot peut être interprété comme *pêche* (Nom féminin)*, péché* (nom masculin et participe passé du verbe pecher) et *pèche* (Verbe, Présent de l'indicatif, Voix active, 3ème personne, masculin|féminin singulier). De façon analogue pour l'arabe, le mot non voyellé « كتب» (*ktb*) peut avoir 17 voyellations différentes :

Voyellation	Translittération	Traduction	Catégorie grammaticale
كَتَبَ	*Kataba*	a écrit	Verbe, Accompli, Voix active, 3ème personne, masculin, singulier
كُتِبَ	*Kutiba*	a été écrit	Verbe, Accompli, Voix passive, 3ème personne, masculin, singulier
كَتَّبَ	*Kattaba*	a fait écrire	Verbe, Accompli, Voix active, 3ème personne, masculin, singulier
كُتِّبَ	*Kuttiba*	a été fait écrire	Verbe, Accompli, Voix active, 3ème personne, masculin, singulier
كَتِّبْ	*Kattib*	fais écrire	Verbe, Impératif, 2ème personne, masculin, singulier
كُتُبُ	*Kutubu*	des livres	Substantif, masculin, pluriel, nominatif, déterminé
كُتُبَ	*Kutuba*		Substantif, masculin, pluriel, accusatif, déterminé
كُتُبِ	*Kutubi*		Substantif, masculin, pluriel, génitif, déterminé
كُتُبٌ	*Kutubũ*		Substantif, masculin, pluriel, nominatif, indéterminé
كُتُبٍ	*Kutubĩ*		Substantif, masculin, pluriel, génitif, indéterminé
كَتْبُ	*Katbu*	un écrit	Substantif, masculin, singulier, nominatif, déterminé
كَتْبَ	*Katba*		Substantif, masculin, singulier, accusatif, déterminé
كَتْبِ	*Katbi*		Substantif, masculin, singulier, génitif, déterminé
كَتْبٌ	*Katbũ*		Substantif, masculin, singulier, nominatif, indéterminé
كَتْبٍ	*Katbĩ*		Substantif, masculin, singulier, génitif, indéterminé
كَ + تَبِّ	*ka + tabbi*	comme le tranchement	Préposition + Substantif, masculin, singulier, génitif, déterminé
كَ + تَبٍّ	*ka + tabbĩ*	comme un trnachement	Préposition + Substantif, masculin, singulier, génitif, indéterminé

Tableau 10 : Voyellations potentielles du mot simple non voyellé "كتب" (*ktb*)

Cet exemple montre que le mot simple non voyellé (ktb) possède au total : 17 voyellations potentielles au total, neuf catégories grammaticales possibles et deux découpages acceptables.
A partir de cet exemple, nous pouvons remarquer que les problèmes de l'accentuation multiple en français et de la voyellation multiple en arabe, bien qu'ils se ressemblent dans le fond, ne sont pas du tout du même ordre d'importance. L'accentuation multiple reste un phénomène rare en français [Debili et Achour, 1998] : l'étude effectuée par M. El Bèze sur l'accentuation automatique du français montre que 91,7% des mots du lexique ne sont pas ambigus et la moyenne est de 1.1 accentuations possibles par mot. Par contre, en arabe, l'étude statistique menée par F. Debili montre que seulement 19% des mots du corpus sont non ambigus et la moyenne est de six voyellations par mot [Debili, 2001]. Ainsi, compte tenu des proportions importantes des mots ambigus, nous pouvons conclure que, pour l'arabe, ce phénomène est très fréquent et nécessite de sérieuses études.

2.5.2 Structure complexe des mots

Le mot graphique[8] en arabe peut provenir d'une structuration assez complexe, auquel cas il est désigné de « mot maximal ». Cette appellation a été attribuée par D. Cohen [Cohen, 1961/1970] à un mot graphique décomposable en : proclitique(s), forme fléchie, enclitique(s). La forme fléchie, désignée de mot minimal, est le noyau lexical du mot graphique, les autres constituants étant des extensions.

Comparé au français, un mot en arabe peut correspondre à une phrase française complète. Par exemple, la forme agglutinée "أَسَتَتَذَكَّرُونَنَا" (*áasatatadakkaruwnanaā* → *áa* + *sa* + *tatadakkaruwna* + *naā*) peut être traduite en « est-ce que vous allez vous souvenir de nous ? ». Dans cette section, nous allons décrire les deux types de clitiques (les proclitiques et les enclitiques) qui peuvent être collés à un mot minimal pour produire une forme agglutinée [Bohas, Guillaume et Kouloughli, 1990].

2.5.2.1 Les proclitiques

Les proclitiques sont en inventaire fini, et se combinent entre eux pour donner les traits syntaxiques (coordonnant, déterminant …) qui peuvent accompagner le mot arabe. Ils se collent au « mot minimal » (forme fléchie) en tant que préfixes.

Dans le cas des verbes, les proclitiques dépendent exclusivement de l'aspect verbal. Dans le cas des noms et les déverbaux, le proclitique dépend du mode et du cas de déclinaison.Dans notre travail, nous n'allons pas retenir l'intégralité de ces clitiques, parce que certains cas sont vraiment rares et d'autres théoriquement possibles mais jamais utilisés dans la langue. Dans la liste des proclitiques pris en compte dans notre étude, nous distinguons trois types :

- Les proclitiques réservés aux noms et adjectifs :
  - ✓ L'article de définition "الْ" (*al* – le) ;
  - ✓ Les prépositions : بِ (*bi* – avec), لِ (*li* – pour) et كَ (*ka* – comme) ;
- Les proclitiques réservés aux verbes :
  - ✓ La particule du subjonctif " نصب " (*naṣb*): لِ (*li* – pour) ;
  - ✓ La particule du futur : "سَـ" (*sa*) ;
  - ✓ La particule de l'apocopé "جزم" (*ğazm*) : لِ (*li* – pour) ;
- Les proclitiques généraux, utilisés indépendamment de la catégorie des mots auxquels ils s'attachent :
  - ✓ Les conjonctions de coordination : فَ (*fa* - et) et وَ (*wa* - et) ;
  - ✓ L'article d'interrogation "أَ" (*á* – est-ce-que) ;
  - ✓ Le marqueur de corroboration "تأكيد" (*taákiyd*): لَ (*la*).

[8] Nous désignons par *mot graphique* toute séquence de caractères arabes délimitée par deux séparateurs (blanc ou autre marqueur de séparation, tel que la ponctuation).

La restriction de la prise en compte à la dizaine de proclitiques cités ci-dessus n'est pas sans désagrément puisque le même clitique peut jouer plusieurs rôles. En l'occurrence, le proclitique و (*wa*) utilisé majoritairement comme particule de liaison (conjonction de coordination), peut être employé en tant que particule d'accompagnement "واو المعية" (*waw al-ma'iyyat*) et, à de moindres usages, en tant que particule de serment "واو القسم" (*waw al-qasam*) (cas rare).

Les proclitiques peuvent se combiner entre eux et forment ainsi un proclitique composé. Une analyse robuste doit permettre l'identification de la catégorie syntaxique de chacune de ces composants d'une telle composition. Ce découpage est particulièrement utile, voir indispensable, pour les analyses syntaxiques. Les proclitiques sont classés en quatre catégories suivant la possibilité de leur apparition dans un proclitique composé :

1ère position	2ème position	3ème position	4ème position
L'article d'interrogation "أ" (*á* – est-ce-que)	Les conjonctions de coordination :ف (*fa* - et) et و (*wa* - et)	- Les prépositions : بِ (*bi* – avec), ل (*li* – pour) et ك (*ka* – comme); - La particule du subjonctif "نصب" (*naṣb*): ل (*li* - pour) ; - La particule du futur : "سَ" (*sa*) ; - Le marqueur de corroboration "تأكيد" (*taákiyd*): لَ (*la*) ; - La particule de l'apocopé "جزم" (*ğazm*) : ل (li).	L'article de définition "الْ" (Ál – le)

Tableau 11 : Agencements possibles des proclitiques

La fusion de l'ensemble de proclitiques est régie par deux types de contraintes :

- **Une relation d'ordre :** Chaque proclitique est incompatible, dans une relation d'ordre strict, avec un proclitique de même position [Dichy, 1990]. De même, un proclitique, qui occupe une position d'antériorité par rapport à un autre proclitique sur la classification, n'a aucune chance de le suivre dans la construction d'un mot graphique arabe.
- **Des règles de compatibilité :** Les proclitiques respectant la relation d'ordre ne sont pas forcément compatibles entre eux, pour des raisons d'ordre syntaxique et sémantique. [Dichy, 1997]. A ce propos, nous signalons que la combinaison de proclitiques appartenant à quatre positions différentes est très peu fréquente. Par exemple, une construction telle que "أَفَبِالْبَيْتِ" décomposable en "أ + فَ + بِـ + الْـ + بَيْتِ" (*áa + fa + bi + l + bayti* – et + est ce qu' + à + la + maison ?) est rarement utilisée dans les textes courants.

Outre les règles de compatibilités entre proclitiques, d'autres règles s'imposent pour vérifier la compatibilité du proclitique lui-même avec les bases voir même les enclitiques qui s'y rattachent.

2.5.2.2 Les enclitiques

Ils sont en inventaire fini, leurs emplois respectent certaines restrictions. Contrairement aux proclitiques, nous ne pouvons avoir qu'un seul enclitique à la fois dans un mot graphique.

Dans le cas d'une forme verbale, le lien entre celle-ci et l'enclitique dépend de sa propriété de transitivité. Nous distinguons principalement les enclitiques compatibles avec les verbes transitifs aux humains uniquement et les enclitiques compatibles avec les verbes transitifs aux humains et aux non humains. Les verbes intransitifs et ceux conjugués au passif ne peuvent prendre aucun enclitique.

Dans le cas d'une forme nominale, le choix de l'enclitique est, généralement, régi par la flexion casuelle. En effet, l'enclitique doit respecter une certaine harmonie vocalique avec la voyelle casuelle de la forme à laquelle il se rattache. En outre, les bases nominales qui se terminent par une voyelle double ou *tanwîn* ne peuvent être en aucun cas suivies d'enclitiques.

Contrairement aux enclitiques à la première personne tels que "نِي" (*niy* – moi / mon) ou "نَا" (*naā* – nous / notre) et ceux à la deuxième personne tels que "كَ" (*ka* – toi / ton) ou "كُمْ" (*kum* – vous / votre [masculin pluriel]) dont la forme reste invariable indépendamment des propriétés de la forme à laquelle ils se rattachent, les enclitiques à la troisième personne sont variables. En effet, ils prennent différentes vocalisations suivants les règles suivantes :

- Dans le cas des noms, seul le mode déterminé par annexion est susceptible de prendre des enclitiques. Selon la flexion casuelle de la base nominale, cette dernière prend l'un ou l'autre enclitique :
  - ✓ Si le nom est fléchie au nominatif ou accusatif, il nécessite l'utilisation des enclitiques suivants : هُ [PRON+3+m+s], هُمَا [PRON+3+m|f+d], هُمْ [PRON+3+m+Pp هُنَّ [PRON+3+f+p]
  - ✓ Si le nom est fléchie au génitif, il nécessite l'utilisation des enclitiques suivants : هِ [PRON+3+m+s], هِمَا [PRON+3+m|f+Dd هِمْ [PRON+3+m+p], هِنَّ [PRON+3+f+p].

  Dans le même contexte, nous signalons que les mots qui se terminent par une *hamzä*, une « ى » (*álif maqṣûrä*) ou une "ي" (*y*) nécessitent des transformations morphologiques avant leurs suffixations. Par exemple, la forme "مَقهى" (*maqhaq* – café, salon de thé), qui se termine par « ى » (*álif maqṣûrä*), nécessite une transformation de celle-ci en "ا" (*ā* - *álif*) avant sa suffixation pour produire la forme agglutinée "مَقْهَاهُ"(*maqhaāhu*– son café, son salon de thé).

- Quant aux verbes, l'enclitique peut varier en fonction de l'aspect et du pronom. La répartition de l'utilisation de ces enclitiques selon les aspects est la suivante :
  - ✓ Si le verbe est conjugué à la voix active, inaccompli, à la 2ème personne, féminin, singulier, il prend les pronoms qui portent la marque du nominatif tels que : هُ (*hu*), هُمَا (*humaā*), هُمْ (*hum*) et هُنَّ (*hunna*) ;
  - ✓ Si le verbe est conjugué à la voix active, inaccompli, à la 2ème personne, masculin ou féminin duel, ou 3ème personne, masculin ou féminin duel, il prend les pronoms qui portent la marque du génitif tels que : هِ (*hi*), هِمَا (*himaā*), هِمْ (*him*) et هِنَّ *(hinna*) ;
  - ✓ Si le verbe est conjugué à l'Accompli Actif, Accompli passif, Inaccompli Subjonctif Actif, Inaccompli apocopé Actif ou Impératif, à la 2ème personne, féminin singulier, il prend les pronoms portant la marque du génitif : هِ (*hi*), هِمَا (*himaā*), هِمْ (*him*) et هِنَّ *(hinna*) ;
  - ✓ Si le verbe est conjugué à l'Accompli Actif, Accompli passif, Inaccompli Subjonctif Actif, Inaccompli apocopé Actif, Impératif ou Futur, 2ème personne, masculin ou féminin duel, ou 3ème personne, masculin ou féminin duel, il prend les pronoms portant la marque du nominatif tels que : هُ (*hu*), هُمَا (*humaā*), هُمْ (*hum*) et هُنَّ (*hunna*) ;
  - ✓ Le cas échéant, si le verbe est conjugué à la voix passive, il ne prend jamais d'enclitique.

2.5.2.3 Règles d'agglutination des morphèmes

La structure complexe du mot arabe, ci-dessus décrite, découle des phénomènes de flexions et d'agglutination qui caractérisent la langue. La langue arabe est agglutinative du fait que les clitiques se collent aux substantifs, verbes, adjectifs auxquels ils se rapportent. Ces phénomènes posent de redoutables problèmes pour l'analyse automatique de l'arabe dans la mesure où ils augmentent considérablement le taux d'ambiguïté en introduisant des ambiguïtés supplémentaires au niveau de la segmentation des mots. En effet, un mot arabe peut avoir plusieurs découpages possibles en : proclitique(s), forme fléchie et enclitique. Nous prenons l'exemple du mot « أوحل » qui peut avoir trois découpages potentiels:

✓ 1er découpage :

1er proclitique	2ème proclitique	Forme fléchie
أ (*á*) – article d'interrogation	و (*wa*) – conjonction de coordination	حَلَّ : verbe à l'accompli (*ḥall* – résoudre, ouvrir) ou un substantif (*ḥall* – solution)

✓ 2ème découpage :

Proclitique	Forme fléchie
أ (*á*) – article d'interrogation	وَحَلَ : verbe à l'accompli (*wa ḥal* – se coincer) ou un substantif (*waḥl* – boue)

✓ 3ème découpage :

Forme fléchie
أوحل – verbe à l'accompli (*áawḥala* – faire coincer)

La reconnaissance du bon découpage s'avère indispensable pour qu'une analyse morpho-lexicale puisse attribuer la bonne catégorie grammaticale à un mot. De ce fait, une étude de la combinaison des morphèmes pour former des mots a été partiellement entreprise par K. Beesley [Beesley, 1998]. Ci-après, nous décrivons les principales règles d'agglutination pour les mots graphiques à base verbale ou nominale :

- **Règles d'agglutination pour les verbes** :

  Les enchaînements possibles pour les verbes arabes sont présentés dans le Tableau 12. Tous les éléments sont optionnels, sauf la forme de base ; ils peuvent être reliés entre eux dans une série d'enchaînements.

  Plusieurs restrictions grammaticales sont applicables pour gérer les enchaînements possibles d'une base verbale avec les différents clitiques (enclitiques et proclitiques). Ces restrictions peuvent être considérées comme des contraintes grammaticales ou des spécifications lexique-grammaire, qui régissent le processus d'analyse morphologique. Celles-ci peuvent être résumées comme suit:

  ✓ L'article d'interrogation أ (*á* – est-ce-que) ne peut pas être collé avec un verbe conjugué à l'Impératif ou au Subjonctif ;
  ✓ Le proclitique ل (li – pour) est considéré comme une particule du subjonctif ou de l'apocopé, elle ne peut pas s'agglutiner à une forme fléchie au présent de l'indicatif ;
  ✓ La particule du futur "سَـ" (*sa*) ne peut être adjointe qu'à un verbe conjugué à l'Inaccompli, voix active ou passive ;
  ✓ Les pronoms personnels ne se collent ni aux verbes conjugués à la voix passive, ni les verbes intransitifs ;
  ✓ Lorsque la forme verbale est conjuguée à la première personne (singulier ou pluriel) ou à la deuxième personne (singulier, duel ou pluriel), elle ne peut pas être agglutinée à un pronom personnel de la même personne.

Proclitiques			Base	Enclitique
Article d'interrogation	**Conjonction**	**Marqueurs/Particules**	**Verbe**	**Pronom personnel**
L'article d'interrogation "أ" (*á* – est-ce-que)	Les conjonctions de coordination : ف (*fa* - et) et و (*wa* - et)	La particule du subjonctif "نصب" (*naṣb*): ل (*li* - pour)	Forme fléchie	1ère personne
		Le marqueur de corroboration "تأكيد" (*taákiyd*): لَ (*la*)		2ème personne
		La particule du futur : "سَـ" (*sa*)		3ème personne

Tableau 12 : Agencement des constituants d'une forme verbale agglutinée

Comme le montre ce tableau, le nombre maximum d'enchaînements possibles pour une base verbale est égale à cinq : une forme fléchie avec quatre autres morphèmes représentant les clitiques qui y sont rattachés. Statistiquement, ces enchaînements peuvent donner jusqu'à 396 possibilités. Sur le plan pratique, certains verbes, comme علم (*'allama* - enseigner), peuvent produire jusqu'à 78 formes agglutinées. Ce nombre considérable d'agglutinations potentielles est un bon indicateur de la richesse ainsi que la complexité de la morphologie arabe.

- **Règles d'agglutination pour les noms**

Les enchaînements possibles entre les proclitiques, les bases nominales et les enclitiques sont résumés dans le Tableau 13. Etant donné la non-compatibilité entre l'article défini et les pronoms personnels dans la composition des formes nominales agglutinées, le nombre maximal d'enchaînements des noms s'élève aussi à cinq : une forme fléchie avec quatre autres morphèmes (soit quatre proclitiques ou bien 3 proclitiques et un enclitique).
Les règles d'agglutination des noms peuvent être résumées comme suit:

- ✓ L'article défini ال (*al* - le / la / les) ne peut pas co-exister avec un pronom personnel ;
- ✓ L'article défini ال (*al* - le / la / les) ne peut pas co-exister avec une flexion casuelle indéfinie, notamment le *tanwîn* ;
- ✓ Une préposition telle que بـ (*bi* – avec), كـ (*ka* – comme) ou لـ (*li* - pour) ne peut co-exister qu'avec une flexion casuelle au génitif.

Proclitiques				Base	Enclitiques
Article d'interrogation	**Conjonctions**	**Prépositions**	**Article défini**	**Nom**	**Pronom personnel**
L'article d'interrogation "أ" (*á* – est-ce-que)	Les conjonctions de coordination : ف (*fa* - et) et و (*wa* - et)	بـ (*bi* – avec), كـ (*ka* – comme) ou لـ (*li* - pour)	ال (*el* - le)	Forme fléchie	1ère personne
					2ème personne
					3ème personne

Tableau 13 : Agencement des constituants d'une forme nominale agglutinée

Statistiquement, les enchaînements incontrôlés dans le tableau précédent peuvent donner 448 formes. Réellement, un nom tel que معلم (*mu'allim* - professeur) ne peut produire que 52 formes valables.

2.5.3 Ambiguïté lexicale et syntaxique

L'ambiguïté est un problème central de l'analyse morpho-syntaxique de l'arabe. Les analyseurs se trouvent fréquemment confrontés à des situations d'ambiguïtés à tous les niveaux de l'analyse que ce soit au niveau lexical, syntaxique ou sémantique. Outre la richesse des constructions syntaxiques et leurs interprétations multiples auxquelles les analyseurs sont confrontés, cette ambiguïté est due, essentiellement, à l'ambiguïté des segmentations en unités lexicales et à l'homographie polycatégorielle [Attia, 2006].

Cependant, il convient de signaler que l'ambiguïté ne constitue pas en soi un obstacle insurmontable du point de vue informatique car, grâce à l'utilisation de la technologie à états finis (voir chapitre 1), il est possible d'explorer systématiquement toutes les lectures des mots ainsi que toutes les possibilités de combinaison de syntagmes spécifiées dans la grammaire. C'est plutôt la maîtrise de la combinatoire engendrée par l'ambiguïté qui pose des difficultés pour les analyseurs.

Par ailleurs, le problème ne réside pas dans l'analyse d'un langage ambigu en soi ; mais c'est plutôt au niveau de son traitement de façon robuste et réaliste. En effet, après une première phase de segmentation du texte en unités lexicales, il est convenu de chercher dans le lexique les interprétations correspondantes à chacune d'entre elles. A chaque interprétation, nous associons une catégorie syntaxique reconnue par la grammaire.

En outre, beaucoup de mots en arabe sont homographiques: ils ont la même forme orthographique, bien que la prononciation soit différente. De cette homographie, lorsqu'elle est majorée par d'autres phénomènes (absence de voyellation, morphologie flexionnelle et agglutinante, etc.) découle un taux d'ambiguïté assez élevé. En effet, parmi les nombreux facteurs qui ont contribué à ce problème nous mentionnons le fait que:

- Le lexique arabe contient des mots homographes qui, sans flexion préalable, peuvent avoir différentes prononciations, des significations différentes et généralement des catégories grammaticales différentes.

  ذهب (*dhb*)

ذهب	ذهب
(*dahab* – or)	(*dahaba* – il est allé)

- Lors de la flexion des verbes, des successions d'opérations morphologiques et orthographiques (comme la suppression de caractères ou l'assimilation) produisent fréquemment des formes fléchies homographes qui peuvent appartenir à deux ou plusieurs lemmes. L'exemple suivant montre qu'une forme verbale simple peut être interprétée comme appartenant à cinq lemmes.

  يعد (*y ʿd*)

يُعِد (أعاد)	يَعُد (عاد)	يَعِد (وعد)	يَعُدّ (عد)	يُعِدّ (أعد)
(*yu ʿid, áa ʿāda* – il refait)	(*ya ʿud, ʿāda* – il retourne)	(*ya ʿid, wa ʿada* – il promet)	(*ya ʿudd, ʿadda* – il compte)	(*yu ʿidd, áa ʿadda* – il prépare)

- Certains lemmes sont différents uniquement par un redoublement, au moyen de la lettre Šadda , sans que cela ne soit explicite à l'écrit. Les deux formes, ci-dessous, diffèrent uniquement au niveau du redoublement de la syllabe du milieu.

  درس (*drs*)

دَرَسَ	دَرَّسَ
(*darasa* – il a étudié)	(*darrasa* – il a enseigné)

- Beaucoup de paradigmes flexionnels se fondent sur une légère modification au niveau des voyelles brèves. Ces modifications restent sans effet en cas d'absence de voyellation. Plusieurs exemples sont à noter :
  - ✓ L'exemple, ci-dessous, montre l'ambiguïté entraînée par la conjugaison du même verbe à l'accompli, voix active et voix passive et à l'impératif

    أرسل (*ársl*)

أرسَل	أُرسِل	أرسِل
(*áarsala* – a envoyé)	(*áursila* – est envoyé)	(*áarsil* - envoie ! [impératif])

  - ✓ Les deux formes suivantes sont homographiques tandis que la première est conjuguée à la 3ème personne du féminin et que la deuxième est fléchie à la 2ème personne du masculin.

    تكتب (*tktb*)

تَكْتُبُ	تَكْتُبُ
(*taktubu* - elle écrit)	(*taktubu* - tu écris [masculin])

  - ✓ Le même type d'ambiguïté peut être retrouvé au niveau des quatre formes fléchies, ci-dessous :

    كتبت (*ktbt*)

كتبتُ	كتبتَ	كتبتِ	كتبتْ
(*katabtu* - j'ai écrit)	(*katabta* - tu as écrit [masculin])	(*katabti* - tu as écrit [féminin])	(*katabat* - elle a écrit)

  - ✓ De même, la forme duelle est toujours confondue avec la forme au pluriel régulier à l'accusatif et au génitif.

    أمريكيين (*ámrykyyn*)

أمريكيَّيْن	أمريكيِّينَ
(*áamriykiyyayni* - américains [duel])	(*áamriykiyyiyna* - américains [pluriel])

- La flexion d'un verbe peut produire une forme homographique ayant un lemme nominal.

  أسد (*ásd*)

أسُدّ	أسد
(*áasuddu* - je bloque)	(*áasad* - un lion)

- Les proclitiques peuvent accidentellement engendrer deux formes homographiques.

  علمي (*ʿlmy*)

علمي	علمي (علم + ي)
(*ʿilmiyy* - scientifique)	(*ʿilm* + *y* - mes connaissances)

2.5.4 Flexibilité de l'ordre des mots dans une phrase simple

L'ordre des mots dans une phrase simple en arabe est relativement flexible. Nous disposons, généralement, d'un libre choix du terme que nous voulons mettre en valeur, en tête de phrase ; la même phrase peut être ordonnée comme :

- Verbe + sujet + complément :

  سَافَرَ الرَّئِيس إلى بريطانيا (*saāfara al-rraýiys ílaą briyṭaāniyaā* – le président s'est rendu en Grande Bretagne ;

- Sujet + verbe + complément :

  الـرَّئـيس سَافَرَ إلى بـريطانـيا (*al-rraýiys saāfara ílaą briyṭaāniyaā* – c'est le président qui s'est rendu en Grande Bretagne) ;

- Complément + verbe + sujet :

  إلى بـريطانـيا سَافَرَ الـرَّئـيس (*ílaą briyṭaāniyaā saāfara al-rraýiys* – c'est en Grande Bretagne que le président s'est rendu).

A l'issue de cette illustration, nous signalons que l'ordre relativement libre des mots d'une phrase arabe ne vient que s'accumuler aux problèmes morpho-syntaxiques spécifiques à l'arabe et limiter considérablement les éventualités de résolution de ceux-ci. En effet, la prise en compte de toutes les règles de combinaisons possibles des constituants de la phrase provoquerait, sans doute, des ambiguïtés syntaxiques artificielles supplémentaires.

2.6 Conclusion

Compte tenu de la difficulté de la mise en œuvre d'une analyse morpho-syntaxique traitant les problèmes cités ci-dessus, il n'existe pas encore d'analyseur général de la langue arabe, c'est-à-dire dont la couverture soit suffisamment large pour satisfaire aux exigences d'applications comme la traduction automatique. Ce défi constitue notre principale motivation pour la conception et la réalisation d'un analyseur morpho-syntaxique automatique pour l'arabe ; la description de ce travail fait objet de la suite de ce mémoire.

Chapitre 3

Un état de l'art sur l'analyse morpho-lexicale de l'arabe

3.1 Stratégies de représentation et d'analyse

La langue arabe est connue pour sa richesse et sa complexité morphologiques [McCarthy, 1985 ; Azmi, 1988 ; Beesley, 1998 ; Ratcliffe, 1998 ; Ibrahim, 2002] ; sa morphologie a toujours été un défi pour les spécialistes du traitement automatique des langues naturelles et un terrain tenace d'essai pour les différentes technologies et approches d'analyse automatique.

Dans un but de développement d'un analyseur automatique, d'un côté, la première prise de décision qui s'impose concerne l'approche à adopter pour la représentation des connaissances linguistiques (lexiques et grammaires) ; de l'autre côté, le niveau de liaison de ces ressources avec l'algorithme d'analyse lui-même. Cette démarche doit être en adéquation avec l'ensemble des traitements envisagés et envisageables. Parmi les principales approches de représentation des connaissances linguistiques, nous citons:

- **Approche procédurale**

  Dans le cadre d'une approche procédurale, une partie des connaissances linguistiques est directement codée dans l'algorithme d'analyse lui-même. Ceci ne peut que compliquer la tâche du linguiste chargé de l'extension et de la maintenance des connaissances linguistiques (grammaires et lexiques). Nous assistons, ainsi, à une fusion des différents niveaux d'analyse ; un très large ensemble de règles est alors mis en place. Cette approche a été adoptée par Berrondonner qui décrit la difficulté de sa démarche comme suit :

  « *Le problème est évidemment, de définir une sorte de « méta-algorithme », ou processus stratégique, qui soit capable, en fonction des problèmes de reconnaissance que posent les structures linguistiques rencontrées, d'ordonner convenablement les divers sous-ensemble de règles, et de les activer au bon moment …* » [Berrendonner, 1990].

  Bien que cette approche soit extrêmement intéressante sur le plan théorique et qu'elle semble être proche du modèle de compréhension humain, sa réalisation informatique ne semble pas être évidente, plus particulièrement, pour la langue arabe présentant une forte proportion de formes ambiguës (voir Section 2.5.3). Nous signalons qu'Alain Berrondonner, lui-même, reconnaît cette difficulté de mise en œuvre :

  « *… il nous faut reconnaître que nous n'avons guère avancé dans la définition d'un tel programme stratégique, et nous nous contentons pour l'instant de traiter cette question de façon totalement empirique* » [Berrendonner, 1990].

- **Approche modulaire**

  Dans le cadre d'une approche modulaire, toutes les connaissances linguistiques sont séparées de l'algorithme d'analyse lui-même ; elles sont décrites dans des structures de données autonomes. Par exemple, nous pouvons les retrouver dans des formats textuels lisibles et accessibles par un non-informaticien qui peut aisément les contrôler et les modifier sans avoir à maîtriser l'ensemble du programme [Sabah, 1989]. Cette approche, nommée aussi approche déclarative, permet de séparer les différents niveaux d'analyse. Ainsi, la sortie de l'analyse morphologique (incluant les cas d'ambiguïtés si présents) constitue l'entrée de l'étape d'analyse syntaxique qui suit. Cette analyse tente d'explorer les résultats ainsi fournis afin d'éliminer les solutions indésirables par application de règles syntaxiques. Le principal avantage de cette deuxième approche réside dans la division du travail de formalisation des connaissances linguistiques entre l'informaticien (celui qui écrit les programmes de génération et d'analyse) et le linguiste (celui qui définit les lexiques et les grammaires). La relative commodité de sa mise en place a persuadé la majorité des concepteurs des modèles d'analyse actuels à l'adopter [Ouersighni, 2002].

Compte tenu des difficultés de mise en place de la première approche, la plupart des analyseurs actuels procèdent par une approche déclarative ou modulaire. Cependant, même dans le cadre d'une même approche, différentes stratégies de représentation et d'analyse sont envisageables. Ces stratégies se différencient selon la proportion des connaissances linguistiques codées et les règles d'analyse mises en place. Nous distinguons trois différentes stratégies d'analyse:

3.1.1 Analyse à base de racines et schèmes

Cette analyse se base sur une décomposition des mots en une racine et un schème en s'appuyant sur un ensemble de règles de transformations et de concaténations. La racine est une unité lexicale abstraite commune à plusieurs mots, constituée d'une séquence de trois (rarement deux ou quatre) consonnes linéairement ordonnées et véhiculant un sens général. Le schème est un ensemble de modèles de vocalisations, ou encore de combinaisons de consonnes et de voyelles avec indication des endroits d'insertion de ces dernières. Ce processus d'insertion s'appelle habituellement l'interdigitation [Beesley, 2001].

Racine	درس (*drs*) [$C_1C_2C_3$][9]			
Schèmes	$C_1\acute{}C_2\acute{}C_3\acute{}$ ($C_1aC_2aC_3a$)[10]	$C_1\acute{}C_2C_2\acute{}C_3\acute{}$ ($C_1aC_2C_2aC_3a$)	$C_1\acute{ا}C_2\acute{}C_3\acute{}$ ($C_1aāC_2iC_3$)	مُ$C_1\acute{}C_2C_2\acute{}C_3$ ($muC_1aC_2C_2iC_3$)
Formes générées	دَرَسَ (*darasa* - étudier)	دَرَّسَ (*darrasa* - enseigner)	دَارِس (*daāris* - étudiant)	مُدَرِّس (*mudarris* - enseignant)

Tableau 14 : Interdigitation de la racine verbale "درس" et ces schèmes

En dépit du nombre limité de données à stocker que requiert cette stratégie d'analyse, elle nécessite le plus grand nombre de connaissances linguistiques écrites sous forme de règles morphologiques de réécriture, de règles syntaxiques, de règles phonologiques ou d'harmonie vocalique, etc. En effet, la prise en compte des phénomènes morphologiques et syntaxiques particuliers de la langue arabe conduit à multiplier dangereusement les règles de la grammaire et à accroître considérablement le nombre de règles.

Il y a eu de nombreux débats entre les adeptes et les opposants de l'emploi de la racine comme forme de base. Beesley (2001) a défendu « *la réalité linguistique des racines sémitiques* » et a cité le fait que les dictionnaires traditionnels sont classés par racines comme motivation pratique. Darwish (2002) a même soutenu l'idée que « *l'utilisation des racines en arabe comme termes d'indexation améliore sensiblement l'efficacité de récupération de l'information par rapport à l'utilisation des lexèmes ou de lemmes* ».

Cependant, plusieurs chercheurs ont critiqué cette approche. Kamir et al. (2002) ont supposé que le lexème est le lemme ou l'unité grammaticale de base en arabe et ont conclu que la racine n'est qu'un « *super-lemme* » abstrait qui regroupe tous les mots qui partagent un même champ sémantique. Ce constat ne peut pas occulter le rôle de la racine dans le processus de formation de mot, la flexion et la dérivation. Toutefois, Dichy et Fargaly (2003) ont mené une étude à ce sujet où ils ont conclu qu'un système à base de racines et schèmes enferme des problèmes de surgénération

[9] [$C_1C_2C_3$] représente les trois consonnes de la racine, où : C_1= "د" (*d*), C_2="ر" (*r*) et C_3="س" (*s*).
[10] Le Schème « $C_1\acute{}C_2\acute{}C_3\acute{}$ » ($C_1aC_2aC_3a$) est équivaut à l'ensemble des opérations suivantes : 1/ insérer la voyelle brève "´(*a*) après la première consonne C_1 + 2/ insérer la même voyelle à la suite de la deuxième consonne C_2 + 3/ faire de même à la suite de la troisième consonne C_3.

se manifestant par la production de nombreuses formes qui n'appartiennent pas réellement au lexique de la langue. Ils finissent par mettre en cause l'efficacité d'un tel système pour la recherche d'information. Ils donnent l'exemple des verbes : أمِنَ (*áamina* - être sûr), أمُنَ (*áamuna* - être honnête) et آمَن (*âamana* - croire) qui, bien qu'ils aient la même racine, chacun d'eux a un modèle flexionnel différent et appartient à un champ sémantique distinct. De façon analogue, le nom أمـان (*áamaān* - sécurité), أمانـة (*áamaānat* - loyauté) et إيمـان (*íiymaān* – la foi) ne devraient pas être connexes dans une application de recherche d'information [Dichy et Fargaly, 2003].

3.1.2 Analyse à base de lexèmes

Cette stratégie privilégie le stockage d'un maximum d'informations à l'intérieur des données lexicales. Elle ne nécessite, ainsi, qu'un nombre réduit de règles pour l'analyse. Cette approche se base sur des lexèmes[11] définis comme étant le plus petit morphème, appelé aussi unité minimale, dans un mot graphique qui n'est ni fonctionnel, ni dérivationnel. Ce morphème est, généralement, dépourvu de toute marque flexionnelle ; il ne contient ni affixes de déclinaison ou conjugaison, ni clitiques. Généralement, en arabe, nous trouvons les lexèmes verbaux représentés par la forme fléchie à l'accompli, 3ème personne, masculin, singulier, et dans le cas des noms et des adjectifs, les lexèmes prennent la forme indéfinie au singulier. Des matrices de compatibilités sont, par la suite, mises en place afin d'associer chaque entrée aux affixes et autres informations utiles en vue de formaliser les modèles de conjugaisons des verbes et les modèles de déclinaisons des noms et des déverbaux. Par l'intermédiaire de relations plus complexes, il est, aussi, possible de retrouver la racine, le schème, le nombre, le mode, le cas, etc. Quant à l'analyse, elle se base sur deux types de variations morphologiques :

- **Par suffixation et préfixation** : par exemple, il suffit de greffer le suffixe "ـَات" (*aāt*) au lexème "مُدَرِّس" (*mudarris* – enseignant) pour passer du masculin singulier au féminin pluriel : "مُدَرِّسَات" (*mudarrisaāt* – enseignantes) ;
- **Par fléchage ou « dérivation interne »** : par exemple, le passage de la forme au singulier "مَدْرَسَة" (*madrasat* – école) à la forme plurielle "مَدَارس" (*madaāris* – écoles) nécessite un certain nombre de transformations morphologiques.

A vrai dire, nous voyons que, dans cette stratégie, la majorité des règles d'analyse sont greffées directement dans le lexique. Ceci ne peut qu'ajouter une couche de complexité pour la maintenance et la mise à jour de l'ensemble des ressources linguistiques.

3.1.3 Analyse à base de lemmes

Cette stratégie est reconnue comme étant la solution intermédiaire entre les deux approches déjà citées. Elle permet de faire l'équilibre entre la proportion des données lexicales à stocker et des connaissances linguistiques à coder. Plusieurs définitions ont été attribuées aux lemmes, parmi lesquelles nous citons :

« *Le lemme est l'unité autonome constituante du lexique d'une langue. C'est une suite de caractères formant une unité sémantique et pouvant constituer une entrée de dictionnaire…* » (Source : Wikipedia)

L'utilisation des lemmes permet d'éviter les deux problèmes majeurs posés par les deux autres stratégies :

- La surgénération de formes qui n'appartiennent pas réellement au lexique de la langue ;
- La perte des liens sémantiques qui existent entre les différentes formes fléchies rattachées à une même forme canonique : racine ou lemme.

[11] Un lexème est la sous-chaîne présente dans toutes les formes d'un même terme fléchissable : aimer, aime, aimé, aimions ou aimât sont des formes différentes d'un même lexème aim-. [Wikipédia]

3.2 Etudes sur la représentation des dictionnaires électroniques

Diverses études sur la représentation des données lexicales sous forme de dictionnaires électroniques ont été menées ; parmi les plus récentes, nous citons les travaux réalisés au sein de la société Sakhr par [Ait Taleb, 2005] et ceux rélisés par Ben Hamadou et al. [Baccar et al., 2008]. Ces études ont montré une diversité des modèles adaptés. Cette diversité admet plusieurs facettes :

- le type des entrées des dictionnaires électroniques : les lemmes, les racines ou les lexèmes ;
- la description des informations morpho-syntaxiques qui se rattachent à chacune des entrées lexicales ;
- l'organisation de ces entrées dans le dictionnaire ; par exemple, nous trouvons une classification des entrées lexicales selon l'ordre alphabétique direct comme le dictionnaire "لسان العرب" (*lisaān al-ʿarab*), selon l'ordre alphabétique inversé tel que le dictionnaire "الغنيّ" (*al-ġaniy*), selon l'ordre phonétique tel est le cas pour le dictionnaire "العين" (*al-ʿayn*) ;
- le mode de liaison entre l'entrée lexicale et les différentes formes fléchies, déclinées ou dérivées qui s'y rattachent.

A titre d'exemple, nous citons les dictionnaires de l'arabe "الوسيط" (*al-wasiyṭ*), "المصباح" (*al-muṣbaāḥ*) et "المنير" (*al-muniyr*) organisent leurs entrées selon l'ordre alphabétique de la racine du mot. Cette organisation permet d'inclure toutes les formes verbales et nominales dérivées d'une racine sous la même entrée ; l'analyse lexicale qui s'ensuit serait fondée sur des racines et des schèmes.

Récemment, un modèle normatif et représentatif des données des dictionnaires arabes a été proposé dans [Baccar et al., 2008]. La démarche suivie pour l'élaboration de ce modèle respecte le standard LMF[12] « Lexical Markup Framework » (ISO/TC 37). Le but d'une telle standardisation est de :

- fournir un modèle commun pour la création et l'utilisation des ressources langagières ;
- gérer l'échange des données entre ces ressources ;
- permettre la fusion d'un grand nombre de ressources électroniques afin de constituer un vaste réseau de descriptions linguistiques.

Le travail réalisé a permis de proposer des extensions morphologiques, syntaxiques et sémantiques au noyau de LMF afin de tenir compte des spécificités de la langue arabe. Le modèle de représentation ainsi obtenu ressemble à celui que nous décrirons dans la section 4.2.

3.3 Quelques analyseurs morphologiques de l'arabe

Depuis plusieurs décennies, des recherches se sont poursuivies dans le cadre du traitement automatique de la langue arabe afin de proposer un analyseur morphologique robuste. L'un des premiers théoriciens de ce domaine, David Cohen, a proposé un essai d'analyse automatique dès 1961 [Cohen, 1961/1970]. Par la suite, les recherches se sont multipliées et plusieurs travaux ont été menés pour le développement d'analyseurs morphologiques pour l'arabe. Vu que la majorité de ces travaux sont des applications commercialisées ou des propriétés privées de laboratoires de recherche, seulement une minorité d'entre eux sont disponibles pour la recherche et l'évaluation. Parmi les analyseurs connus dans la littérature, nous citons :

3.3.1 Premier essai d'analyse automatique par David Cohen

Le premier essai d'analyse automatique de la langue arabe a été proposé par David Cohen dès les années 1960. Cet analyseur morphologique a été, par la suite, programmé en Lisp par J. Mac Carthy puis en COMIT par Y. Yngve. La méthode proposée est une apriorique : elle part d'un traitement morphologique minimaliste basé sur de simples considérations des caractéristiques

[12] http://fr.wikipedia.org/wiki/Lexical_markup_framework

structurelles de la langue arabe. Ainsi, David Cohen s'est octroyé une facilité remarquable de mise en œuvre du programme ; mais, ceci en a restreint la portée pratique. L'analyse s'appuie sur le principe que toute base d'une forme linguistique c'est-à-dire toute forme linguistique dépourvue de ses éléments flexionnels, peut s'analyser fondamentalement en une racine et un schème. Cependant, la réalité de la racine a été mise en doute par divers sémitisants comme C. Brockemann qui n'y voyait rien d'autre qu'un moyen commode mais artificiel de classement. Plus tard, à partir d'une étude menée sur la dérivation nominale en arabe, D. Cohen remarque que la formule de croisement de racines et schèmes était insuffisante pour une bonne partie du lexique de l'arabe dont la construction nécessitait d'autres constituants comme les suffixes. Cette première tentative est toujours considérée comme un premier essai d'une analyse purement grammaticale où les problèmes syntaxiques tels que ceux liés à la traduction étaient ignorés.

3.3.2 Analyseur morphologique de Buckwalter - AraMorph

L'analyseur morphologique de Buckwalter, baptisé AraMorph, est bien connue dans la littérature et a été même considérée comme « la ressource lexicologique la plus respectée de son genre» [Hajič et al., 2005]. Il est employé au sein de l'étiqueteur grammatical de l'arabe au LDC[13], Penn Treebank de l'arabe, et la Dependancy Treebank de l'arabe à Prague. AraMorph est actuellement développé sous Java et, comme son ancêtre en Perl, travaille sur une translittération du mot arabe. Cette translittération utilise naturellement le système de translittération de Buckwalter. Ainsi, la forme "كتاب" est translittérée en « *ktAb* »[14] avant de passer à l'étape d'analyse morphologique.

Le système adopte une approche procédurale pour la représentation des différentes ressources linguistiques. Certaines règles orthographiques nécessaires à l'analyse sont construites directement dans le lexique lui-même au lieu d'être spécifiées en tant que règles générales qui interagissent pour produire les sorties de l'analyse.
Les données lexicales sont représentées dans 3 bases de données [Buckwalter, 2002]:

- Les préfixes : 299 entrées
- Les suffixes : 618 entrées
- Les lexèmes : 82 158 entrées représentant 38 600 lemmes

Comme montré sur la Figure 17, chaque entrée du lexique prend un lexème comme forme de base et nous fournit des informations qui concernent la racine, la catégorie grammaticale et la traduction anglaise, excepté les préfixes et suffixes.

و/wa	Pref-Wa	*and*
ب/bi	NPref-Bi	*by/with*
وب/wabi	NPref-Bi	*and + by/with*
ال/Al	NPref-Al	*the*
بال/biAl	NPref-BiAl	*with/by + the*
وبال/wabiAl	NPref-BiAl	*and + with/by the*
ة/ap	NSuff-ap	*[fem.sg.]*
تان/atAni	NSuff-atAn	*two*
تين/atayoni	NSuff-tayn	*two*
تاه/atAhu	NSuff-atAh	*his/its two*
ات/At	NSuff-At	*[fem.pl.]*

;; $_1$كَتَبَ /katab-u$_1$		
كَتَبَ /katab	PV	*write*
كُتُبَ /kotub	IV	*write*
كُتِبَ /kutib	PV_Pass	*be written; be fated; be destined*
كُتَبَ /kotab	IV_Pass_yu	*be written; be fated; be destined*
;; $_1$كِتَاب /kitAb$_1$		
كِتَاب /kitAb	Ndu	*book*
كُتُب /kutub	N	*books*
;; $_1$كِتَابَة /kitAbap$_1$		
كِتَاب /kitAb	Nap	*writing*

Figure 17 : Quelques entrées lexicales de l'analyseur AraMorph

[13] LDC : Linguistic Data Consortium, université de Pennsylvanie
[14] Voir table de translittération de Tim Buckwalter (http://www.qamus.org/transliteration.htm)

Ces trois bases lexicales sont en interaction avec trois tables de compatibilité morphologique afin de pouvoir traiter les concaténations. Ces tables sont utilisées pour contrôler les combinaisons entre préfixes et lexèmes (1 648 entrées), entre lexèmes et suffixes (1 285 entrées), et entre préfixes et suffixes (598 entrées). Elles sont alors utilisées pour préciser les possibilités de co-apparition des catégories morphologiques. Par exemple, la catégorie morphologique de la conjonction de coordination "و" (*wa* - et), Pref-Wa, est compatible avec tous les radicaux nominaux et radicaux verbaux à l'accompli. Toutefois, Pref-Wa n'est pas compatible avec les radicaux verbaux à l'inaccompli puisque ces derniers nécessitent un préfixe sujet.

Le lexème كِتَاب / kitAb du lemme $_1$كِتَاب/ kitAb$_1$ (livre) a pour catégorie (Ndu), qui n'est pas compatible avec le marqueur du féminin ة/ap ayant pour catégorie NSuff-ap. Le même lexème, كِتَاب / kitAb, apparaît comme l'un des radicaux du lemme $_1$كتابة/kitAbap$_1$ (écriture) accompagné de l'information : « nécessite un suffixe avec un marqueur féminin ». De tels cas sont assez fréquents et constituent un défi à l'utilisation des lexèmes comme des formes de base compte tenu de l'ambiguïté supplémentaire inutilement engendrée.

Ce système d'analyse à base de lexèmes utilise un algorithme d'analyse assez simple puisque toutes les décisions d'analyse sont codées dans le lexique et les matrices de compatibilité [Habash, 2004]. Cependant, lorsqu'il s'agit de l'analyse d'une forme agglutinée, les segmentations ne seraient valables que si les différents composants existaient dans le lexique et sont triplement compatibles (préfixe - lexème, lexème - suffixe et préfixe - suffixe). Parmi les insuffisances de cet analyseur, nous pouvons citer [Attia, 2006] :

- *L'absence de règles générales de génération.* En effet, tous les lemmes sont listés manuellement. Pour chaque entrée, tous les lexèmes des formes fléchies associées sont énumérés. Ceci augmente, considérablement, le coût de maintenance du lexique. Le système n'est donc pas approprié à la génération.

- *Insuffisance au niveau du traitement du proclitique de question* qui s'agglutine en tête des verbes et des noms. Il est probable que ceci ait été prévu pour réduire l'ambiguïté ; mais, ça limite la couverture du système. Désormais, les deux formes suivantes sont inanalysables :
  - ✓ أأقول (*áa áaquwl* - est ce que je dis ?)
  - ✓ أمحمد (*áa muḥammad* - est-ce Mohamed ?)

- *Manque de spécification de certaines formes à l'impératif.* Seulement 22 verbes sur un total de 9198, soit %0.002, ont des formes impératives. Ceci limitera, entre autres, l'analyse des expressions d'ordre. En l'occurrence, malgré l'homographie entre la forme impérative et celle à l'inaccompli que présente les verbes suivants, seule l'analyse associée au deuxième cas serait attribuée :
  - ✓ حاول → "حَاوِلْ " (*ḥaāwil* - essaye !) ou "حَاوَلَ " (*ḥaāwala* – il a essayé)
  - ✓ انتظر → "اِنْتَظِرْ " (*íintaẓir* - attends !) ou "اِنْتَظَرَ " (*íintaẓara* – il a attendu)

- *Manque de spécification de certaines formes à la voix passive.* Sur 9198 verbes, seulement 1404 verbes (%15) sont conjugués à la voix passive au présent, contre uniquement 110 verbes au passé. Certaines formes passives ayant une fréquence d'apparition assez élevée, ne sont pas prises en compte par l'analyseur, nous citons :
  - ✓ يُقابل (*yuqaābalu* - est rencontré)
  - ✓ يُسْتَعْمَل (*yusta'malu* - est utilisé)

  Cependant, nous trouvons des formes passives n'ayant qu'un sens impresonnel et à très faible probabilité d'apparition dans les textes contemporains telles que :
  - ✓ يُمَات (*yumaātu* – on meurt)

- ✓ يُعَاش (*yu'aāšu* – on vit)[15]

- *Inclusion de certaines entrées de l'arabe classique.* L'inclusion de telles entrées dans le lexique d'un analyseur morphologique est très coûteuse en termes d'ambiguïté. Ainsi, les entrées devraient être extraites à partir des données contemporaines, plutôt qu'à partir des dictionnaires traditionnels, surtout si elles sont censées traiter les textes modernes. Dans la littérature, plusieurs indices indiquent que Buckwalter et Xerox ont pris comme référence le dictionnaire arabe-anglais de Hans Wehr (1979)[16]. Cependant, dans l'introduction de ce dictionnaire, Hans Wehr a déclaré que le dictionnaire « *énumère non seulement des mots classiques mais aussi des expressions du modèle rhétorique élégant. Ceux-ci sont mis côte à côte avec les nouvelles notions conformément aux demandes des puristes, il contient également des néologismes, traductions d'emprunts et expressions familières qui peuvent ne pas être au goût linguistique de beaucoup d'arabes instruits* » [Wehr, 1979].

  En effet, Buckwalter inclut quelques termes qui ne sont plus utilisés, nous pouvons citer :
  - ✓ قفّ (*qaffa* - sécher)
  - ✓ قلفط (*qalfaṭa* - calfater, boucher)
  - ✓ حنيفة (*ḥaniyfat* - orthodoxe)

  Notons que ce dernier exemple est utilisé, en arabe moderne, comme étant un prénom, alors que l'analyseur de Backwalter l'associe plutôt à des sens et des traductions anglaises qui n'étaient utilisés qu'en arabe classique.

- *Aucune reconnaissance des expressions figées.* Les expressions figées en arabe sont assez fréquentes dans les textes, leurs identifications ajoutent un sens de certitude à l'analyse et réduisent l'ambiguïté. Cependant, quand ces expressions figées sont analysées compositionnellement, elles perdent leur signification et amplifient le problème d'ambiguïté car leurs éléments peuvent être individuellement ambigus.

- *Règles inexactes de décomposition orthographique.* Buckwalter a justifié l'inclusion de certaines règles de décontraction de formes mal orthographiées par le fait qu'elles sont communes dans les données analysées [Buckwalter, 2004]. Habituellement, les mots mal orthographiés bénéficient d'un traitement spécifique ; cependant, cet analyseur essaie de les analyser à l'aide de règles généralisées et sans aucune heuristique particulière. A défaut de telles heuristiques, ces règles peuvent s'appliquer sur des mots correctement écrits en entraînant une surproduction des analyses proposées et donc une augmentation du niveau d'ambiguïté. A ce propos, nous pouvons citer le remplacement de tout *álif* de la forme en entrée par une *hamzä* même lorsque celle-ci se trouve à l'intérieur du mot tels que :
  - ✓ فاشل (*faāšil* - inopérant), ce mot est aussi analysable en tant que :
    فأشل (*fa áašullu* - alors je paralyse)
  - ✓ واقف (*waāqif* - être debout), de la même façon, ce mot est analysable en tant que :
    وأقف (*wa* áaqifu - et je me tiens debout)

3.3.3 Analyseur morphologique de Xerox

L'analyseur morphologique de Xerox « *est basé sur une innovation de la technologie à états finis* » [Dichy et Fargaly, 2003]. Il adopte l'approche déclarative pour une analyse morphologique à base de racines et schèmes. Il inclut 4 930 racines et 400 modèles, produisant efficacement de

[15] On peut trouver cette forme dans des phrases impersonnelles telle que : 'يُعَاش طويلا في بلدتنا ' (*yu'aāšu ṭawiylan fiy baldatnaā* – dans notre village, on vit plus longtemps)

[16] La version éditée en 1979 du dictionnaire de H. Wehr a été revue par J. Milton Cowan partant de sa version originale qui date de 1920. Cette version reste la plus complète et la plus fiable des dictionnaires bilingues d'arabe moderne à usage général, même s'il fait une une place limitée aux terminologies techniques récentes.

90.000 lexèmes. Les avantages de cet analyseur résident dans l'utilisation de règles à large couverture. Il restitue également les marques de vocalisations et fournit un glossaire anglais pour chaque mot. Toutefois, le système a hérité beaucoup d'insuffisances de l'analyseur morphologique de Buckwalter telles que l'inclusion de termes de l'arabe classique, l'absence de spécifications des mots composés et les règles inexactes pour la décomposition orthographique. Parmi les carences additionnelles de l'analyseur morphologique de Xerox, nous citons :

- *Une surgénération dans la dérivation des mots.* La distribution des modèles pour des racines n'étant pas régulière, et bien que chaque racine ait été codée manuellement dans le système pour choisir parmi les 400 modèles, le travail est tout naturellement à forts risques d'erreurs.
- *Un manque de spécification des combinaisons potentielles des différents morphèmes d'une forme agglutinée*. A ce propos, nous citons le problème d'analyse provenant du fait qu'un pronom personnel à la première personne, singulier ou pluriel, ne peut pas être agglutiné à une forme verbale conjuguée à la même personne (voir Section 2.5.2.3), nous parlons alors de « *Co-référence du sujet et de l'objet* » [Dichy, 2001]. Malgré cette règle, la forme :
  نضربنا (*naḍribunaā* - nous nous sommes frappés) reste décomposable avec cet analyseur.
- *Un taux d'ambiguïté assez élevé.* En raison des facteurs mentionnés ci-dessus, le système souffre d'un niveau élevé d'ambiguïté ; il fournit tant d'analyses (bon nombre d'entre elles fausses) pour la plupart des mots.

3.3.4 Analyseur morpho-syntaxique AraParse

L'analyseur AraParse est basé sur les ressources lexicales du projet européen DIINAR (DIctionnaire INformatisé de l'ARabe). Cet analyseur est conçu suivant une architecture modulaire [Ouersighni, 2002]. Il utilise des ressources linguistiques à large couverture. Cet analyseur distingue trois organisations différentes du lexique :

- *Un lexique de mots minimaux* constitués de formes canoniques (lemmes ou bases), proclitiques, préfixes, suffixes, et enclitiques avec un algorithme relativement complexe qui traite l'agencement des différents morphèmes.
- *Un lexique de formes fléchies minimales* (préfixe-base-suffixe) avec de petites listes de proclitiques et d'enclitiques. Dans ce cas, l'algorithme d'analyse est moins complexe dans la mesure où il ne traite que les phénomènes d'agglutination des enclitiques et ne s'occupe plus des phénomènes de flexion et de dérivation.
- *Un lexique de toutes les formes fléchies maximales* (le mot arabe en entier). Dans ce cas, il suffit d'effectuer une simple consultation du lexique pour extraire les informations relatives au mot à analyser.

Malgré la richesse des données lexicales mises en jeu, cette approche reste très coûteuse en termes de maintenance et de synchronisation.
Le processus d'analyse passe par plusieurs étapes intermédiaires telles que les régularisations pré-morphologiques (traitement de la Hamza, la šadda , etc.), la dévoyellation des entrées, … De tels choix ne peuvent qu'augmenter considérablement le temps de traitement et compliquer la tâche d'un non-informaticien quant à la mise à jour des ressources de l'analyseur.

3.3.5 Autres analyseurs morphologiques de l'arabe

Mis à part les analyseurs de Buckwalter et de Xerox qui sont les plus connus, cités en littérature, bien documentés et disponibles pour l'évaluation, plusieurs autres travaux d'élaboration d'analyseurs morphologiques et morpho-syntaxiques ont vu le jour ; mais, sans jamais être disponible pour la communauté scientifique. Outre l'analyseur AraParse développé à l'université Lyon2 et décrit ci-dessus, nous pouvons encore citer les travaux suivants :

- *AMSAAR (1998)* est un Analyseur Morpho-Syntaxique Assisté de l'Arabe qui offre à l'utilisateur la possibilité d'intervenir après chaque étape d'analyse automatique pour la vérification ou la correction du résultat de l'analyse. Ce système a été conçu et réalisé au CNRS[17] au sein de l'équipe de F. Debili [Achour, 1998] ;
- *Berri, Zidoum et Atif (2000)* ont développé un analyseur morphologique qui comporte trois principaux composants: une base de connaissances contenant les règles régulières et irrégulières de la grammaire arabe, un ensemble de listes de mots contenant les exceptions traitées par les règles irrégulières et un algorithme d'interaction entre les formes à analyser ainsi que les règles d'analyse correspondantes ;
- *Sebawai (2002)* est un analyseur morphologique développé par [Darwish, 2002] en un jour ! Cet analyseur utilise des modèles orthographiques pour retrouver les racines. "*... sa couverture n'est pas parfaite ... cet analyseur morphologique reconnaît la racine des mots avec un taux de succès de 84%*" ;
- La société *Sakhr* a développé, de son coté, un analyseur morphologique par [Chalabi, 2004]. Le site web de Sakhr[18] affirme que cet analyseur traite aussi bien l'arabe moderne que l'arabe classique. Il identifie la forme de base par suppression de tous les affixes du mot à analyser, et décrit la structure morphologique de celui-ci.

Bien que naturellement non exhaustif, le panorama de ces différents travaux portant sur l'analyse morpho-syntaxique de la langue arabe a permis d'établir les différentes méthodes et approches existantes. L'intérêt d'une telle démarche étant de pouvoir positionner nos travaux par rapport à ces avancées.

Néanmoins, cet état de l'art et la description des problèmes morpho-syntaxiques soulèvent certaines interrogations quant à l'approche de représentation formelle des données et la stratégie d'analyse les plus appropriées pour garantir la robustesse et la rapidité des traitements. Dans la suite de ce mémoire, nous adopterons comme environnement de travail, la plateforme linguistique de développement ***NooJ***.

3.4 Quelques travaux sur la voyellation automatique

En dépit du nombre d'analyseurs automatiques mis en œuvre pour le traitement de la langue arabe, le problème de la voyellation reste insuffisamment étudié. A vrai dire, les travaux sur la voyellation ont commencé assez tardivement. Parmi les premiers travaux, nous citons celui de H. Achour [Achour, 1998] qui, partant d'une étude comparative du problème de la voyellation automatique et de celui de la réaccentuation automatique, propose un système de restitution des voyelles. Ce système commence par associer, à chaque unité lexicale reconnue dans le texte, l'ensemble de ses vocalisations potentielles. Ensuite, il résout ou réduit le plus possible l'ambiguïté de voyellation de chaque unité en faisant intervenir des connaissances de nature morphologique et syntaxique. Plus tard, ce travail a aboutit sur un système interactif d'aide à la voyellation, baptisé VOWEL, qui a été intégré dans le logiciel de traitement de texte Microsoft Word [Mesfar, 2003].

Simultanément, la société Sakhr[19] a commercialisé un système de voyellation automatique. Ce système est totalement fermé et la seule information fournie par les concepteurs est celle attestant une précision qui atteint les 98%.

Au début des années 2000, Y. Gal a utilisé un modèle de Markov caché (Hiden Markov Model – HMM) basé sur un apprentissage effectué sur des textes totalement voyellés, notamment le Coran

[17] CNRS : Centre National de la Recherche Scientifique, France
[18] http://www.sakhr.com/
[19] Site web de la société : http://www.sakhr.com/Technology/Diacritization/Default.aspx

[Gal, 2002]. L'outil développé a atteint 85% de bonnes vocalisations en situation ad-hoc ou auto-cohérence, c'est-à-dire que ce pourcentage est obtenu sur un texte qui appartient au corpus d'apprentissage.

En 2005, Nelken et Shieber ont utilisé un transducteur à états finis pondéré basé sur un apprentissage réalisé sur les corpus du LDC de M. Maamouri et al. Les résultats rapportés montrent environ 93% de bonnes vocalisations [Nelken et Shieber, 2005].

En 2006, Safadi et al. ont développé une technique de voyellation non-contrôlée. Ils s'appuyaient sur une combinaison de catégorisation grammaticale automatique et de règles écrites manuellement. Selon une évaluation d'expert sur les résultats fournis, le taux d'erreur varie entre 10 et 20% [Safadi et al. , 2006].

Dans la même période, Zitouni et al. (2006) ont présenté une approche basée sur un calcul de l'entropie maximale pour la restauration d'une liste exhaustive des voyelles. Ils ont aboutit à un taux d'erreur de 5,1% au niveau des voyelles contre 17,3% au niveau des mots. Ils ont également remarqué une nette amélioration dans les taux d'erreur lorsque les voyelles casuelles sont omises. Dans ce dernier cas, une amélioration des scores a été nettement remarquée dans environ 60% des cas. Ces résultats assez encourageants n'occultent pas certaines insuffisances de ce travail tel que le traitement de la *hamzä*. En revanche, parmi ces apports remarquables, nous citons l'utilisation d'une catégorisation grammaticale fine.

Conjointement, le travail de M. Maamouri sur la voyellation [Maamouri, 2006] venait soutenir cette dernière théorie après une étude entreprise sur les annotations du treebank de la langue arabe élaborée à l'université de Pennsylvanie. Cette étude a révélé la forte liaison entre voyellation, catégorisation grammaticale et annotation.

A l'issue de cette étude, nous pouvons constater que le problème de la présence partielle ou l'absence des voyelles des formes d'un texte pose davantage de difficultés au niveau du développement des analyseurs morphologiques automatiques. En effet, certains travaux récents tels que ceux menés par M. Attia à l'université de Manchester [Attia, 2006] et ceux de M. Alharabi à l'université Paris-Sorbonne [Alharabi, 2006] ont eu recours à une dévoyellation des entrées de l'analyseur afin de pouvoir garantir la réalisabilité des tâches d'analyse à accomplir.

3.5 NooJ : une plateforme de développement linguistique

NooJ, tout comme son prédécesseur INTEX, est un environnement de développement linguistique utilisé comme outil de formalisation des langues naturelles et de développement d'applications de traitement automatique des langues naturelles (TALN). Aujourd'hui, il existe des modules linguistiques NooJ pour une douzaine de langues. Une demi-douzaine d'applications informatiques du TAL ont été construites avec NooJ (voir http://www.nooj4nlp.net pour des exemples d'utilisation de NooJ).

Les fonctionnalités de NooJ sont adaptées à un public très varié incluant des linguistes (description de la morphologie et de la syntaxe des langues, analyse de corpus), des informaticiens du TAL (applications d'extraction d'information), ainsi que des enseignants tant en Linguistique qu'en enseignement des langues [Garrigues, 1999]. Le nouveau moteur linguistique du logiciel NooJ a été réécrit à partir de son prédécesseur INTEX (voir http://intex.univ-fcomte.fr) pour apporter plus de facilité de manipulation et répondre aux besoins de la communauté qui s'est constituée autour de lui. Parmi les principales caractéristiques de NooJ, nous citons :

- **Architecture intégrée :**
  Dans NooJ, toutes les connaissances linguistiques sont séparées de l'algorithme d'analyse lui-même. L'architecture globale de NooJ est basée sur un ensemble de modules linguistiques : orthographique, flexionnel, morphologique, dérivationnel et syntactico-sémantique. Cette

structure est garnie par un ensemble d'outils tels que : un éditeur de graphes, un concordancier, un débogueur de grammaires, des outils statistiques, etc. Cette collection constitue un ensemble en interaction permettant de répondre aux besoins les plus pointus de ses utilisateurs et notamment des spécialistes du traitement des langues naturelles d'entre eux.

Cette modularité a été ornée par une architecture intégrée assurant l'efficacité de l'interaction entre les différents modules. Traditionnellement, tel est le cas pour INTEX, les logiciels de TAL utilisent une architecture en flot de données composée de plusieurs composants logiciels reliés entre eux par des flux de données. L'information circule dans le réseau, par le biais de pipelines par exemple ; elle est, ensuite, transformée par les différents composants qu'elle traverse. Lorsque les composants se distribuent sur une seule ligne et qu'ils ne font que passer l'information transformée à leur voisin, nous parlons alors d'architecture par lot (batch). Dans ces cas, les différents composants perdent du temps à l'écriture de données dans des fichiers intermédiaires (stockés sur le disque) qui doivent être rechargés à chaque appel. Cette interaction est de plus en plus coûteuse lorsqu'il s'agit d'applications à grandeur réelle. Cependant, l'intégralité des composants dans NooJ est regroupée dans une seule application ne chargeant les ressources requises qu'une seule fois.

- **Architecture orientée objet :**
  NooJ est développé sous une architecture orientée objets [Silberztein, 2004]. Les composants du système (objets) intègrent les données ainsi que les routines nécessaires pour leurs traitements. La communication et la coordination entre les objets sont réalisées par un mécanisme interne. Cette architecture repose souvent sur les trois piliers : encapsulation, héritage et polymorphisme. Elle présente les avantages suivants:
  - ✓ Eviter la redondance dans le code source grâce au concept d'héritage. Ceci rend le code source plus lisible et gérable ;
  - ✓ Accéder directement à toutes les méthodes publiques dans NooJ, au lieu d'avoir à sélectionner un ensemble de sous-programmes. Ceci apporte une flexibilité et utilisabilité supplémentaires au système ;
  - ✓ Pouvoir ajouter d'autres fonctionnalités supplémentaires ou des méthodes spécifiques à des langues sans avoir à modifier l'architecture globale de l'application.

- **Utilisation de la technologie à états finis :**
  NooJ contient des outils permettant d'éditer, de tester, de déboguer et de gérer des ensembles importants de graphes à états finis. Tous les objets traités par NooJ sont ou peuvent être transformés sou forme de transducteurs à nombre fini d'états. Toutes les opérations sur les textes, grammaires et dictionnaires se ramènent ainsi à des opérations sur des transducteurs. NooJ utilise aussi des outils plus puissants comme les graphes récursifs qui sont équivalents à des grammaires hors-contexte. Lorsque ses transducteurs récursifs à variables sont utilisés en cascade, ils donnent à NooJ la puissance d'une machine de Turing [Silberztein, 1999b]. Toutefois, l'application impose toujours au linguiste une approche ascendante, et en largeur : chaque phénomène doit être traité localement par un ou plusieurs graphes le plus souvent à états finis, ces graphes sont ensuite réutilisables pour traiter d'autres phénomènes plus généraux. Cette approche en cascade met en lumière la complexité de la variation lexicale et la faiblesse de toute généralisation.

- **Développement de ressources linguistiques à large couverture :**
  NooJ est un environnement de développement linguistique permettant de construire, de tester et de gérer des descriptions formalisées à large couverture des langues naturelles, sous forme de dictionnaires et de grammaires électroniques. Les différentes ressources linguistiques sont décrites dans des structures de données autonomes [Silberztein, 2005b]. En l'occurrence, elles sont représentées soit dans des formats textuels lisibles et accessibles par un non-informaticien

(comme pour les dictionnaires et les fichiers des paradigmes flexionnels et dérivationnels), soit sous forme de graphes facilement accessibles et compréhensibles qui peuvent aisément être contrôlés et modifiés sans avoir à maîtriser l'ensemble du programme (comme pour les grammaires flexionnelles, morphologiques et syntaxiques).

- **Moteur linguistique robuste :**
  Etant donné le nombre étendu de langues prises en charge dans NooJ, son moteur linguistique est assez robuste. En effet, analyser des textes écrits en français, allemand, thaïlandais, chinois et arabe exige divers algorithmes d'analyse pour traiter :
  - ✓ L'absence d'accentuation pour les lettres majuscules en français ;
  - ✓ L'absence de vocalisations dans les textes arabes standards ;
  - ✓ La taille importante de l'alphabet chinois ;
  - ✓ Le nombre potentiellement illimité des mot-composés dans les mots en allemand ;
  - ✓ Etc.

  De façon similaire, presque chaque langue a un système de tri différent ; par exemple, en espagnol, les lettres doubles telle que 'll' sont considérées comme une seule lettre quand il s'agit d'effectuer un tri dans l'ordre lexicographique. Malgré toutes ses spécificités et divergences, NooJ ne propose à ses utilisateurs qu'une seule et unique application quelque soit le traitement envisagé.

- **Traitement de corpus :**
  NooJ peut traiter directement des ensembles potentiellement importants de documents. Ces documents peuvent être codés dans plus d'une centaine de formats : tous les formats des types ASCII, EBCDIC, Unicode (UTF-8, UTF-16, etc.), ainsi que les formats standard tels que HTML (format par défaut des textes disponibles en ligne), XML (format de bases de données textuelles), PDF (format de documents portables), RTF (format de textes enrichis), Microsoft WORD, etc.

- **Construction, édition et gestion de concordances sophistiquées :**
  NooJ permet de construire et de gérer des concordances sophistiquées. Il permet de combiner plusieurs requêtes, filtrer manuellement les résultats incorrects et de sauvegarder la concordance résultante. Les outils proposés pour la gestion des concordances permettent d'aborder certains problèmes linguistiques de façon incrémentale.
  Par exemple, il est envisageable de commencer son travail avec NooJ par l'identification de certains termes ou expressions dans un corpus, tout d'abord de façon naïve (par exemple, rechercher tous les mots finissant par –ment pour identifier des adverbes, ou rechercher tous les nombres pour identifier des dates). Puis, au vu de la concordance résultante, il est possible de raffiner progressivement la caractérisation des phénomènes, d'une part en améliorant les requêtes (par ex. en tenant compte du contexte des mots), d'autre part en classifiant les résultats obtenus (par ex. pour distinguer les adverbes en –ment, par ex. « régulièrement », des noms en « –ment », par ex. « investissement »). Les résultats des requêtes peuvent être facilement contrôlés.

- **Annotation interactive de corpus :**
  NooJ permet d'appliquer des grammaires représentées sous forme de graphes ou d'expressions rationnelles augmentées à un corpus. Le résultat est affiché dans une table de concordances où la colonne du milieu contient le mot ou la séquence de mots reconnus. L'utilisateur peut valider ou éliminer chaque entrée de cette concordance. La concordance ainsi filtrée peut, ensuite, être utilisée pour annoter le corpus.
  A ce propos, nous notons que l'annotation d'un corpus à partir d'une concordance permet aux utilisateurs de vérifier, adapter et corriger les résultats d'un traitement automatique dont les

sorties ne sont, pratiquement, jamais parfaites. Ceci est très important surtout lors d'une étape de préparation d'un corpus.

3.6 Tableau comparatif des analyseurs morphologiques

	Technologie à états finis	Génération	Voyellation	Agglutination	Désambiguisation
David Cohen	×	✓	×	×	×
AraMorph	×	✓	✓	±	×
Xerox	✓	✓	✓	±	×
AraParse	×	×	✓	✓	±
AMSAAR	×	×	✓	✓	×
Sebawai	×	±	±	±	×
NooJ	✓	✓	✓	✓	✓

Tableau 15 : Tableau comparatif des analyseurs morphologiques

Légende :

- ✓ : désigne une parfaite prise en charge de l'application ;
- × : désigne une non prise en charge de l'application ;
- ± : désigne une partielle prise en charge ou que celle-ci présente quelques insuffisances.

A ce niveau de la description de notre travail, nous notons que la dernière ligne (c'est-à-dire celle qui correspondant à NooJ) désigne les applications que nous développerons avec la plateforme NooJ. Celles-ci seront détaillées tout au long de ce mémoire.

3.7 Conclusion

Cet état de l'art ainsi que la description des problèmes morpho-syntaxiques spécifiques à la langue arabe soulèvent certaines interrogations quant à l'approche de représentation formelle des données et la stratégie d'analyse les plus adéquates pour garantir la robustesse et la rapidité des traitements. Compte tenu des avantages qu'il présente, nous adoptons NooJ comme plateforme linguistique de développement pour la formalisation, la description et l'analyse de la langue arabe. Dans la suite de ce mémoire, nous détaillerons notre modèle de représentation des données pour la formalisation du vocabulaire aussi bien que les approches préconisées pour la résolution des problèmes déjà décrits. Partant du constat que toute analyse linguistique doit passer par une première phase d'analyse lexicale qui consiste à tester l'appartenance de chaque mot du texte au lexique de la langue, nous entamons notre travail sur la langue arabe par une phase de formalisation de son lexique. Nous commençons par une description détaillée des macrostructures et microstructures des dictionnaires électroniques de stockage de ce lexique.

Chapitre 4

Formalisation et réalisation du dictionnaire arabe : « El-DicAr »

4.1 Introduction

Les dictionnaires électroniques sont, avant tout, construits pour être utilisé par des programmes informatiques d'analyse linguistique. Ils doivent être aussi complets que possible pour réduire au maximum les situations d'échec. Généralement, les dictionnaires électroniques contiennent suffisamment d'informations qui sont rattachées aux entrées lexicales et utilisables pour les différentes étapes d'analyse.

La reconnaissance des mots nécessite une analyse morphologique de façon à retrouver la forme de base à partir d'une forme fléchie. Ce passage d'une forme fléchie (telle qu'elle est dans le texte) à sa forme de base (telle qu'elle est listée dans un dictionnaire) n'est pas évident et difficilement maîtrisable pour une langue comme l'arabe. Traditionnellement, ce passage peut se faire de deux façons équivalentes :

- soit par un *algorithme* qui *examine* les affixes potentiels de la forme fléchie, *construit* les segments potentiels, puis *vérifie* la validité de chaque composant par consultation du lexique ;
- soit en construisant un dictionnaire de *lemmes* ; ensuite, *générer automatiquement* toutes les formes fléchies qui s'y rattachent.

La première méthode permet de travailler avec une liste de formes de base (lemmes ou lexèmes) dont la taille est relativement réduite (1 ou 2 Mo lorsque la liste est non compactée). Toutefois, elle nécessite de nombreux calculs pour analyser chaque mot du texte. Notons que la présence de nombreuses exceptions dans la langue arabe multiplie le nombre d'hypothèses à vérifier, et par conséquent le nombre de contraintes de validation. Par exemple, tous les mots qui se terminent par "ة" devront être testés soit comme des mots invariables, soit des formes féminines d'un adjectif ou d'un nom. Cette approche est utilisée au niveau des analyses traditionnelles de la langue arabe (l'analyse à base de racines et schèmes et celle à base de lexèmes). Les hypothèses de reconstitution des formes fléchies sont énoncées sous forme de bases de règles ou de matrices de compatibilité, généralement, très compliquées à maintenir.

En revanche, la seconde approche nécessite un dictionnaire de formes fléchies préalablement engendrées par génération automatique. Des descriptions de règles flexionnelles et dérivationnelles sont associées aux différentes entrées du dictionnaire (noms, adjectifs, verbes, etc.). Ces paradigmes flexionnels et dérivationnels sont écrits à l'aide d'un ensemble d'opérateurs morphologiques permettant, à partir des entrées du dictionnaire, d'engendrer toutes les formes fléchies potentielles. Dans le dictionnaire généré, chaque entrée est une forme fléchie associée à sa forme de base (lemme), ainsi que l'ensemble des informations qui s'y rattachent tels que le genre, le nombre, le temps, la personne, etc.

La mise en œuvre des dictionnaires électroniques dans NooJ se base sur quatre parties essentielles :

- *une liste d'entrées lexicales* : la transcription de cette liste sera détaillée dans la section suivante ;
- *une liste de propriétés* : ces codes désignent l'ensemble des informations flexionnelles, morphologiques, syntaxiques et sémantiques convenablement choisies par le concepteur pour être rattachées à aux différentes formes du lexique ;
- *des descriptions formelles des règles de flexion et dérivation automatique des lemmes* : ces paradigmes sont associés de façon univoque à des codes alphanumériques permettant de les désigner ;
- *des opérateurs de transformations morphologiques et phonologiques* : ils sont employés par les routines de flexion et de dérivation automatiques, élaborées sur la base des algorithmes de flexion. Nous pouvons distinguer aussi bien des transformations génériques que des transformations spécifiques à la langue du dictionnaire.

Ces parties ne sont toutefois pas indépendantes, mais, comme le montre la Figure 18 ci-dessous, sont en stricte relation même si elles prévoient des phases séparées de planification.

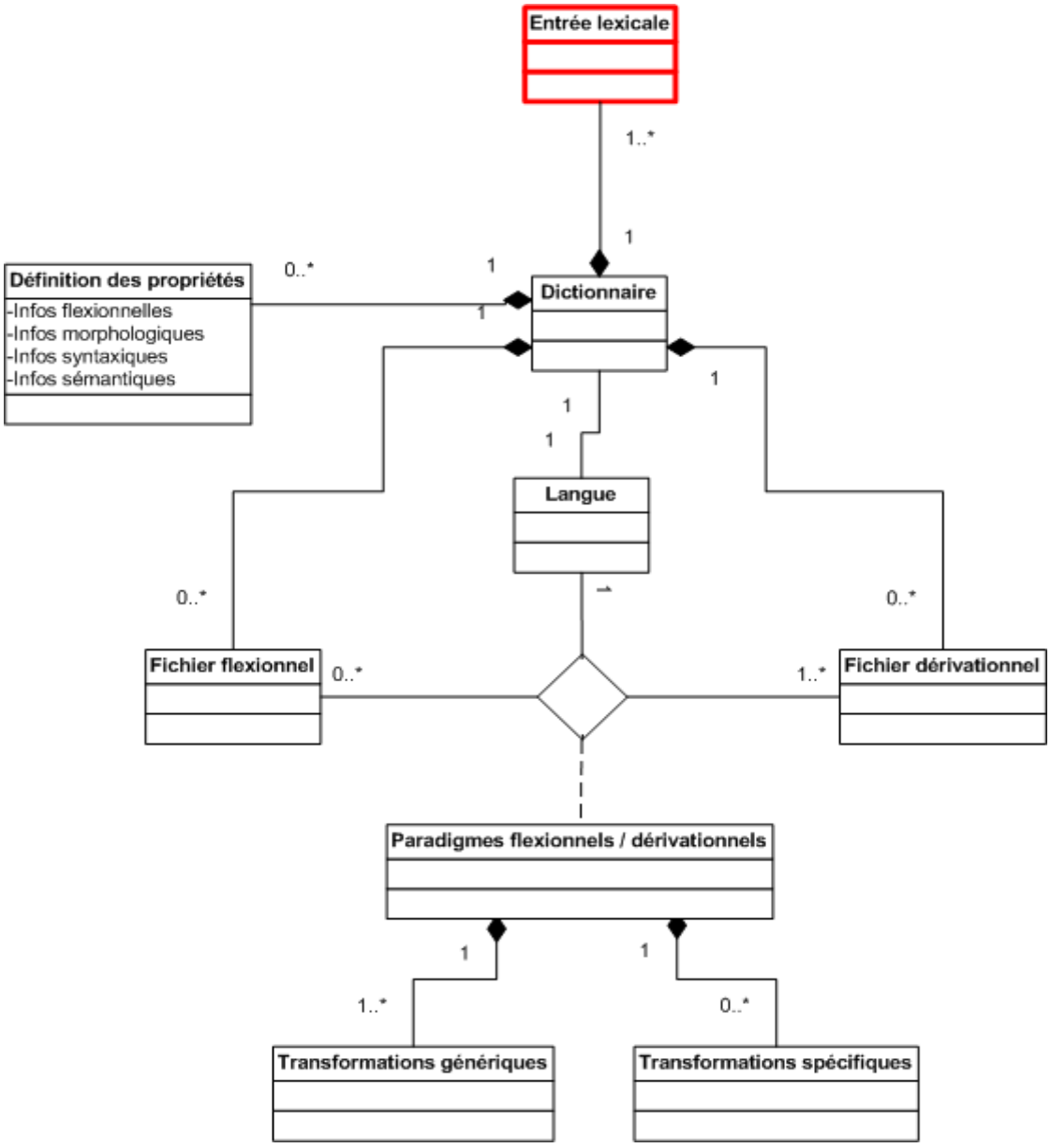


Figure 18 : Macrostructure des dictionnaires électroniques dans NooJ

4.2 Modèle morphologique de base

La formalisation de la liste des entrées lexicales sera le premier pas vers la création de notre dictionnaire électronique qui en contient au moins une. Le modèle morphologique des entrées forme la microstructure du dictionnaire. Celle-ci peut être vue comme une structure composée d'un ensemble d'informations linguistiques. Sur le plan pratique, comme décrit sur la Figure 19, toute entrée lexicale dans NooJ est constituée d'un ensemble de données, nous citons :

- un lemme : considéré comme forme de base ;
- une étiquette : qui en indiquera la catégorie grammaticale d'appartenance ;
- une liste optionnelle d'informations syntactico-sémantiques ;
- une liste éventuelle de codes alphanumériques désignant les modèles flexionnels et dérivationnels à y appliquer.

En dépit de la présence indispensable des deux premières parties, nous signalons que l'apparition des deux dernières dépend des propriétés de l'entrée lexicale. Par exemple, une préposition n'est ni fléchissable, ni dérivable.

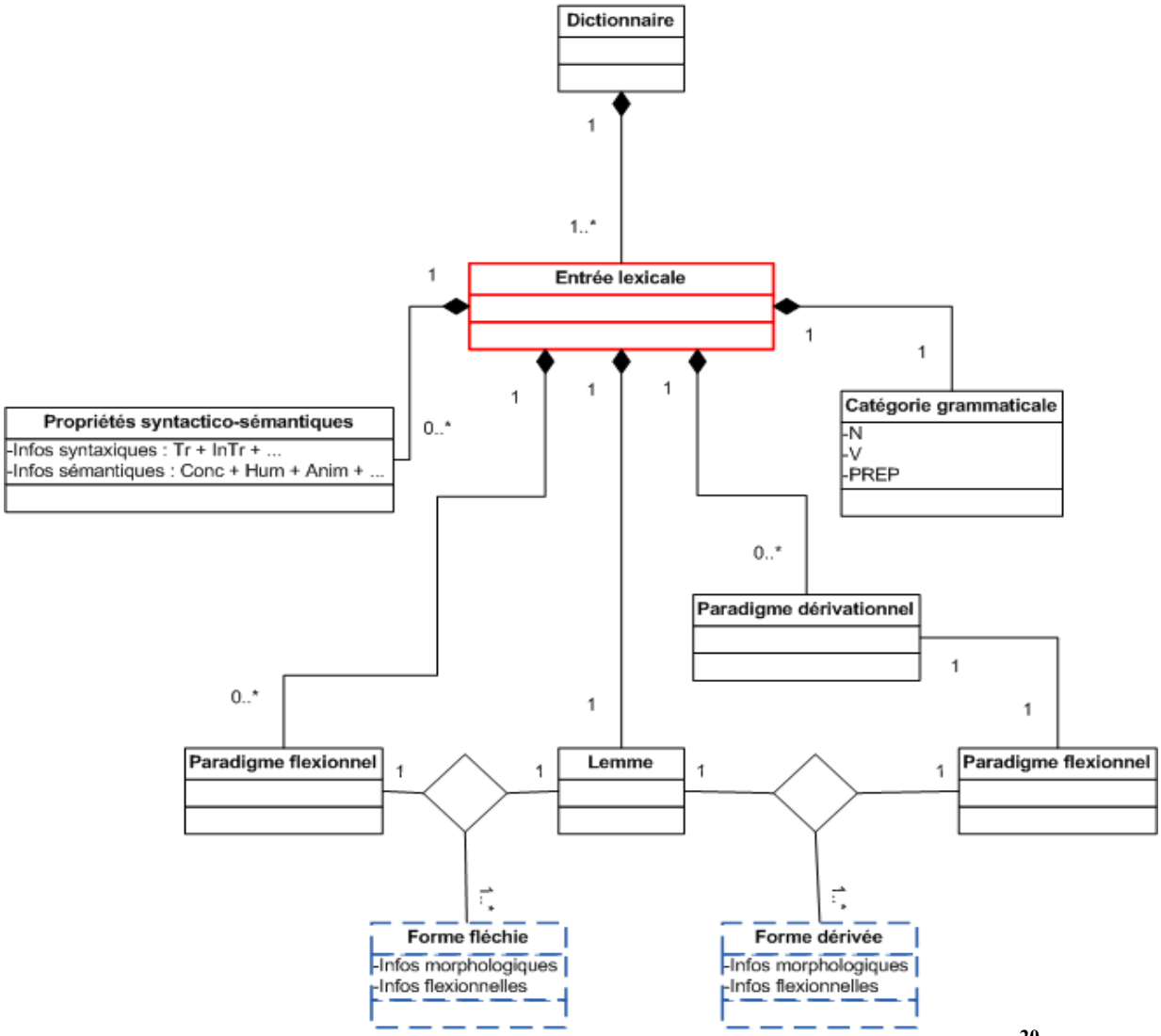


Figure 19 : Microstructure des dictionnaires électroniques dans NooJ [20]

En se basant sur ce modèle de représentation, nous voyons que le module lexical de NooJ permet de recenser et de décrire le lexique courant d'une langue, du point de vue morphologique, syntaxique, voire sémantique :

- **Le niveau morphologique :**
  Le niveau morphologique décrit les relations entre les mots et leurs variations de forme : la conjugaison des verbes, la déclinaison des noms et des adjectifs (au féminin, duel et pluriel) ainsi que la morphologie dérivationnelle (génération de la liste des déverbaux). Par exemple, un analyseur morphologique doit être capable d'interpréter que la forme "ذَهَبَ" (*dahaba*), par exemple, peut représenter le nom "ذَهَب" (*dahaba* – or) indéterminé, décliné à l'accusatif ou le verbe "ذَهَبَ" (*dahaba* – aller) conjugué à la 3ème personne du singulier à l'accompli actif.

[20] Les éléments représentés en pointillés sont générés automatiquement par les routines de flexion et de dérivation dans NooJ

En effet, chaque entrée lexicale introduite au moyen d'un lemme est associée à l'ensemble de ses paradigmes flexionnels et dérivationnels potentiels afin de générer, automatiquement, toutes les formes fléchies qui y sont rattachés. A ce propos, nous notons que les entrées lexicales ne peuvent être associées qu'à un seul modèle flexionnel ; par contre, elles peuvent être associées à autant de paradigmes dérivationnels que nécessaire.

- **Le niveau syntaxique :**
  Le niveau syntaxique décrit les propriétés grammaticales des formes à analyser. Par exemple, la description syntaxique des verbes contient, pour chaque verbe, les informations relatives à la transitivité, la possibilité de se mettre au passif, etc. Nous pouvons aussi garnir chacune de ces entrées verbales par le type de ses compléments ou les prépositions qui les introduisent. Ces entrées associées à leurs propriétés syntaxiques peuvent constituer un lexique–grammaire. Ainsi, le dictionnaire peut prendre la forme d'une table où chaque ligne représente une entrée lexicale (voir Annexe D). Chaque intersection d'une ligne et d'une colonne affiche la valeur de la propriété correspondante (sur la colonne) en se basant sur le fichier de définition des propriétés « Properties.def » (voir Annexe C3).

- **Le niveau sémantique**
  Le niveau sémantique décrit les traits sémantiques et distributionnels des mots pour représenter le sens perceptible par une entrée quelconque. Nous distinguons deux types de traits sémantiques :
  - ✓ les *traits sémantiques* propres à un terme donné : par exemple l'entrée "كُرْسِيّ" (*korsiyy* – siège) peut être associé au trait « concret » ;
  - ✓ les *traits contextuels* qui permettent de restreindre l'emploi d'un terme dans un énoncé : par exemple le verbe "كَتَبَ" (*kataba* – écrire) peut être associé au trait « à sujet humain ».

L'ensemble de ces informations sera d'une extrême importance pour les tâches d'analyse morpho-syntaxique automatique à effectuer ultérieurement.

4.3 Construction du noyau de base du dictionnaire

Le lexique de l'arabe est recensé dans des dictionnaires électroniques dont le modèle a été décrit plus haut. Ces dictionnaires se présentent sous la forme de listes d'associations lemme/ information. A ce niveau, nous convenons de signaler que, contrairement au dictionnaire classique de type DELA utilisé au sein de son prédécesseur INTEX, les entrées des dictionnaires dans NooJ sont codées en Unicode / UTF-8. Nous utilisont, alors, les caractères de l'alphabet arabe pour le codage des entrées lexicales (voir Annexe C).

La construction du noyau de base du dictionnaire électronique comporte une phase préliminaire importante qui correspond à la consultation de sources adéquates au repérage des données à insérer. En général, les sources sont non seulement les dictionnaires en papier, mais aussi tous les autres contextes médiatiques de divulgation qui présentent une crédibilité sociolinguistique suffisante, c'est-à-dire tous ces domaines communicatifs dans lesquels nous utilisons la langue standard d'une façon plus ou moins déclarée. Compte tenu de la diversité et la quantité de sources analysables, nous convenons de se focaliser, dans un premier lieu, sur la création d'une liste de lemmes de base, qui inclut, au moins, les termes les plus fréquemment utilisés dans la langue. Par la suite, cette liste peut être mise à jour en se basant sur les résultats des étapes d'analyse ultérieures.

En commençant la phase de lemmatisation, il est important d'effectuer une distinction préliminaire entre mots fléchis et non fléchis. Il s'agit d'une différenciation formelle très importante ; d'un point de vue morphologique cela nous aidera à définir deux sous-ensembles spécifiques du lexique d'une langue, avec des caractéristiques distinctives qui seront en conséquence formalisées par des algorithmes de flexion également distinctifs.

Ainsi, les entrées flexionnelles de la liste de lemmes seront insérées comme suit :

- la forme conjuguée à la 3ème personne du singulier de l'accompli actif pour les verbes ;
- la forme au masculin singulier pour les noms variables selon le genre ;
- la forme au singulier pour les noms invariant en genre.

En revanche, les entrées non fléchissables seront listées telles qu'elles apparaissent dans le lexique.

De plus, chaque entrée est associée, de façon non ambiguë, à une catégorie grammaticale désignée par un code écrit au moyen de lettres en majuscules. Les codes que nous avons utilisés sont les suivants :

Catégorie	Code	Exemple
Adjectif	ADJ	"سِيَاحِيّ" (*siyaāḥiyy* – touristique)
Nom	N	"كِتَاب" (*kitaāb* – livre)
Verbe	V	"كَتَبَ" (*kataba* – écrire)
Adverbe	ADV	"أُفُقِيًّا " (*ùfuqiyyã* – horizontalement)
Préposition	PREP	"إِلَى" (*ílaą* – jusqu'à)
Pronom personnel	PRON	"نَحْنُ " (*naḥnu* – nous)
Démonstratif	DEM	"هَذِهِ" (*haḏihi* – celle-ci)
Relatif	REL	"الَّذِي" (*alladi* – qui)

Tableau 16 : Liste des catégories grammaticales

En outre, des parties optionnelles sont associées aux entrées de notre dictionnaire. Nous pouvons citons :

- des informations syntactico-sémantiques : la majorité de ces codes a été assignée aux entrées lexicales nominales. Parmi les traits utilisés, nous citons :

Trait syntactico-sémantique	Code	Exemple
Abstrait	Abs	"كِتَابَة" (*kitaābat* – écriture)
Concret	Conc	"كِتَاب" (*kitaāb* – livre)
Animal	Anl	"خَيْل" (*ḫayl* – cheval)
Végétal	Veg	"طماطم" (*tamaāṭim* – tomate)
Humain	Hum	"وَزِير" (*waziyr* – ministre)
Médical	Medic	"دم" (*dam* – sang)
Lieux / Localisation	Loc	"باريس" (*bariys* – Paris)
Date / Heure	Date	"يُونيو" (*yuwnyuw* – janvier)
Organisation	Org	"يُونيسكُو" (*yuwneskuw* – UNESCO)

Tableau 17 : Liste des traits syntactico-sémantiques

- un appel à un paradigme flexionnel introduit par « +FLX= »[21]
- un appel à un ou plusieurs paradigmes dérivationnels introduits par « +DRV= »[22]

[21] FLX : fonctionnalité permettant la description des formes fléchies potentielles à partir d'un lemme.
[22] DRV : fonctionnalité permettant la description des formes dérivées potentielles à partir d'un lemme.

Après application des modèles flexionnels et dérivationnels, chaque forme générée est associée automatiquement à l'ensemble des informations flexionnelles correspondantes. Ces codes sont listés dans le Tableau 18, ci-dessous :

	Code	Signification
Codes exclusifs pour les verbes	P	Inaccompli - Présent de l'indicatif
	I	Accompli – Passé
	S	Subjonctif
	C	Apocopé
	F	Futur
	Y	Impératif
	A	Voix active
	K	Voix passive
Codes exclusifs pour les noms et adjectifs	a	Accusatif
	u	Nominatif
	i	Génitif
	an	Tanwîn[23], Accusatif
	un	Tanwîn, Nominatif
	in	Tanwîn, Génitif
Codes communs aux verbes, noms et adjectifs	1, 2, 3	1ére , 2ème ou 3ème Personne
	M, f	masculin, féminin
	s, d, p	singulier, duel, pluriel

Tableau 18 : Liste des codes flexionnels

Dans ce qui suit, nous reprenons la classification décrite à la section 2.2 : les verbes, les noms, les pronoms et les mots outils.

4.3.1 Les verbes

Le dictionnaire des verbes contient 10 500 entrées complètement voyellées. Une étude approfondie nous a permis de décrire l'organisation d'ensemble du système verbal et d'en proposer une classification. Cette classification a permis d'associer chaque verbe, ramené à la 3ème personne du singulier à l'accompli actif, à un modèle de flexion (parmi 130 modèles développés pour la totalité des verbes) et un ou plusieurs modèles dérivationnels (parmi 125 modèles développés pour la totalité des verbes) [Abou Il Azm, 2003] et [Bescherelle, 1999].

Généralement, les entrées verbales ressemblent à :

دَرَّسَ, V+Tr+FLX=V_darrasa+DRV=D_darrasa:FLXDRV (*darrasa* - enseigner)

Outre le symbole désignant la catégorie grammaticale ('V' pour les verbes), à chaque entrée verbale, nous fournissons l'information liée à sa transitivité. Nous avons distingué trois différents cas :

- Les verbes transitifs (+Tr) : Ce sont des verbes qui peuvent admettre un complément d'objet direct ;

[23] Le Tanwîn est une marque d'indéfinitude.

- Les verbes transitifs indirects (+TrInd) : Ce sont des verbes qui peuvent admettre un complément d'objet précédé d'une préposition (par exemple, le verbe "ضَحِكَ مِنْ" (*ḍaḥika min-* rire de)) ;
- Les verbes intransitifs (+InTr) : Ce sont des verbes qui n'admettent ni complément d'objet direct, ni indirect (par exemple, le verbe "هَلَكَ" (*halaka* – mourir)).

En outre, nous associons chaque entrée à des paradigmes de conjugaison et dérivation. En l'occurrence, pour l'exemple du verbe "دَرَّسَ" (*darrasa* - enseigner) nous avons :

- L'expression "+FLX=V_darrasa" qui désigne un appel au paradigme flexionnel "V_darrasa" à utiliser pour fléchir le lemme en entrée. Par paradigme flexionnel nous désignons l'ensemble des transformations permettant de générer automatiquement, à partir de l'entrée lexicale, l'ensemble de ses formes conjuguées. Ces paradigmes flexionnels incluent le mode (accompli, inaccompli, subjonctif, apocopé et impératif), la voix (active et passive), le genre et le nombre. En moyenne, pour chaque entrée verbale, u paradigme flexionnel permet de générer 122 formes fléchies.

- L'expression "+DRV= D_darrasa:FLXDRV" qui désigne un appel au modèle de dérivation "D_darrasa" à utiliser pour produire les deux seuls déverbaux dont la production est régulière : "اسم الفاعل" (*ísm al-faā'il* - participe actif) et "اسم المفعول" (*ísm al-maf'uwl* - participe passif). Ce modèle de dérivation fait usage des règles de flexions introduites par ":FLXDRV" pour décliner chaque déverbal généré suivant le genre (masculin ou féminin), le nombre (singulier, duel ou pluriel), le mode (déterminé, indéterminé), le cas (génitif, accusatif ou nominatif).

Une description détaillée des paradigmes flexionnels et dérivationnels sera donnée plus loin.

Notre étude sur les verbes a révélé une complexité non redoutable du système verbal en arabe. Une telle complexité a été exhibée par l'ensemble des irrégularités dégagées lors de cette étude. Nous commençons par le cas de racines "faibles", qui contiennent l'une des lettres "و" (*w*), "ي" (*y*), *Hamza* ("إ" ,"أ") (*á* ou *í*) ou *Alif* "ا" (*ā*), qui introduisent certaines irrégularités flexionnelles et suivent, souvent, plus qu'un modèle de conjugaison. En l'occurrence, le verbe "وَأَبَ" (*waáaba* – être en colère / être embarassé) peut être associée à deux paradigmes flexionnels pour donner les deux formes possibles:

- "يَوْأَبُ" (*yawáabu* – est en colère) avec une simple concaténation de "يَ" (*ya*) à la tête du verbe ;
- "يَئِبُ" (*yaýibu* – est embarassé, gêné) avec la concaténation de "يَ" (*ya*) à la tête du verbe, la suppression de "وَ" (*wa*) et, enfin, substituer le support de la hamza "أ" (*á*) en "ـئـ" (*ý*).

D'autres irrégularités au niveau de la conjugaison ont été remarquées. Elles se présentent, généralement, sous la forme d'un rejet de certaines formes de conjugaisons, notamment l'impératif et la voix passive. Ces irrégularités sont au nombre de quatre :

- **Les verbes qui n'acceptent pas de conjugaison à l'impératif**, tels que : "حَسُنَ" (*ḥasuna* – être beau), "رَؤُفَ" (*raẃufa* – être clément), "يَمُنَ" (*yamuna* – être béni par la chance), "سَرُوَ" (*saruwa* – être noble). Généralement, ces verbes expriment un état ou une qualité durables. Par exemple, pour dire à quelqu'un « Sois beau ! », nous ne pouvons pas recourir à l'impératif du verbe"حَسُنَ" (*ḥasuna* – être beau) ; nous pouvons, cependant, faire usage du verbe كَانَ (*kaāna* – être) à l'impératif "كُنْ" (*kun* – sois !), suivi de l'adjectif correspondant : كُنْ حَسَنًا (*kun ḥasanã* - Sois beau !) ;

- **Les verbes qui n'acceptent aucune conjugaison à la voix passive**. Ce sont les mêmes verbes qui n'acceptent pas l'impératif, cités ci-dessus. En effet, ces verbes qui, rappelons-le, décrivent un état ou qualité durables, ont un sens qui exclut le passif. Nous notons que, théoriquement, pour certains verbes de qualité, le passif peut avoir une valeur impersonnelle qui serait acceptable malgré la rareté de son apparition dans les textes courants. En addition, les sens de

ces verbes excluent les participes actifs et les participes passifs. Toutefois, la liste est étendue à certains verbes acceptant des constructions impersonnelles tels que le verbe "وَجَبَ" (*wağaba* – falloir, devoir) dans la phrase "يَجِبُ أنْ أذهَبَ" (*yağibu áan áaḏhaba* - il faut que je m'en aille) et le verbe "أمْكَنَ" (*ámkana* – être possible) dans la phrase "يُمْكِنُنِي الآنَ أنْ أذهَبَ" (*yumkinuniy al-ân áan áaḏhaba* - il m'est possible maintenant de s'en aller). Ces verbes n'ont pas d'impératif et ne se conjuguent pas au passif, tout comme les verbes devoir et pouvoir en français ;

- **Les verbes qui ne se conjuguent au passif qu'à la forme impersonnelle** (3ème personne, masculin, singulier : +3+m+s) : Ces verbes sont intransitifs ou transitifs indirects. La forme du passif impersonnel est suivie d'une préposition. Cette préposition précède, soit un complément d'objet indirect, soit un complément circonstanciel (le plus souvent de lieu ou de temps). Par exemple :
  - ✓ Le verbe *transitif indirect* "ضَحِكَ" (*ḍaḥika min* - rire de qqchose ou qqu'un) peut être utilisé dans la phrase : "أمورٌ لا يُضْحَكُ مِنْهَا" (*ùmuwrũ laā yuḍḥaku minhaā* - des choses dont on ne rit pas) ;
  - ✓ Le verbe *intransitif* "نَامَ" (*naāma* - dormir) peut être accepté dans : "دَارٌ لا يُنَامُ فِيهَا" (*daārũ laā yunaāmu fiyhaā* - une maison dans laquelle on ne peut dormir).

  Dans ces deux cas, nous soulignons que le passif acquiert une valeur modale : déontique dans le premier exemple et potentielle dans le second.

  Parmi les cas recensés, nous citons : "وَلِعَ" (*wali'a* – éprouver une passion pour), "أسِيَ" (*áasiya* - être triste), "وَئِبَ" (*waýiba* – se mettre en colère), "وَبِئَ" (*wabiýa* – être frappé par une épidémie), "وَجِيَ" (*wağiya* – avoir les pieds usés), "اِنْصَاعَ" (*ìinsaā'a* – obéir), "اِنْتَلَى" (*ìiýtalaā* – jurer), "تَكَأكَأَ" (*takaákaáa* – ne pas pouvoir parler), etc. ;

- **Les verbes qui ne se conjuguent au passif qu'à la troisième personne du singulier et du duel** : Ce sont les verbes transitifs dont le sens exclut les compléments d'objet humains, nous disons que le verbe est "مُتَعَدٍّ إلى غَير العَاقِل" (*muta'addi ílą ġayr al-'aāqil* - transitif à des non-humains). Ceci entraîne deux conséquences :
  - ✓ Les 1ères et 2èmes personnes, ainsi que les formes des pluriels masculin et féminin, qui sont réservées en arabe à des humains, sont exclues[24] ;
  - ✓ Ces verbes ne conjuguent qu'au passif à la 3ème personne, masculin et féminin, au singulier ou au duel.

  Parmi ces verbes, nous citons : "قُرِأَ" (*qaraáa* – lire) et "وَبَأَ" (*wabaáa* – préparer des bagages).

Quant aux irrégularités au niveau de la dérivation des participes actifs et passifs, nous avons énuméré deux types d'anomalies :

- **Les verbes qui n'acceptent pas de participe actif**, tels que : "سَئِمَ" (*sayima* – s'ennuyer), "وَجِلَ" (*wağala* - craindre, redouter), "يَقِنَ" (*yaqina* – être sûr), "حَيَّ" (*ḥayya* – être vivant), "وَجِيَ" (*wağiya* – avoir les pieds usés), etc. ;

- **Les verbes qui n'acceptent aucun des deux participes**, tels que : "أمُرَ" (*áamura* - devenir prince), "شَرَّ" (*šarra* - être méchant), "رَؤُفَ" (*raùufa* – être clément), "جَرُؤَ" (*ğaruùa* - oser), "يَمُنَ" (*yamuna* – être béni par la chance), "جَيِدَ" (*ğayida* – être bien), "سَرُوَ" (*saruwa* – être noble), "أمُوَ" (*áamuwa* – être esclave), "وَبِئَ"(*wabiáa* – être frappé par une épidémie), "وَضُؤَ" (*waḍuẃa* – procéder à des ablutions, être en état de pureté légale), etc.

[24] En arabe, le pluriel des non-humains est construit avec la forme du féminin singulier.

4.3.2 Les noms

Le recensement des lemmes nominaux s'est basé sur l'élaboration de cinq listes :

1) Nous avons construit un dictionnaire qui contient environ 6 204 entrées sous forme de noms primitifs[25] (par exemple, le nom " كُرْسِيّ " (*korsiyy* - siège)). Ces entrées lexicales sont listées dans le dictionnaire comme suit :
   - ✓ la forme au masculin singulier pour les noms variables selon le genre telle que : إِبْن,N+FLX=Flexion1+DRV=Abnaa:FlexionPL (*ìbn* – fils) ;
   - ✓ la forme au singulier pour les noms invariant en genre telle que : كُرْسِيّ,N+FLX=Flexion2+DRV=ArAdhi50:FlexionPL2 (*korsiyy* - siège)
2) Nous avons construit, semi-automatiquement, une liste de 11 300 déverbaux autres que les participes actif et passif. Apartir de l'association de classes dérivationnelles à la liste de tous les verbes, nous avons généré la liste de tous les noms verbaux ou *Masdars*. En fait, contrairement aux participes actif et passif, nous avons renoncé à la génération automatique des *Masdars* à partir des formes simples à cause des irrégularités multiples qu'ils présentent. D'abord, cette liste a été générée automatiquement. Ensuite, elle a été revue manuellement pour tenir compte des diverses anomalies. Notre étude a révélé que la majorité des cas particuliers est recensée du côté des verbes trilitères. Cependant, les verbes augmentés ont présenté une régularité remarquable.

   Par exemple, bien que les deux verbes "كَتَبَ" (*kataba* - écrire) et "ذكَرَ" (*dakara* - citer) se conjuguent de la même façon et nécessitent les mêmes règles pour dériver les participes actif et passif, les noms verbaux qui y sont rattachés ne présentent aucune similarité. En effet :
   - ✓ Pour le verbe "كَتَبَ" (*kataba* – écrire), nous avons recensé les 3 noms verbaux : "كَتْب" (*katb* – un écrit) de schème "فَعْل" (*fa'l*), "كِتَاب" (*kitaāb* – un livre) de schème "فِعَال" (*fi'aāl*) et "كِتَابَة" (*kitaābatat* – une écriture) de schème "فِعَالَة" (*fi'aālat*);
   - ✓ Pour le verbe "ذَكَرَ" (*dakara* – rappeler, mentionner), nous avons recensé les 3 noms verbaux : "ذِكْر" (*dikr* – le fait de rappeler ou mentionner quelque chose) de schème "فِعْل" (*fi'l*), "ذِكْرَى" (*dikraą* – commémoration, hommage) de schème "فِعْلَى" (*fi'laą*) et "تَذْكَار" (*taḏkaār* – souvenir) de schème "تَفْعَال" (*taf'aāl*).

   Les entrées ainsi introduites s'écrivent sous la forme :

   كِـتَـاب,N+FLX=KitAb+DRV=Koutoubon:FlexionPL

3) Nous avons introduit dans le même dictionnaire quelques mots au pluriel qui n'ont pas de correspondant singulier utilisé ; par exemple, le mot " مَخَاوِف " (*maḫaāwif* – dangers, périls). Pour ces mots, nous n'avons besoin que d'une flexion simple du lemme à l'aide de « +FLX= ». Par exemple, nous pouvons citer :

   مَـخَـا وِف,N+p+FLX=FlexionPL
   تَـضَـا رِيـس,N+p+FLX=FlexionPL

4) Nous avons recensé quelques centaines de mots composés, environ 720 entrées lexicales. Nous distinguons deux catégories principales de mots composés :
   - ✓ ceux formés par association entre deux mots : "عيد الميلاد" (*'iyd al-miylaād* – anniversaire), "قوس قزح" (*qaws qouzaḥ* – arc en ciel), "وزير الداخلية" (*waziyr al-daāḫiliyyat* – ministre de l'intérieur) ;
   - ✓ ceux formés par fusion entre deux mots : c'est surtout le cas des mots (noms ou adjectifs) composés avec la négation "لا" (*laā* – in-, non). Nous pouvons citer : "اللانِهَايَة" (*al-*

[25] Un nom primitif désigne un nom qui ne peut pas être dérivé d'un verbe.

laānihaāyat – l'infini), "اللاشَيْء" (*al-laāšay'* – le néant), "لادِينِيّ" (*laādiyniyy* – irréligieux, athée), etc.

5) Enfin, pour répondre aux besoins du module de reconnaissance d'entités nommées qui sera décrit dans la section 6.7, nous avons construit plusieurs listes de noms propres :
   - ✓ Une liste de 12 400 prénoms : cette liste est constituée de prénoms arabes simples (tel que "أَحْمَد" (*áḥmad* - Ahmed)) et composés (tels que "زين الدِّين" (*zin al-diyn* - Zineddine) ou "عبد الكَرِيم" (*'bd al-karim* - Abdel Karim)), ainsi que des prénoms étrangers transcrits (tels que "جُون" (*ğuwn* - John)) ;
   - ✓ Une liste de 5 038 noms de lieux contenant des noms de pays tels que "تُونس" (Tūnus - la Tunisie) ou "فِرَنْسَا" (*fìransaā* - la France), des noms de villes ayant plus de 100 000 habitants tels que "بَارِيس" (*bariys* - Paris) ou "بوينس آيرس " (*biwiynis âyris* - Buenos Aires), etc.
   - ✓ Une liste de 843 organisations telle que "منظّمة الأمم المتّحدة" (*munaẓẓamat al-ùmam al-muttaḥidat* – Organisation des Nations Unies, ONU).

L'association des paradigmes flexionnels et dérivationnels aux différentes entrées nominales a fait l'objet d'une étude du système nominal arabe, cf. Section 2.3.2. En effet, dans la majorité des cas, nous avons associé chaque entrée nominale à :

- **Une classe flexionnelle** introduite par l'intermédiaire du code spécial « +FLX= ». Elle calcule les déclinaisons de chaque lemme aux différents cas possibles (nominatif, accusatif ou génitif), ainsi qu'aux cas définis et indéfinis. Généralement, cette déclinaison se base sur des suffixations des voyelles casuelles.

- **Une ou plusieurs classes dérivationnelles** appelées pour reconnaître les formes dueles et plurielles. Notons que, le calcul des formes plurielles a nécessité le développement de 125 paradigmes dérivationnels pour la description du masculin pluriel régulier, féminin pluriel régulier et le pluriel brisé. Ces modèles ont été minutieusement mis au point afin de traiter certaines spécificités du système nominal de la langue arabe comme la différence entre les pluriels de petits nombres et collectifs pluriels tels que le nom " نَفْس " (*nafs* – âme) qui admet deux formes plurielles : " أَنْفُس " (*ánfus* - quelques âmes) et " نُفُوس " (*nufuws* - un grand nombre d'âmes).

4.3.3 Les pronoms et les mots outils

En raison de leurs nombres réduits et de la grande fréquence d'utilisation de ces mots, nous les stockons dans une liste fermée.

Pour les pronoms, nous avons dénombré les ensembles suivants :
- 14 pronoms démonstratifs ;
- 9 pronoms relatifs ;
- 13 pronoms personnels ;

Quant aux mots outils, nous avons répertorié :
- environ 720 mots outils simples incluant les prépositions, adverbes, conjonctions, interjections et les outils d'exceptions, de négation, etc. ;
- 613 mots outils composés comprenant majoritairement :
   - ✓ des locutions adverbiales telles que : "بطريقة مختلفة" (*biṭariyqatin muẖtalifatin* – différemment), "بِشَكْلٍ غَيْرَ قَانُونِيٍّ" (*bišaklĩ ġayr qaānuwniyyĩ* – illégalement), etc.
   - ✓ des groupes prépositionnels tels que : "عَلَى مُسْتَوَى" (*'alaq mustawaq* – au niveau de), "نَظَرًا إِلَى" (*naẓarã ílaq* – en raison de, au vu de), etc.

Les entrées de la liste de lemmes ainsi structurées et équipées des informations nécessaires forment ce que nous convenons d'appeler dans la suite de ce mémoire **El-DicAr** (**El**ectronic

Dictionary for **Ar**abic). Par application des routines de flexion et de dérivation automatiques, nous générons, à partir de ce dictionnaire, la liste de toutes les formes fléchies potentielles. Les entrées de cette liste, comme nous le verrons dans la section suivante, sont accompagnées, en plus du lemme et de la catégorie grammaticale, d'un ensemble d'étiquettes relatives aux temps, mode, personne, genre, nombre, etc. (voir Tableau 18). La liste ainsi générée sera stockée dans des transducteurs à états finis. Ces transducteurs sont minimisés et compressés à l'aide des algorithmes décrits dans le premier chapitre.

Les machines à états finis ainsi construites auront comme principale application la reconnaissance automatique des mots simples et composés dans les textes. En effet, elles permettent de reconnaître, avec un pourcentage de rappel de 100%, les formes fléchies telles qu'elles apparaissent dans les textes, c'est-à-dire que, généralement :

- les noms et les adjectifs apparaissent sous leurs formes déclinées ;
- les verbes apparaissent conjugués.

4.4 Génération automatique des formes fléchies

Lors de cette étape, nous générons automatiquement toutes les formes fléchies potentielles au moyen d'appels aux paradigmes flexionnels et dérivationnels préalablement assignés à chaque entrée lexicale. La méthode que nous illustrons a le but de décrire exhaustivement le système flexionnel de l'arabe. Notre approche doit, donc, tenir compte de chaque particularité, pour arriver à formaliser une quantité suffisante de descriptions formalisées. Ces descriptions se fondent sur un exposé de l'ensemble des transformations morphologiques et phonologiques nécessaires pour le passage de la forme de base (le lemme) à toutes les formes fléchies qui s'y rattachent.

4.4.1 Opérateurs morphologiques génériques

NooJ dispose de routines et d'algorithmes permettant la genèse de l'ensemble des formes fléchies en s'appuyant sur un appel aux descriptions flexionnelles et dérivationnelles attribuées à chaque entrée du dictionnaire. Ces descriptions peuvent être décrites sous la forme d'expressions rationnelles ou de graphes établies à l'aide de l'éditeur graphique de NooJ. Elles reposent sur des opérateurs morphologiques effectuant des transformations au sein des lemmes en entrée. Ces transformations sont basées sur l'utilisation de certaines commandes génériques prédéfinies :

- <L>: déplacement vers la gauche (Left arrow),
- <R>: déplacement vers la droite (Right arrow),
- <B>: suppression du dernier caractère (Backspace),
- <S>: suppression du caractère courant (Suppr),
- <N> : déplacement vers le mot suivant (Next word form),
- <P> : déplacement vers le mot précédent (Previous word form),
- <D> : duplication du caractère courant (Duplicate)
- <E>: chaîne vide (Empty string)

Ces commandes peuvent être associées à deux types d'argument :

- un nombre : par exemple :
  - ✓ <B2>: suppression des deux derniers caractères
  - ✓ <L3>: déplacement à gauche, trois fois
  - ✓ <R4>: déplacement à droite, quatre fois
  - ✓ <S5>: suppression des 5 caractères qui suivent
- un «W» : par exemple :
  - ✓ <BW>: suppression à partir de la lettre courante et jusqu'au premier caractère du mot
  - ✓ <LW>: aller au début du mot
  - ✓ <RW>: aller à la fin du mot

- ✓ <SW>: suppression à partir de la lettre courante et jusqu'au dernier caractère du mot

La totalité de ces opérateurs morphologiques fonctionnent sur une pile, ils nécessitent un temps de transformation en $O(n)$. Ainsi, ils garantissent une correspondance entre le lemme et sa forme fléchie en un temps linéaire.

Pour illustrer l'utilisation de ces opérateurs transformationnels, nous considérons l'entrée lexicale :

دَرَّسَ,V+Tr+FLX=V_darrasa+DRV=D_darrasa:FLXDRV (*darrasa* - enseigner)

Cette entrée verbale ramenée à la 3ème personne du singulier à l'accompli actif, دَرَّسَ (*darrasa* – enseigner), est associée à la description flexionnelle "V_darrasa" pour générer l'ensemble des formes conjuguées. Rappelons que, chaque paradigme inclut le mode (indicatif, subjonctif, apocopé et impératif), la voix (active et passive), le genre et le nombre, ce qui produit, en moyenne, 122 formes fléchies par entrée lexicale.

Parmi les 122 transformations flexionnelles qui sont intégrées dans le paradigme flexionnel V_darrasa, en voici une :

<LW>يُ<R4><S>ِ<R><S>ُ/A+P+3+m+s (*<LW>yu<R4><S>i<R><S>u/A+P+3+m+s*)

Cette commande NooJ inclut les transformations suivantes :

- <LW> : positionner le curseur (|), initialement placé à la fin du mot (دَرَّسَ| - *darrasa*|), à la tête du lemme par un déplacement vers la gauche → (|دَرَّسَ - |*darrasa*) ;
- insérer (يُ - *yu*) à la tête de la forme → (يُ|دَرَّسَ - *yu*|*darrasa*) ;
- <R4> : sauter quatre lettres vers la droite → (يُدَرَّ|سَ - *yudarr*|*asa*) ;
- <S> : effacer la lettre suivante → (يُدَرّ|سَ - *yudarr*|*sa*) ;
- insérer la voyelle (ِ - *i*) → (يُدَرِّ|سَ - *yudarri*|*sa*) ;
- <R> : sauter une lettre vers la droite → (يُدَرِّس|َ - *yudarris*|*a*) ;
- <S> : effacer la lettre suivante → (يُدَرِّس| - *yudarris*|) ;
- insérer la voyelle finale (ُ - *u*) → (يُدَرِّسُ| - (*yudarrisu*|).

Cette commande permet de générer la forme suivante : يُدَرِّسُ (*yudarrisu* – il enseigne). Cette forme sera associée aux informations flexionnelles : V+Tr+A+P+3+m+s, c'est-à-dire : verbe transitif direct (V+Tr) conjugué au masculin (m) singulier (s), troisième personne (3), présent de l'indicatif (P), à la voix active (A).

Quant à la dérivation, elle est assurée par le biais d'un appel au paradigme dérivationnel "D_darrasa". Celui-ci inclut les transformations qui permettent de générer :

1) Le participe actif, par le biais de la commande :

<LW> مُ <R4><S> ِ<R><S>/N+PA → "مُـدَرِّس"
<LW>mu<R4><S>i<R><S>/N+PA → (*mudarris* – professeur)

Cette commande inclut les opérations morphologiques suivantes :

- <LW>: positionner le curseur (|) au début du verbe → (|دَرَّسَ - |*darrasa*) ;
- insérer (مُ - *mu*) à la tête de la forme → (مُ|دَرَّسَ - *mu*|*darrasa*) ;
- <R4> : sauter quatre lettres → (مُدَرّ|سَ - *mudarr*|*asa*) ;
- <S> : supprimer la lettre suivante → (مُدَرّ|سَ - *mudarr*|*sa*) ;
- insérer la voyelle (ِ - *i*) → (مُدَرِّ|سَ - *mudarri*|*sa*) ;
- <R> : sauter une lettre → (مُدَرِّس|َ - *mudarris*|*a*) ;
- <S> : supprimer la dernière lettre → (مُدَرِّس| - *mudarris*|/ N+PA).

La forme dérivée ainsi générée : مُدَرِّس (*mudarris* - professeur) est associée au code (N+PA) indiquant qu'elle est nominale (+N), représentant un Participe Actif (+PA). Cette forme est automatiquement associée au paradigme flexionnel « :FLXDRV » pour engendrer toutes les formes fléchies qui s'y rattachent selon le genre (masculin ou féminin), nombre (singulier, duel et pluriel) et la déclinaison casuelle (nominatif, accusatif et génitif).

2) Le participe passif, par le biais de la commande :

<B><LW>مُ /N+PP → "مُـدَ رَّس"
<B><LW>mu/N+PP → (*mudarras* – enseigné, élève)

Cette commande inclut les opérations morphologiques suivantes :

- <B> : supprimer la dernière voyelle → (دَرَّس| - *darras*|) ;
- <LW> : positionner le curseur (|) au début du verbe → (|دَرَّس - |*darras*) ;
- insérer (مُ - *mu*) à la tête de la forme → (مُ|دَرَّس - *mu*|*darras*/N+PP).

La forme dérivée ainsi générée : مُدَرَّس (*mudarras* - élève) est associée au code (N+PP) indiquant qu'elle est nominale (+N), représentant un Participe Passif (+PP). A l'instar du participe actif, cette forme est automatiquement associée au paradigme flexionnel « :FLXDRV » pour engendrer toutes les formes fléchies qui s'y rattachent selon le genre (masculin ou féminin), le nombre (singulier, duel et pluriel) et la déclinaison casuelle (nominatif, accusatif et génitif).

Tous ces opérateurs morphologiques génériques peuvent être redéfinis, d'autres opérateurs peuvent être ajoutés pour traiter des phénomènes spécifiques à la langue[26]. En l'occurrence, pour le traitement de la langue arabe, nous avons eu besoin de définir trois nouvelles commandes morphologiques : <M>, <Z> et <T>. Ces opérateurs, que nous décrivons dans la section suivante, nous ont aidés à réduire le nombre de paradigmes flexionnels ; les deux premiers ont été utilisés pour les verbes et le troisième opérateur a été utilisé dans les descriptions flexionnelles des noms. En outre, compte tenu de la morphologie flexionnelle de l'arabe à la fois d'une grande régularité et d'une relative complexité, nous avons eu recours à des opérations de factorisation des règles ainsi que des appels récursifs entre les paradigmes.

4.4.2 Opérateurs morphologiques spécifiques à l'arabe

La première version de nos descriptions flexionnelles contenait environ 170 modèles pour décrire toutes les classes flexionnelles verbales. Cependant, la définition des deux nouveaux opérateurs morphologiques (<Z> et <M>) nous a permis de regrouper certaines descriptions où il n'y avait pas de grandes différences flexionnelles. Ces opérateurs sont à l'origine d'une réduction du nombre de descriptions pour n'avoir que 130 modèles.

- L'opérateur morphologique **<Z>** : il a été introduit pour pouvoir décrire certains verbes comme "كَتَبَ" (*kataba* - écrire) et "ثَبَتَ" (*ṯabata* – s'avérer) par le même paradigme en dépit de deux différentes transformations pour donner la conjugaison au passé. En effet, lors de sa flexion à la première personne du singulier à l'inaccompli actif, le premier verbe "كَتَبَ" nécessite la suppression de la dernière voyelle et l'ajout de la séquence "ْتُ" (*'tu*) pour obtenir la forme fléchie "كَتَبْتُ" (*katabtu* - j'ai écrit) et pour le second verbe la même conjugaison est accompagnée de l'ajout de la marque de redoublement ou Šadda " ّ ", et une dernière voyelle " ُ " (*u*) après suppression de la dernière voyelle pour obtenir "ثَبَتُّ" (*šabattu* – je me suis avéré). Le

[26] Nous citons, notamment, les opérateurs ajoutés pour le français, espagnol, catalan et portugais :
- ✓ <Á>: ajouter un accent aigu
- ✓ <À>: ajouter un accent grave
- ✓ <Ä>: ajouter un tréma
- ✓ <Â>: ajouter un circonflexe
- ✓ <A>: supprimer l'accent

regroupement des deux descriptions flexionnelles a été possible grâce à la définition de l'opérateur morphologique <Z> qui procède de deux façons différentes suivant le dernier caractère lu :
- ✓ si la dernière consonne du verbe est un "ت" (*t*), il fait suivre ce caractère par une Šadda ;
- ✓ sinon, il adjoint la séquence "تْ" (*'tu*).

- L'opérateur morphologique **<M>** : il est utilisée pour l'unification des descriptions flexionnelles des verbes comme "كَتَبَ" (*kataba* - écrire) et "دَهَنَ" (*dahana* - peindre). Cette unification a été effectuée en dépit de l'existence de deux transformations différentes pour conjuguer les deux verbes au passé, à la première personne du pluriel. En effet, le premier verbe devient "كَتَبْنَا" (*katabnaā* - nous avons écrit) et le deuxième devient "دَهَنَّا" (*dahannaā* – nous avons peint). De la même façon que l'opérateur morphologique précédent, <M> procède de deux façons différentes suivant le dernier caractère lu :
  - ✓ si la dernière consonne du verbe est un "ن" (*n*), il fait suivre ce caractère par une Šadda ;
  - ✓ sinon, il adjoint la séquence "نَ" (*'na*).

- L'opérateur morphologique **<T> :** il a été introduit lors de la description des paradigmes de déclinaisons nominales. Il permet de vérifier si la dernière consonne du nom à décliner est une "ة" (*ħ - tâ' marbûṭa*) qu'il remplacera par une "ت" (*t - tâ' maftuwḥa*) dans certaines règles flexionnelles ou dérivationnelles. Par exemple, lors de la description de la forme duale du lemme " مَدْرَسَة" (*madrasaaħ* - une école), il est nécessaire de substituer la lettre finale "ة" (*ħ - tâ' marbûṭa*) par une "ت" (*t - tâ' maftuwḥat*) avant d'ajouter le suffixe "يْن" (*ayni*) pour obtenir "مَدْرَسَتَيْن" (*madorasatayni* - deux écoles). Au cas où la dernière lettre est différente de "ة" (*ħ - tâ' marbûṭat*), cette commande est comparable à l'opérateur neutre <E> (Empty string) et n'a aucun effet sur la suite des opérations morphologiques.

4.4.3 Factorisation des règles de flexion

Dans certaines descriptions flexionnelles verbales, les formes générées sont différentes à un suffixe près. Auquel cas, il est convenable de préparer le radical commun avant de procéder à la suffixation. Pour illustrer ce propos, nous considérons les règles de conjugaisons du verbe "كَتَبَ" (*kataba* – écrire) à l'impératif. La description directe donne lieu à l'ensemble de règles :

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>*ْ /Y+2+m+s + #→ أُكْتُبْ/Y+2+m+s (*ùktub* - écris ! [masculin, singulier])

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>*ي/Y+2+f+s + #→ أُكْتُبِي/Y+2+f+s (*ùktubiy* - écris ! [féminin, singulier])

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>*ا و/Y+2+m+p + #→ أُكْتُبُوا/Y+2+m+p (*ùktubuwā* - écrivez ! [masculin, pluriel])

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S><M>/Y+2+f+p + #→ أُكْتُبْنَ/Y+2+f+p (*ùktubna* - écrivez ! [féminin, pluriel])

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>*ا ´/Y+d; #→ أُكْتُبَا/Y+2+d (*ùktubaā* - écrivez ! [duel])

Nous remarquons que les règles, ci-dessus, ont leur plus grande partie commune, soit

*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>* → أكْـتُب (*ùktub*)

Leurs factorisations donnent les règles suivantes:

(*<LW>*ا*<R><S>*ْ*<R><S>*ِ*<R><S>*) #→ أكْـتُب (*ùktub*)

(ْ/Y+2+m+s + #→ أكْـتُبْ/Y+2+m+s *(ùktub* - écris ! [masculin, singulier])

ي/Y+2+f+s + #→ أكْـتُبِي/Y+2+f+s (*ùktubiy* - écris ! [féminin, singulier])

ا◌ُو/Y+2+m+p + #→ أُكْتُبُوا/Y+2+m+p (*ùktubuwā* - écrivez ! [masculin, pluriel])

<M>◌ِي/Y+2+f+p + #→ أُكْتُبْنَ/Y+2+f+p (*ùktubna* - écrivez ! [féminin, pluriel])

ا◌ِ/Y+d); #→ أُكْتُبَا/Y+d (*ùktubaā* - écrivez ! [duel])

Les deux ensembles de règles de flexions sont équivalents ; la factorisation permet ainsi la simplification des descriptions flexionnelles et dérivationnelles.

Par analogie au système de paradigme flexionnel à base d'expressions régulières, nous pouvons écrire les règles de flexion sous forme d'un graphe flexionnel à l'aide de l'éditeur graphique de NooJ. Nous obtenons le graphe de la Figure 20.

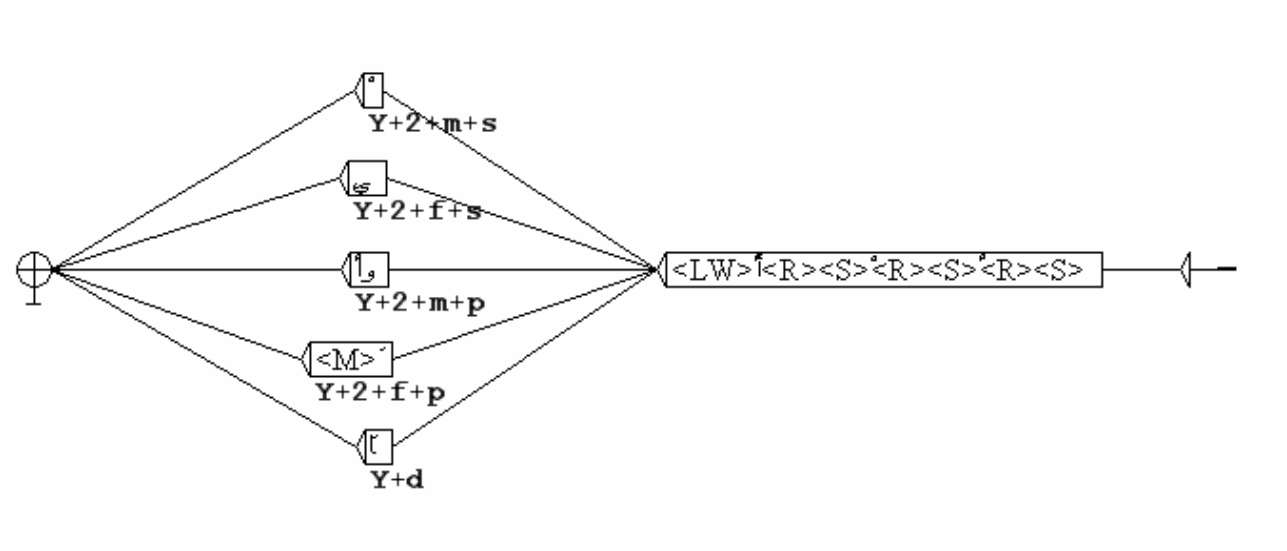


Figure 20 : Graphe flexionnel après factorisation des règles

4.4.4 Récursivité des règles de flexion

Dans d'autres cas, les descriptions flexionnelles se croisent dans une partie des règles de conjugaisons. Nous pouvons illustrer ce constat par une étude du système flexionnel des verbes sains qui ne présentent aucune cause de défectuosité et dont les lemmes suivent le schème "فَعَلَ" (*fa'ala*). Ces verbes sont partagés selon trois classes flexionnelles différentes suivant qu'au présent de l'indicatif, à la 3ème personne du singulier, la deuxième consonne du radical prenne :

- une "ضَمَّة - ُ" (*u, ḍammä*), par exemple: "كَتَبَ – يَكْتُبُ" (*kataba, uktubu*– écrire)
- une "فَتْحَة - َ" (*a, fatḥä*), par exemple : "جَرَحَ – يَجْرَحُ" (*jaraha, yajraḥu* – blesser)
- une "كَسْرَة - ِ" (*i, kasrä*), par exemple: "ضَرَبَ – يَضْرِبُ" (*ḍaraba, yaḍribu* – frapper)

Toutefois, tous les verbes, cités ci-dessus, se conjuguent identiquement à l'accompli. Les trois classes flexionnelles, dont ils dépendent, se partagent la totalité des règles de conjugaison au passé, à la voix active et passive. Une possibilité de factorisation des paradigmes est ainsi introduite afin d'éviter la redondance dans les descriptions. En effet, dans un paradigme flexionnel, nous pouvons faire appel à d'autres paradigmes en utilisant le caractère spécial « : ».

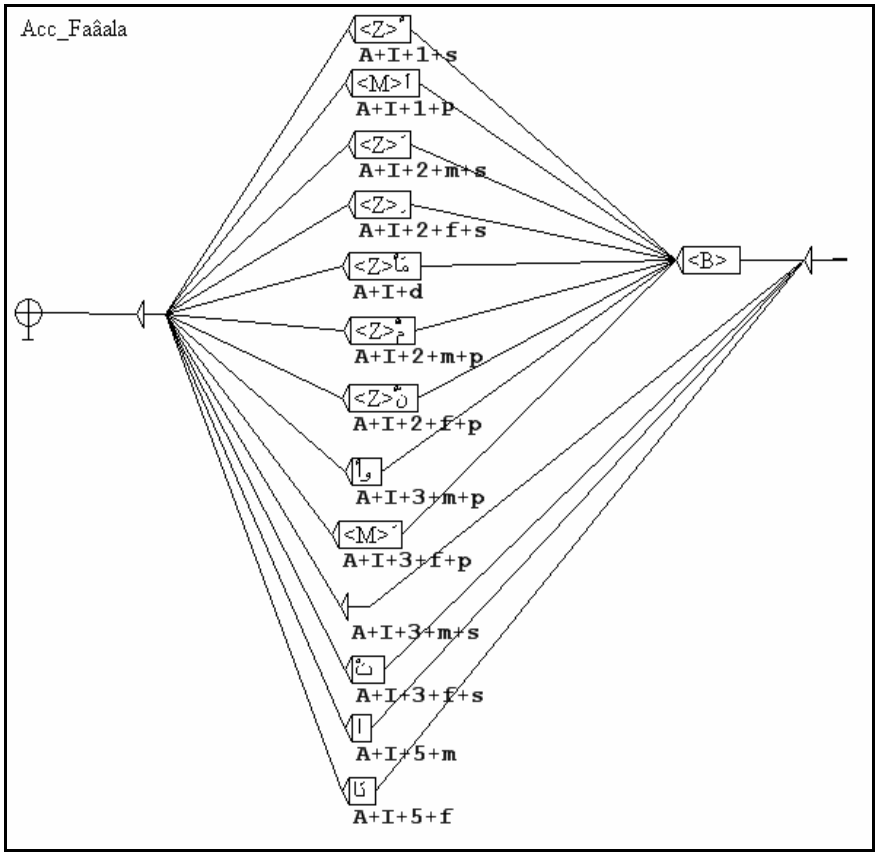
Dans l'exemple ci-dessous, le paradigme flexionnel « Acc_Faâala » contient les règles qui permettent de conjuguer les toirs types de verbes à l'accompli.

Acc_Faâala=

Voici les formes à

l'accompli, voix active :

<B>(<Z>ُ/A+I+1+s +
<M>ا/A+I+1+P +
<Z>َ/A+I+2+m+s +
<Z>ِ/A+I+2+f+s +
<Z>ْمَا/A+I+d +
<Z>ْمُ/A+I+2+m+p +
<Z>ْنَّ/A+I+2+f+p +
وا/A+I+3+m+p +
<M>َ/A+I+3+f+p) +
<E>/A+I+3+m+s +
تْ/A+I+3+f+s +
ا/A+I+5+m +
تَا/A+I+5+f ;

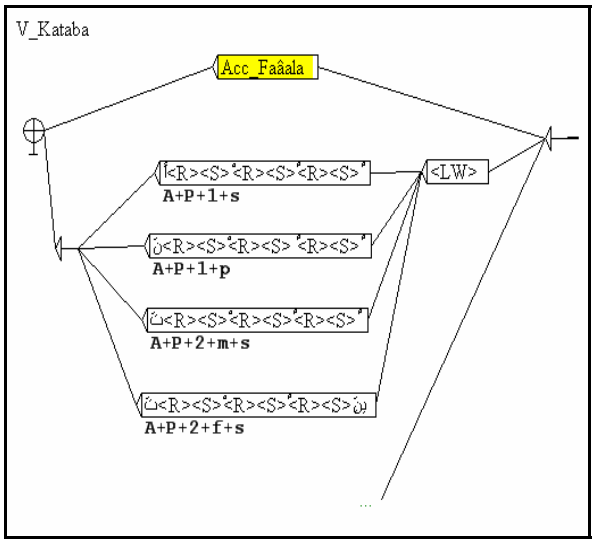


Le paradigme « :Acc_Faâala » est employé au sein des trois descriptions flexionnelles : V_Kataba, V_Jaraha et V_Daraba. Ces trois paradigmes se présentent comme suit :

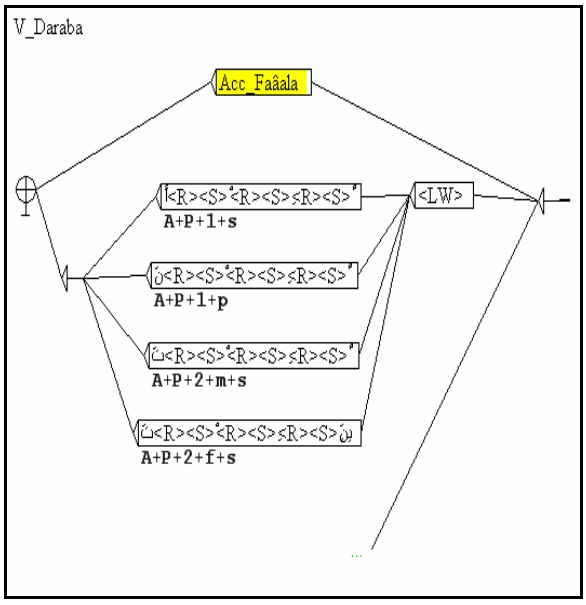
V_Kataba =

:Acc_Faâala +

Voici les formes au présent de l'indicatif, voix active :
<LW> (أ<R><S>ُ<R><S>َ<R><S>ُ/A+P+1+s +
نَ<R><S>ْ<R><S>ُ<R><S>َ/A+P+1+p +
تَ<R><S>ْ<R><S>ُ<R><S>َ/A+P+2+m+s +
تَ<R><S>ْ<R><S>ُ<R><S>ِينَ/A+P+2+f+s +
… ;



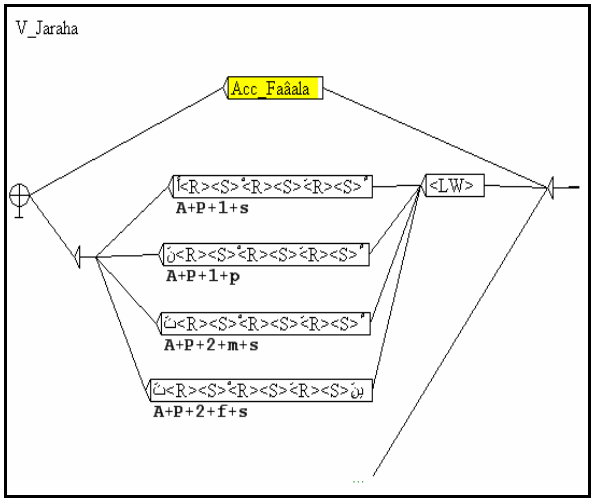
V_Daraba=



:Acc_Faâala +

Voici les formes au présent de l'indicatif, voix active:
<LW> (أ<R><S>ْ<R><S>ِ<R><S>ُ/A+P+1+s +
نَ<R><S>ْ<R><S>ِ<R><S>ُ/A+P+1+p +
تَ<R><S>ْ<R><S>ِ<R><S>ُ/A+P+2+m+s +
تَ<R><S>ْ<R><S>ِ<R><S>ِينَ/A+P+2+f+s +
… ;

V_Jaraha=



:Acc_Faâala +

Voici les formes au présent de l'indicatif, voix active:
<LW> (أ<R><S>ْ<R><S>َ<R><S>ُ/A+P+1+s +
نَ<R><S>ْ<R><S>َ<R><S>ُ/A+P+1+p +
تَ<R><S>ْ<R><S>َ<R><S>ُ/A+P+2+m+s +
تَ<R><S>ْ<R><S>َ<R><S>ِينَ/A+P+2+f+s +
… ;

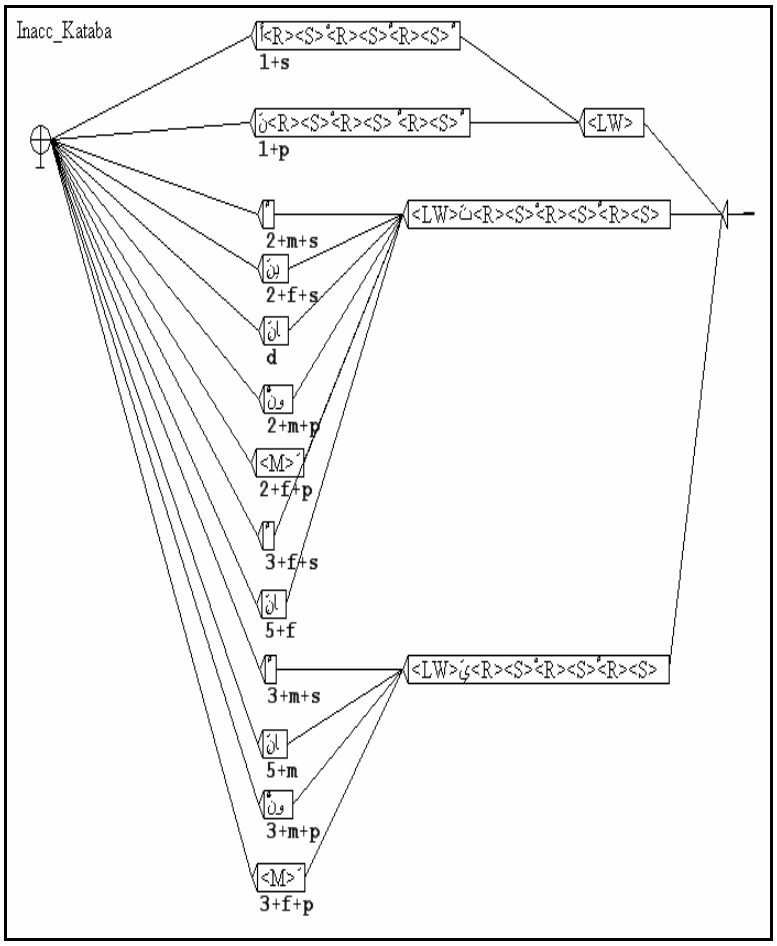
De façon analogique, nous pouvons tirer profit de la récursivité des règles pour simplifier la description de la conjugaison du verbe "كَتَبَ", plus particulièrement, au présent de l'indicatif et le futur. En effet, le futur morphologique se forme par une simple préfixation du morphème "سَـ" à partir des formes de l'inaccompli (présent de l'indicatif) générées à l'aide du paradigme flexionnel « Inacc_Kataba ».

Inacc_Kataba=

<LW>
(أ<R><S>ْ<R><S>ُ<R><S>ُ/1+s +
نَ<R><S>ْ<R><S>ُ<R><S>ُ/1+p) +

(<LW>تَ<R><S>ْ<R><S>ُ<R><S>)
(ُ/2+m+s +
ِينَ/2+f+s +
انِ/d +
ُونَ/2+m+p +
<M>ُ/2+f+p +
ُ/3+f+s +
انِ/5+f) +

(<LW>يَ<R><S>ْ<R><S>ُ<R><S>)
(ُ/3+m+s +
انِ/5+m +
ُونَ/3+m+p +
<M>ُ/3+f+p);



Le paradigme flexionnel « V_Kataba », de la page 95, devient alors :

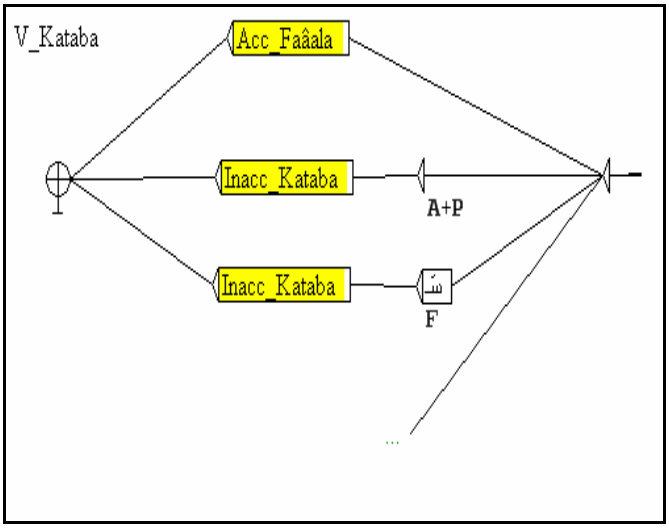
V_Kataba=

Voici les formes à l'accompli, voix active :
:Acc_Faâala +

Voici les formes au présent de l'indicatif, voix active :
<E>/A+P **: Inacc_Kataba** +

… +

Voici les formes au futur :
سَ/F **: Inacc_Kataba** +
… ;



La totalité des règles flexionnelles des trois verbes est donné en Annexe (cf. Annexe C).

4.4.5 Compilation automatique des dictionnaires électroniques

Suite à la phase de description des paradigmes flexionnels et dérivationnels, nous procédons à l'application de ceux-ci au dictionnaire des lemmes « El-DicAr » afin de générer la liste de toutes formes fléchies potentielles. En effet, à l'issue de la phase de compilation des dictionnaires, nous obtenons à partir de chaque entrée lexicale l'ensemble des formes qui s'y rattachent. Par exemple, lors de la phase de compilation du dictionnaire, nous partons de l'entrée lexicale simple telle que :

"دَرَّسَ",V+Tr+FLX=V_Darrasa+DRV=D_darrasa:FLXDRV(*darrasa* - enseigner)

nous obtenons, en moyenne, 170 entrées secondaires incluant :

- 122 formes verbales fléchies telle que :

  يُدَرِّسُونَ,دَرَّسَ,V+Tr+P+3+m+p (*yudarrisuwna, darrasa* – ils enseignent,enseigner) ;

- 48 formes déverbales fléchies partagées entre participes actifs et participes passifs telles que :

  مُدَرِّسُونَ,دَرَّسَ,N+PA+u+m+p (*mudarrisuwna, darrasa* –enseignants,enseigner) au nominatif
  مُدَرَّسَاتٌ,دَرَّسَ,N+PP+a+f+p (*mudarrasaāt, darrasa* – enseignées,enseigner)) à l'accusatif ;

Les entrées ainsi produites sont structurées de façon différente de celle du dictionnaire de lemmes. En lisant de gauche à droite nous trouvons les éléments suivants :

- le mot dans sa forme fléchie ;
- un premier séparateur de champ, c'est-à-dire la virgule "," ;
- le mot dans sa forme non fléchie, c'est-à-dire le lemme ;
- un second séparateur, c'est-à-dire la deuxième virgule "," ;
- l'étiquette grammaticale (V ou N) ;
- un troisième séparateur, c'est-à-dire le signe "+" ;
- l'ensemble de toutes les informations corrélées à la forme fléchie séparées par des signes « + » :
  - ✓ les informations syntactico-sémantiques : +Tr
  - ✓ la ou les informations flexionnelles correspondant à la forme : +P+3+m+p, +PA+u+m+p, +PP+a+f+p

Globalement, le ratio entre les versions actuelles du dictionnaire des lemmes et de celui des formes fléchies, pris dans leur totalité, est à peu près de 1:75. Ce rapport est calculé sur la base des presque quarante cinq milles lemmes présents dans le noyau de base du dictionnaire électronique « El-DicAr ». La liste générée automatiquement comprend, actuellement, environ 3 245 000 formes fléchies. Dans sa version lisible, ce dictionnaire a une taille énorme d'environ 350 millions d'octets. Toutefois, sur le plan pratique, les programmes de consultation de ce dictionnaire utilisent une version compactée sous forme d'un transducteur à états finis, acyclique, déterministe, minimal (voir section1.3) stocké dans un fichier dont la taille est inférieure à 50 Mo. A ce niveau, il nous importe de signaler que chaque mot à analyser ne serait attesté reconnu que s'il appartient au langage représenté par cet automate ou s'il est constitué d'une composition valide de morphèmes faisant partie du même langage.

4.5 Conclusion

Les dictionnaires électroniques sont étroitement liés aux programmes de traitement automatique. Toutes les informations qui y sont contenues doivent être cohérentes, aussi bien du point de vue de la forme des entrées que du point de vue des informations associées. Le dictionnaire « El-DicAr » que nous avons construit associe chaque entrée lexicale, qu'elle soit verbale, nominale ou autre, à l'ensemble des informations correspondantes ainsi que le ou les paradigmes flexionnels qui permettent la génération automatique de toutes les formes fléchies. Une liste des formes fléchies potentielles est ainsi générée: elle inclut des informations linguistiques utiles pour les tâches d'analyses ultérieures telles que : le lemme, la catégorie

grammaticale, les informations flexionnelles (genre, nombre, temps, mode, etc.), les informations syntaxiques (par exemple, la transitivité : +Transitif) et les informations distributionnelles et sémantiques (par exemple : +Humain, +Concret).

Chapitre 5

Utilisation de « El-DicAr » pour une analyse morpho-lexicale automatique

5.1 Introduction

Dans ce chapitre nous allons présenter un analyseur morphologique de l'arabe. Les ressources linguistiques de cet analyseur s'articulent autour du lexique précédemment construit « El-DicAr ». Ce dernier contient suffisamment d'informations pour jouer le rôle de moteur linguistique à l'intérieur des routines d'analyse textuelle automatique [Silberztein, 1991]. Grâce à ses qualités spécifiques et ses méthodes de représentation et de stockage, les données qui y sont incluses pourraient être facilement utilisées avec un accès très rapide pour l'analyse et la récupération des informations.

Toutefois, la langue arabe est une langue fortement agglutinante (voir Section 2.5.2). Du fait que notre analyseur sera destiné à des applications qui nécessitent des analyses fines au niveau du mot (y compris l'étiquetage automatique, la voyellation automatique, la recherche d'information, etc.), il ne s'agira pas seulement de vérifier si le mot appartient au lexique, mais aussi de donner tous les découpages potentiels en morphèmes. Pour chaque découpage retenu, chaque morphème sera associé à l'ensemble des informations morpho-syntaxiques respectives.

5.2 Tokenisation des formes agglutinées

Compte tenu de leurs abondances, il est indispensable de construire un système de tokenisation afin de remédier à la complexité de la structure agglutinante de certaines formes dans les textes arabes [Tuerlinckx, 2004].

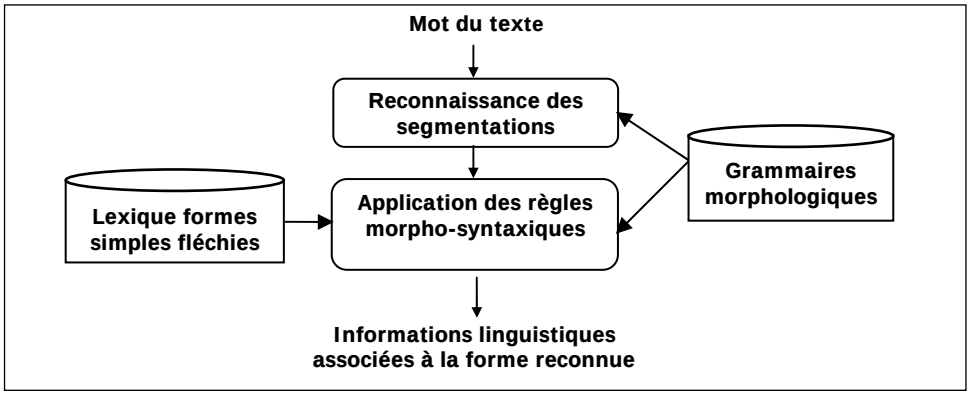


Figure 21 : Architecture de l'analyseur morphologique : chaîne de tokenisation d'un mot

La méthode que nous proposons peut-être vue comme un processus à deux phases :

1) Dans un premier lieu, nous appliquons un système de décomposition des formes agglutinées, implémenté sous forme de transducteurs à états finis (grammaires morphologiques NooJ), à chaque forme du texte. Ce système permet de reconnaître les segmentations potentielles en identifiant le radical et les différents affixes qui lui sont collés.

2) En second lieu, nous appliquons des règles morpho-syntaxiques aux différentes segmentations afin d'associer la reconnaissance d'une forme à un ensemble de contraintes lexicales permettant de travailler uniquement avec des combinaisons valides des différents constituants préalablement identifiés. Les segmentations retenues sont approuvées grâce à une consultation du lexique des formes fléchies précédemment construit.

Ces contraintes sont extrêmement utiles aussi bien pour représenter l'information linguistique que pour contrôler le processus d'analyse. Elles peuvent être considérées comme un mécanisme d'appoint. En effet, pour chaque découpage, l'analyseur doit attester l'existence du radical et sa

compatibilité avec le reste des morphèmes. A l'intérieur des grammaires morphologiques, nous avons recensé quatre types de contraintes lexicales :

5.2.1 Contraintes sur les propriétés des verbes

Ces contraintes lexicales sont les plus simples à mettre en place. Contrairement aux contraintes qui seront décrites ci-dessous où certains traitements supplémentaires accompagnent la segmentation des formes agglutinées, ce type de contraintes ne nécessite qu'une lecture des propriétés syntaxiques introduites dans les entrées verbales du dictionnaire. Elles prennent en compte les propriétés syntaxiques des verbes telles que la marque de transitivité, « +Transitif », ou les propriétés grammaticales telles que la voix de conjugaison du verbe. Ces propriétés nous permettent d'attester la validité de l'agglutination des enclitiques aux verbes. En effet, la transitivité du verbe et sa conjugaison autre qu'à la voix passive sont deux conditions nécessaires pour accepter son éventuelle agglutination à un pronom personnel. En l'occurrence :

- le verbe "كَتَبَ" (*kataba* - écrire) est transitif : l'agglutination de toutes ses formes fléchies à la voix active est acceptable. Par exemple, "كَتَبَهُ" (*katabahu* – il l'a écrit) est une forme agglutinée acceptable puisqu'elle est formée par une adjonction entre la forme verbale fléchie à la voix active "كَتَبَ " (*kataba* – il a écrit) au le pronom personnel "هُ" (*hu* – le) ;
- la conjugaison à la voix passive du verbe transitif "كَتَبَ" (*kataba* - écrire) mène à des agglutinations interdites avec les pronoms personnels en tant qu'enclitiques. Par exemple, après conjugaison du verbe "كَتَبَ" (*kataba* - écrire) à la voix passive, nous pouvons obtenir la forme fléchie "كُتِبَ" (*kutiba* – a été écrit), toutes les agglutinations de cette forme sont interdites tel est le cas pour "هُ + كُتِبَ" (*kutiba* + *hu*) ;
- le verbe "مَاتَ" (m*aā*ta - mourir) est intransitif : aucune de ses formes fléchies ne peut être collée à un pronom personnel en tant qu'enclitique : l'agglutination "هُ + مَاتَ" (*maāta* + *hu*) est illicite.

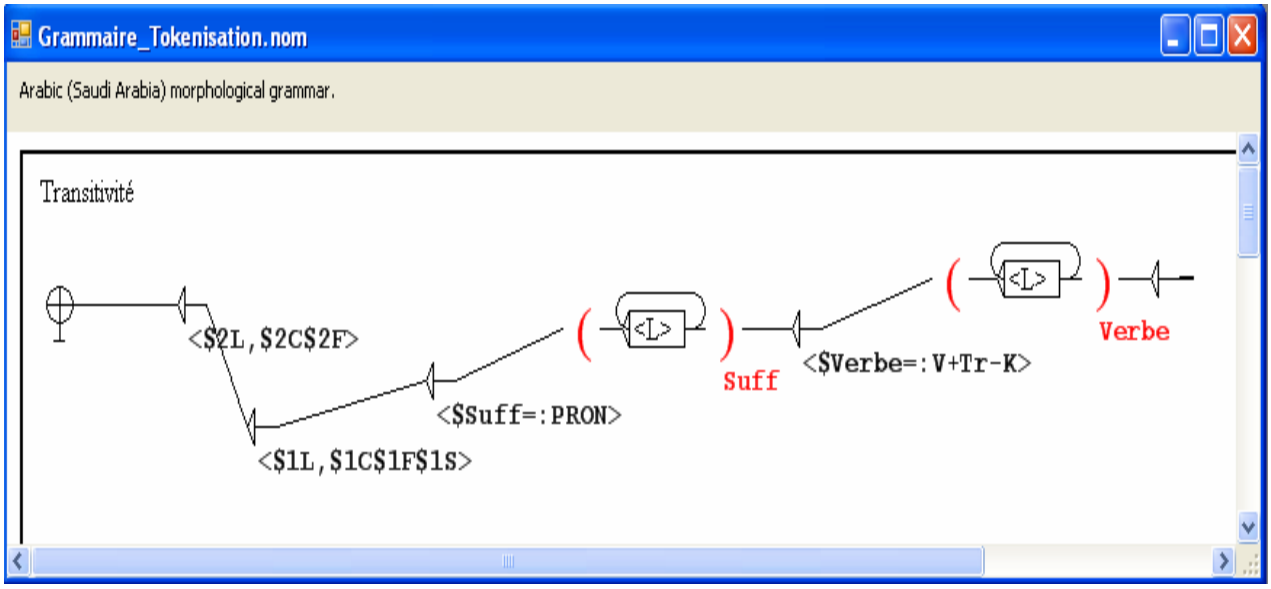


Figure 22 : Grammaire morphologique de tokenisation : Contraintes lexicales sur la transitivité des verbes

Pour faciliter la lecture et la compréhension du graphe de la Figure 22, nous nous limitons à une proportion simplifiée de la partie consacrée à la vérification de la transitivité des formes verbales au sein de notre grammaire d'analyse morphologique[27]. Ce graphe montre qu'une forme n'est

[27] Nous signalons que l'intégralité de la grammaire de tokenisation est téléchargeable avec l'ensemble des ressources linguistiques fournies gratuitement dans la rubrique « Resources » sur la page d'accueil de notre site web à l'adresse http://www.nooj4nlp.net, voir Annexe B.

reconnue comme agglutination d'un verbe et un pronom que si elle est décomposable en deux sous-chaînes vérifiant les contraintes mises en place. Ces deux sous-chaînes, constituées par deux suites de lettres représentées par (<L>*)[28], sont sauvegardées respectivement dans les variables $Verbe et $Suff. Les contenus de ces deux variables doivent vérifier les deux contraintes lexicales écrites entre crochets ('<' et '>'). Ces deux contraintes permettent de certifier la validité de la segmentation. En effet, nous avons :

- une première contrainte <$Verbe=:V+Tr-K> formée par la combinaison des deux contraintes simples <$Verbe=:V+Tr> et <$Verbe=:V-K>. Celles-ci permettent, en premier lieu, de vérifier la propriété de transitivité (+Tr) et en second lieu, de s'assurer que le verbe n'est pas conjugué à la voix passive (-K) ;
- une deuxième contrainte <$Suff=:PRON> qui permet de valider le contenu de la variable $Suff en tant que pronom personnel (PRON).

En cas de validité des deux contraintes lexicales citées, nous produisons l'ensemble des informations morpho-syntaxiques rattachées à chaque morphème de la segmentation retenue. En effet :

- à partir de la première contrainte, nous obtenons l'annotation <$1L,$1C$1F$1S> → le lemme ($1L) suivi de la catégorie grammaticale ($1C) et de l'ensemble des informations flexionnelles ($1F) et syntactico-sémantiques ($1S) ;
- à partir de la deuxième contrainte, nous obtenons l'annotation <$2L,$2C$2F> → le lemme ($2L) suivi de la catégorie grammaticale ($2C) et des informations flexionnelles qui s'y rattachent ($2F).

Pour illustrer notre démarche, nous proposons l'exemple de la forme agglutinée " كَتَبْتُهُ " (*katabtuhu* – je l'ai écrit). Suite à l'application des dictionnaires électroniques et de la grammaire morphologique de tokenisation, nous obtenons l'ensemble des contenus suivants :

1er morphème : le verbe	**2ème morphème : le suffixe**
$Verbe = "كَتَبْتُ" (*katabtu* – ai écrit)	$Suff= "ُهُ" (*hu* – le)
$1L = "كَتَبَ" (*kataba* – écrire)	$2L = "ُهُ" (*hu* – le)
$1C = V (verbe)	$2C = PRON (pronom)
$1F = A+I+1+s	$2F = 3+m+s
$1S = Tr	

A l'issue de notre analyse, et vu que les contenus mentionnées ci-dessus vérifient bien les deux contraintes lexicales, nous produisons les annotations[29] suivantes:

- <$1L,$1C$1F$1S> → كَتَبَ,كَتَبْتُ,V+Tr+A+I+1+s (*katabtu,kataba,V+Tr+A+I+1+s* – ai écrit,écrire,V+Tr+A+I+1+s) ;
- <$2L,$2C$2F> → ُهُ,PRON+3+m+s (*hu,hu, PRON+3+m+s* – le,le,PRON+3+m+s).

5.2.2 Contraintes morphologiques

Ces contraintes découlent de l'altération de certains radicaux par agglutination à un préfixe ou un suffixe. Elles permettent de rétablir la graphie initiale du radical telle qu'elle figure dans le

[28] Le symbole <L> désigne n'importe quelle lettre et le symbole « * » désigne l'opérateur Kleene, voir Section 1.1.
[29] Nous détaillons l'affichage des annotations dans la Section 5.5.1.

lexique en traitant les incompatibilités morphologiques qui seraient générées à partir d'une décomposition directe des formes agglutinées. Ces contraintes se basent sur un ensemble de transformations morphologiques telles que l'ajout de lettres, la suppression, la substitution, etc. ainsi que des combinaisons de celles-ci. Les exemples, ci-dessous, nous servent d'illustration pour ce type de contraintes :

- **Ajout de lettres :**
  Le graphe simplifié, ci-dessous, montre qu'une forme pourrait être reconnue comme agglutination d'un verbe et un pronom si elle est décomposable en deux sous-chaînes vérifiant les contraintes mises en évidence. Les deux sous-chaînes doivent être constituées de deux suites de lettres représentées par (<L>*), sauvegardées respectivement dans les variables $Verbe et $Suff, intercalées par l'une des séquences "ـُو" (*uw*) ou "ـْو" (*sukun* + *w*). Les contenus des deux variables doivent vérifier, respectivement, les deux contraintes suivantes :
  - ✓ <$Verbe#$Term#"ا"=:V-K+Tr+3+m+p> : cette contrainte utilise le caractère spécial « # » pour concaténer le contenu de la variable $Verbe, à celui de la variable $Term et la lettre "ا" (*ā*). Afin de valider la segmentation, la forme ainsi construite doit vérifier un ensemble des propriétés morpho-syntaxiques : un verbe (V), non conjugué à la voix passive (-K), transitif (+Tr), conjugué à la 3ème personne (+3), masculin (+m), pluriel (+p) ;
  - ✓ <$Suff=:PRON> : cette deuxième contrainte permet de valider le contenu de la variable $Suff en tant que pronom personnel (PRON).

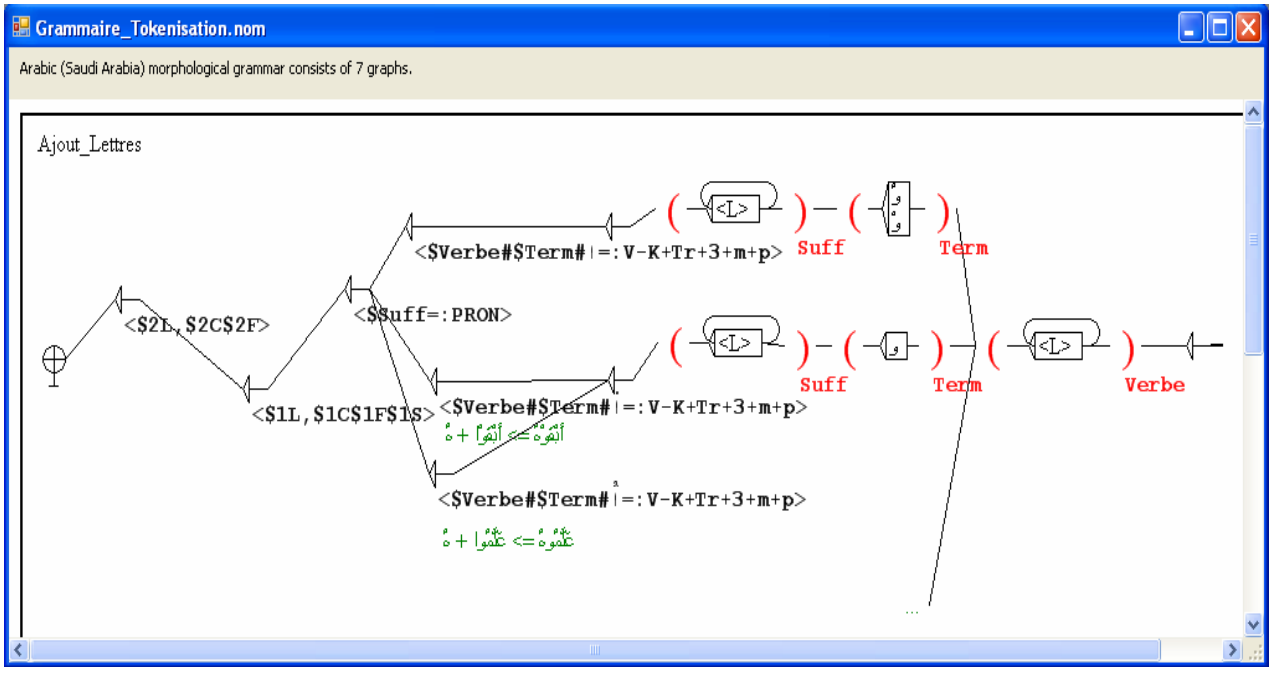


Figure 23 : Grammaire morphologique de tokenisation :
Contraintes morphologiques avec ajout de lettres

En l'occurrence, l'analyse morphologique de la forme "كَتَبُوهُ" (*katabuwhu* - Ils l'ont écrit) exige la restitution de la voyelle longue finale avant la consultation du dictionnaire :

- ✓ 1ère étape : segmentation de la forme en : verbe + suffixe: "كَتَبُو + هُ " (*katabuw* + *hu*)
- ✓ 2ème étape : cette étape peut être décomposée en deux phases :
  - ajout de la voyelle longue finale "ا" (*ā* - *álif*) au radical : "كَتَبُو" (*katabuw*) → "كَتَبُوا" (*katabuwā*)

- accès au dictionnaire: " كَتَبُوا + هُ " (*katabuwā* + *hu*) où "كَتَبُوا" (*katabuwā* – ils ont écrit) est la forme fléchie à la troisième personne, masculin, pluriel, à l'accompli, voix active et "هُ" (*hu* - le) est un pronom personnel.

Etant donné la validité des deux contraintes lexicales citées, nous produisons deux annotations comportant l'ensemble des informations morpho-syntaxiques rattachées à chaque morphème de la segmentation retenue. Si nous reprenons l'exemple détaillé, ci-dessus, nous produisons :

- ✓ <$1L,$1C$1F$1S> → كَتَبَ,كَتَبُوا,V+Tr+A+I+3+m+p (*katabuwā,kataba,V+Tr+A+I+3+m+p* – ils ont écrit,écrire,V+Tr+A+I+3+m+p) ;
- ✓ <$2L,$2C$2F> → هُ,هُ ,PRON+3+m+s (*hu,hu, PRON+3+m+s* – le,le,PRON+3+m+s).

- **Substitution de lettres :**

De façon analogue au graphe précédent, le graphe de la Figure 24 montre qu'une forme pourrait être reconnue comme agglutination d'un verbe et un pronom si elle est décomposable en deux sous-chaînes vérifiant certaines contraintes : les deux sous-chaînes doivent être constituées de deux suites de lettres représentées par (<L>*), sauvegardées respectivement dans les variables $Verbe et $Suff, intercalées par l'une des lettres substituables et vérifiant les deux contraintes lexicales suivantes :

- ✓ <$Verbe#"ى"=:V-K+Tr+3+m+s>[30] : la concaténation entre le contenu de la variable $Verbe et le substitut "ى" forme un verbe (V), non conjugué à la passive (-K), transitif (+Tr), conjugué à la 3ème personne (+3), masculin (+m), singulier (+s) ;
- ✓ <$Suff=:PRON> : cette deuxième contrainte permet de valider le contenu de la variable $Suff en tant que pronom personnel (PRON).

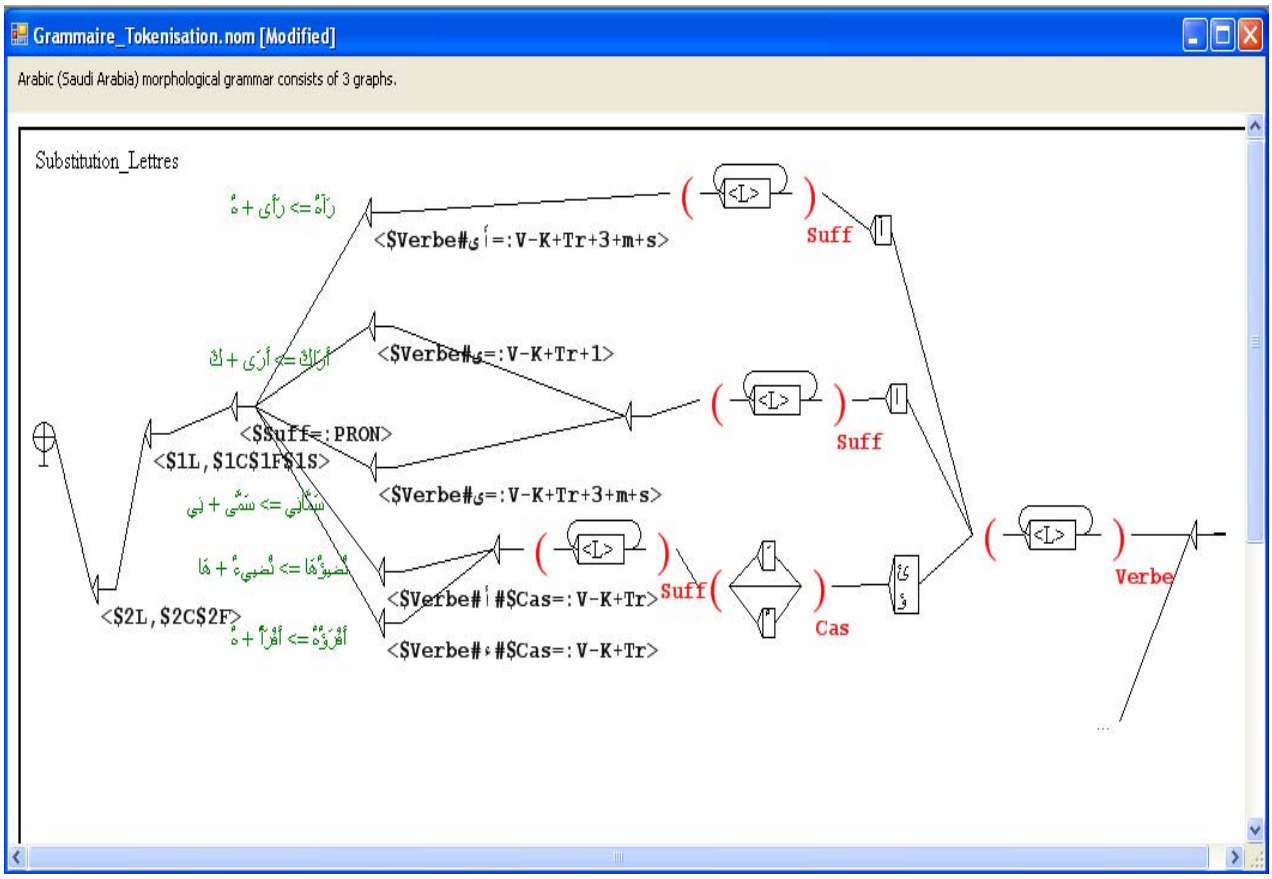


Figure 24 : Grammaire morphologique de tokenisation : Contraintes morphologiques avec substitution de lettres

[30] Dans cet exemple, nous avons repris la contrainte lexicale relative au troisième chemin de notre grammaire.

Par exemple, l'analyse morphologique de la forme "سَمَّانِي" (*sammaāniy* - Il m'a nommé) exige la substitution de lettre finale avant la consultation du dictionnaire :

- ✓ **1ère étape :** segmentation de la forme en : verbe + suffixe: "سَمَّا + نِي" (sammA + niy)
- ✓ **2ème étape :** cette étape est décomposée en deux phases :
  - substitution de la dernière lettre "ا" (*ā* - *álif*) par la voyelle longue "ى" (*ą* - *álif maqṣûrä*) : "سَمَّا" (*sammaā*)→ "سَمَّى" (*sammaą*) ;
  - accès au dictionnaire: "سَمَّى + نِي" (*sammaą* + *niy*) où "سَمَّى" (*sammaą* - nommer quelqu'un) est la forme fléchie à la 3ème personne, masculin, singulier, inaccompli, voix active et "نِي" (*niy* - moi) est un pronom personnel.

A l'issue de cette décomposition, suite à la vérification des deux contraintes lexicales, nous produisons les annotations suivantes :

- ✓ <$1L,$1C$1F$1S> → سَمَّى,سَمَّى ,V+Tr+A+I+3+m+s (*sammaą,sammaą,V+Tr+A+I+3+m+s* – il a nommé, nommer quelqu'un,V+Tr+A+I+3+m+s) ;
- ✓ <$2L,$2C$2F> → نِي,نِي ,PRON+1+s (*niy, niy, PRON+1+s* – moi,moi,PRON+1+s).

- **Suppression de lettres :**

Identiquement aux contraintes précédentes, nous montrons, dans ce qui suit, comment procéder à l'analyse des formes agglutinées lorsque celles-ci demandent une suppression de certaines lettres afin de restituer la graphie initiale avant agglutination. Nous limitons notre description aux deux cas les plus courants :

- ✓ Le cas des formes verbales dont la segmentation est acceptée seulement lorsqu'elles sont dissociables en trois sous chaînes stockées respectivement dans les variables $Verbe, $Suppr et $Suff. Ensuite, après omission du contenu de la deuxième variable ($Suppr) nous procédons aux contrôles des contenus des deux autres variables par le biais des deux contraintes : <$Verbe#"ْ "=:V-K+Tr+2+m+p> et <$Suff=:PRON>. Par exemple, l'analyse morphologique de la forme "كَتَبْتُمُوه" (*katabtumuwhu* – vous l'avez écrit) exige la suppression de deux lettres avant la consultation du dictionnaire :
  - **1ère étape :** segmentation de la forme en : verbe + suffixe : "كَتَبْتُمُو + هُ " (*katabtumuw* + *hu*) ;
  - **2ème étape :** cette étape peut être décomposée en deux phases :
    - suppression des deux dernières lettres "ُو" (*uw*) : "كَتَبْتُمُو " (*katabtumuw*) → "كَتَبْتُم " (*katabtum*) ;
    - accès au dictionnaire : "كَتَبْتُم + هُ " (*katabtum* + *hu*) où "كَتَبْتُم " (*katabtum* – vous avez écrit) est la forme fléchie à la deuxième personne, masculin, pluriel, à l'accompli, voix active et "هُ" (*hu* - le) est un pronom personnel.
- ✓ Le cas des formes nominales dont la segmentation est acceptée uniquement lorsqu'elles sont décomposables en trois sous chaînes stockées respectivement dans les variables $Pref, $1ereLettre et $Nom. A ce propos, nous signalons qu'il existe 14 consonnes[31] arabes parmi les 28 lettres de l'alphabet arabe qui exigent l'ajout du signe de gémination " ّ " (*šaddä* – šadda) lors de leurs préfixations par l'article défini "ال" (*el* - le) lorsqu'ils sont en position de 1ère lettre du radical. Une suppression de cette šadda est nécessaire pour la tokenisation. Par exemple, L'analyse morphologique de la forme "الدَّلْوُ" (*al-ddalwu* - le seau) exige la suppression de la šadda " ّ " (*šaddä*), ce qui équivaut à supprimer la duplication de la lettre "*d*" dans la forme translittérée, avant de consulter le dictionnaire :
  - **1ère étape :** segmentation de la forme en : préfixe + verbe : " ال+ دَّلْوُ " (*el* + *ddalwu*) ;

[31] Les 14 consonnes, qui nécessitent l'ajout de la šadda lors de la concaténation avec l'article de définition "ال" (*el* - le), s'appellent "حروف شمسية" (*ḥuruwf šamsiyyaħ* - lettres solaires). Les autres consonnes s'appellent "حروف قمرية" (*ḥuruwf qamariyyaħ* - lettres lunaires), voir Annexe A.

- **2ème étape :** cette étape est décomposée en deux phases :
  - suppression du signe de gémination " ّ " (*šaddä* – Chadda) : ceci qui correspond à la suppression de la seconde consonne "*d*" dans la forme translittérée. Cette Šadda a été rajoutée automatiquement lors de la détermination du nom "دَلْوُ" (*dalwu* - seau) qui commence par "حرف شمسي" (*ḥuruwf šamsiyyat* - lettre solaire) "د" (*d*) ;
  - accès au dictionnaire : "دَلْوُ + ال" (*el* + *dalwu*) où "ال" (*el* - le) un article défini et "دَلْوُ" (*dalwu* - seau) est la forme fléchie du substantif au singulier nominatif.

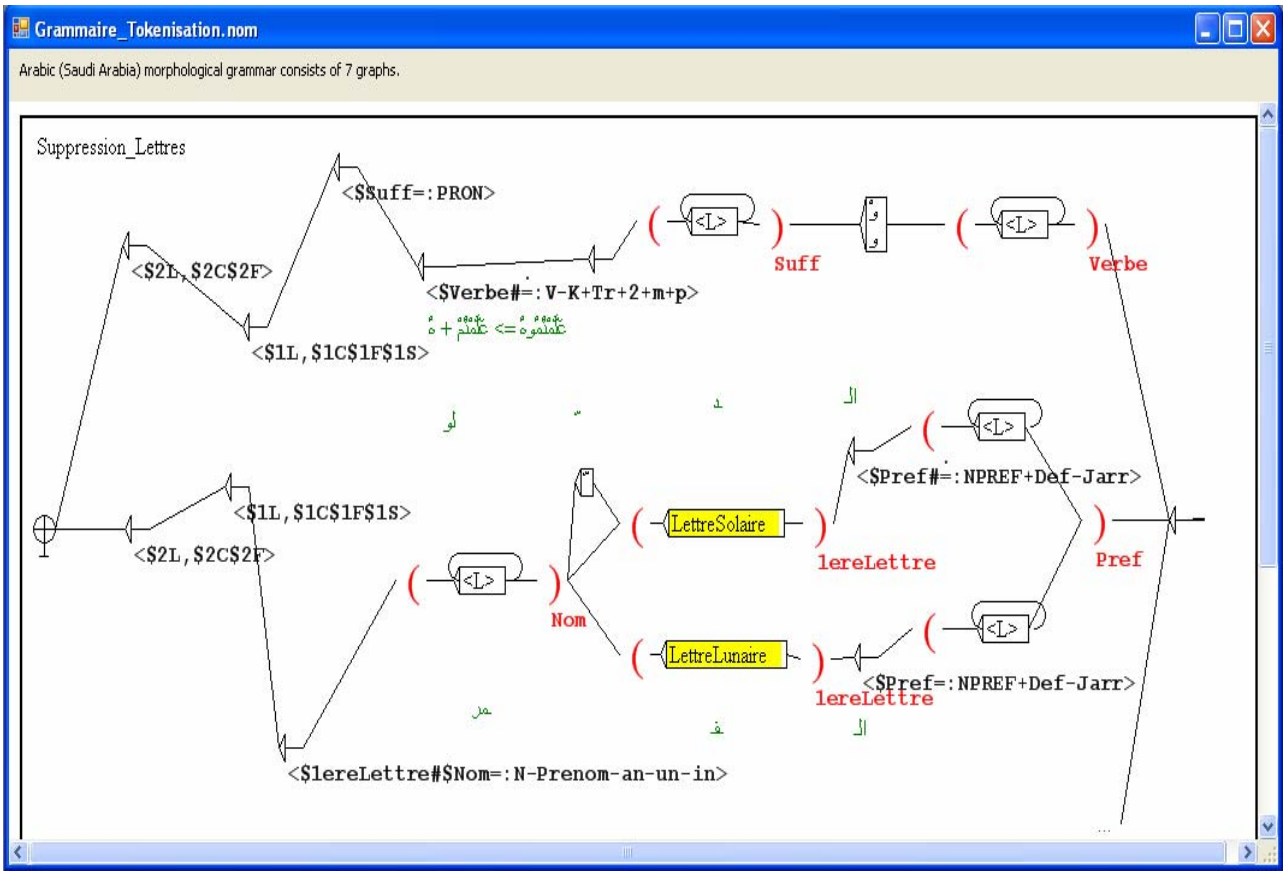


Figure 25 : Grammaire morphologique de tokenisation : Contraintes morphologiques avec suppression de lettres

Les transformations décrites (ajout, substitution ou suppression de lettre) peuvent être combinées entre-elles pour traiter des phénomènes morphologiques plus complexes afin de prendre en considération les incompatibilités qui pourraient être générées à partir d'une décomposition directe. En l'occurrence, la segmentation de la forme agglutinée "للعب" (*li-l-laʿibi* – pour le divertissement) nécessite deux étapes :

- ✓ la reconnaissance de la préposition "لـ" (*li* – pour) ;
- ✓ la suppression du signe de gémination " ّ " (*šaddä* – Chadda). Nous notons que la présence de la šadda dans cette forme agglutinée dénote la présence implicite du proclitique de détermination "ال" (*el* - le). Cette transformation morphologique, c'est-à-dire la suppression de la šadda, occulte une autre transformation, celle liée à la restitution de l'article défini comme proclitique.

A l'issue de notre analyse morphologique automatique, la forme agglutinée proposée, "للعب" (*li-l-la'ibi*), est traitée en tant qu'agglutination de :

- ✓ la préposition "لـ" (*li* – pour) ;
- ✓ l'article défini "ال" (*el* - le) ;
- ✓ la forme nominale "لعب" (*la'ibi* – divertissement).

5.2.3 Contraintes orthographiques

Ces contraintes prennent en compte le changement de l'orthographe de certaines lettres lors d'une agglutination. Nous citons le cas de la lettre *tâ'* acceptant deux orthographes différentes : 'ت' (*t - tâ' maftûhä*) et 'ة' (*ħ - tâ' marbûṭä*) et le cas des *álif*s possédant cinq orthographes différentes : 'ء' (*' – hamzä 'laq es-saṭr*), 'أ' (*á - hamzä*), 'إ' (*í - hamzä*), 'ؤ' (*ẃ - hamzä 'laq al-wâw*) et 'ئ' (*ý – hamzä 'laq al-yâ'*). La présence de l'une ou l'autre des orthographes potentielles est liée à la position de la lettre à variantes dans le mot, de la voyelle qui l'accompagne et, parfois, de la voyelle qui la précède. En fait, lors de leurs agglutinations à des pronoms relatifs en tant qu'enclitiques, les noms marqués au féminin par la lettre finale "ة" (*ħ - tâ' marbûṭä*) subissent une régularisation morphologique en remplaçant celle-ci par une "ت" (*t - tâ' maftûhä*). Des opérations de substitution inverses sont, alors, à prévoir avant la consultation du dictionnaire et l'attribution des informations linguistiques correspondantes à la forme en entrée.

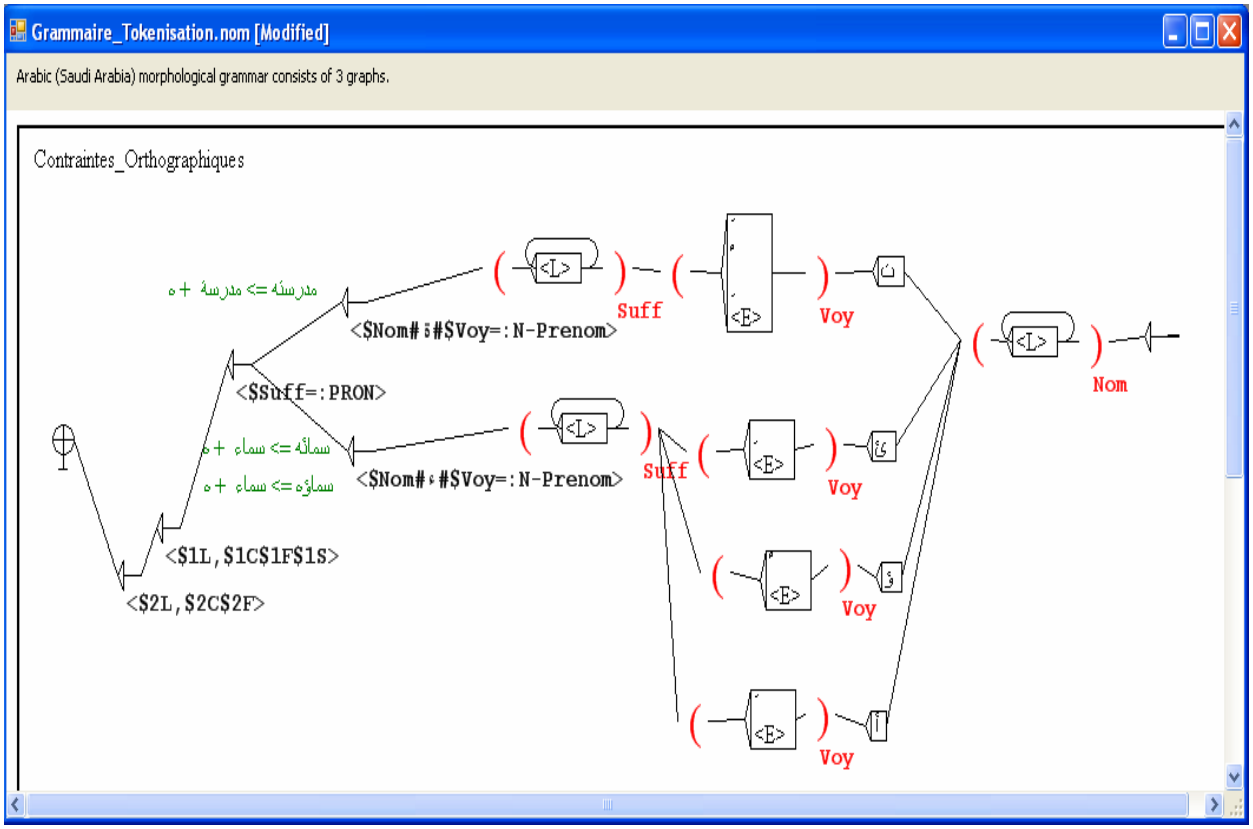


Figure 26 : Grammaire morphologique de tokenisation : Contraintes orthographiques

Par exemple, le mot "مَدْرَسَتِهِ" (*madrasatihi* - son école) est décomposable en "مَدْرَسَتِ + هِ" (*madrasati + hi*). Avant de consulter le dictionnaire, nous devons remplacer la lettre "ت" (*t - tâ' maftûhä*) par la lettre "ة" (*ħ - tâ' marbûṭä*) pour restituer l'orthographe initiale de la forme nominale " مَدْرَسَةِ "(*madrasati* - école) telle qu'elle figure dans le lexique. Ainsi, nous obtenons la

segmentation "مَدْرَسَةِ + ِه"(*madrasati* + *hi*) où " مَدْرَسَةِ "(*madrasati* - école) est le cas génitif du substantif et "ِه" (*hi* - son) est un pronom personnel. Nous signalons que cette analyse a fait usage de la contrainte lexicale <$Nom#ة #$Voy=N-Prenom> qui permet de vérifier si le contenu généré par « $Nom#ة #$Voy » est un nom (N), différent d'un prénom (-Prenom).

De façon analogue, les formes nominales qui se terminent par une hamza "ء" (*' – hamzä 'laq es-satir*) subissent une régularisation du support de cette hamza lors de la phase d'agglutination en tenant compte des signes diacritiques qui entourent la hamza et de la fonction de cette forme dans la phrase. Par exemple, les deux formes "سَمَائِهِ" (*samaāýihi* – son ciel, au génitif) et "سَمَاؤُهُ" (*samaāẃuhu* – son ciel, au nominatif) ont subi deux régularisations différentes liées à leurs voyelles casuelles et leurs fonctions grammaticales. D'une façon générale :

- Si la hamza est accompagnée par une "ُ– ضَمَّة" (*u* - *ḍammä*), elle prend la forme "ؤ" (*ẃ* - *hamzä 'laq al-wâw*) : c'est le nominatif;
- Si la hamza est accompagnée par une "َ– فَتْحَة" (*a* - *fatḥä*), elle prend la forme "أ" (*á* - *hamzä*) ou "ء" (*' – hamzä 'laq es-satir*) : c'est l'accusatif ;
- Si la hamza est accompagnée par une "ِ– كَسْرَة" (*i* - *kasrä*), elle prend la forme "ئ" (*ý – hamzä 'laq al-yâ'*) : c'est le génitif ;
- Si la hamza est accompagnée par un "ْ– سُكُون" (*sukuwn* - signe de quiescence) ou une lettre de prolongation (voyelle longue), elle prend la forme "ء" (*' – hamzä 'laq es-satir*).

5.2.4 Contraintes phonologiques

Ces contraintes sont généralement combinées avec celles déjà citées afin de maintenir une consonance harmonieuse à l'intérieur des formes agglutinées. Elles concernent la compatibilité de la flexion casuelle du radical avec celle du suffixe qui s'y rattache.

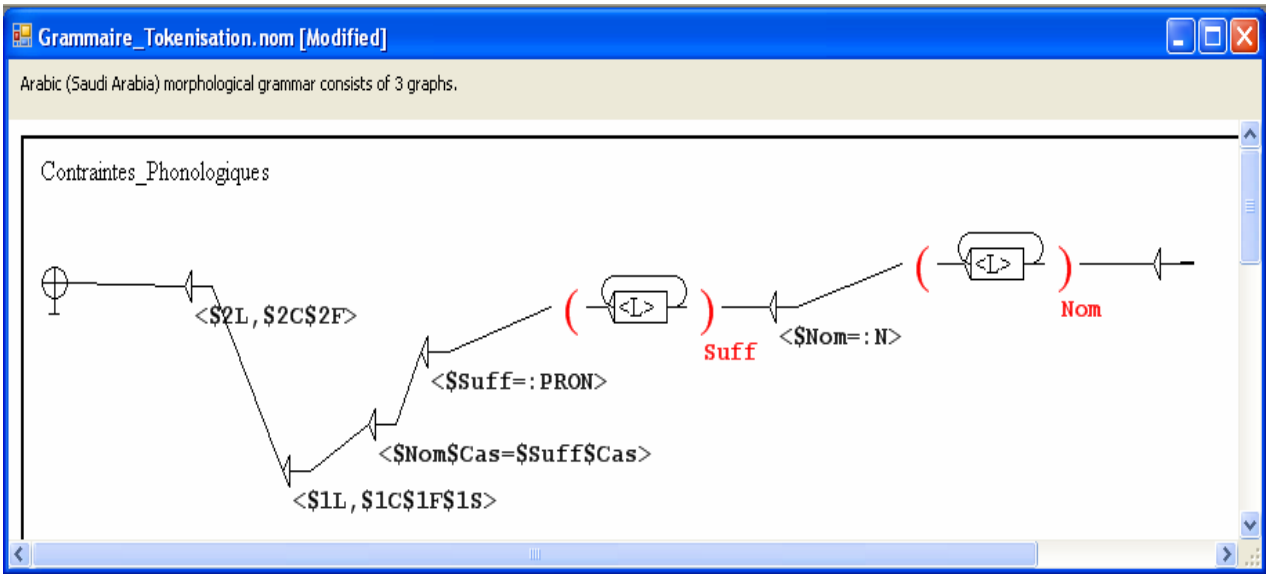


Figure 27 : Grammaire morphologique de tokenisation : Contraintes phonologiques

Par exemple, pour l'analyse morphologique du mot "كِتَابِهِ" (*kitaābihi* - son livre), nous commençons par une première étape de segmentation qui donne "كِتَابِ" (*kitaābi* - livre au génitif) + "ِه" (*hi* - pronom personnel au génitif). Ensuite, nous validons cette segmentation grâce à l'harmonie vocalique que présente les deux morphèmes "كِتَابِ" (*kitaābi* – livre) et "ِه" (*hi* – son) compte tenu de leur déclinaison commune au génitif.

Toutefois, la concaténation de la même forme (*kitaāb* - livre) déclinée à l'accusatif et le même pronom personnel au génitif, produit une agglutination interdite. Ainsi, la forme agglutinée "كِتَابَهِ" (*kitaāba* + hi) est illicite en raison de l'incompatibilité entre la flexion casuelle à l'accusatif au niveau du substantif "كِتَاب" (*kitaāb* - livre), et au génitif pour le pronom personnel en suffixe "ه" (hi - son).

Cette harmonie vocalique est traduite à l'intérieur de la grammaire par l'intermédiaire de la contrainte lexicale <NomCas=$Suff$Cas>. Cette contrainte commence par la lecture de la valeur de la propriété flexionnelle ($Cas) relative à chacune des deux variables. Ensuite, elle procède à la vérification de leur cohérence.

Parallèlement à la phase d'analyse morphologique, et plus précisément lors de la consultation des dictionnaires, trois cas peuvent se présenter :

- **Le mot appartient au lexique et il est totalement voyellé :** dans ce cas, nous nous limitons à un renvoi de l'ensemble des informations morpho-syntaxiques qui s'y rattachent. Aucune intervention de notre part n'est, alors, sollicitée ;
- **Le mot appartient au lexique sauf qu'il est partiellement ou non voyellé :** dans ce cas, nous renvoyons la liste des mots similaires totalement voyellés. Cette fonctionnalité de restitution automatique des voyelles sera détaillée dans la section suivante ;
- **Le mot n'appartient pas au lexique** : ceci peut être dû à une erreur d'orthographe ou une omission de son listage lors de la phase de formalisation du lexique. Au premier cas, nous proposons des heuristiques permettant de reconnaître les erreurs les plus fréquentes (voir section5.4). Dans le second cas, nous proposons à l'utilisateur une liste de toutes les formes inconnues.

5.3 Voyellation automatique

Théoriquement, seuls le Coran et les livres des enfants sont entièrement voyellés. Toutefois, les textes courants de la langue sont non voyellés ou, dans le meilleur des cas, quelques voyelles brèves y sont incluses en vue d'une réduction de l'ambiguïté qui s'ensuit de leurs absences totales ou partielles (voir section 2.5.3). Souvent, ces voyelles sont insérées de manière peu cohérente (par exemple : la šadda sur les lettres "solaires" après l'article), voire erronée (par exemple : la *hamzä qaṭ'iyya* à l'initiale d'un nom).

Compte tenu de l'importance de la part qu'occupe ce problème vis-à-vis des résultats de l'analyse automatique des textes, la robustesse de la solution proposée est directement liée à la possibilité de traiter aussi bien les textes voyellés que ceux qui sont non voyellés ou qui le sont partiellement. Dans les deux derniers cas, une restitution des voyelles manquantes doit accompagner la phase d'analyse morphologique.

Cependant, nous signalons que l'analyseur lexical dans NooJ dans sa première version, v1.x, utilisait un algorithme classique de reconnaissance des formes stockées dans des transducteurs à états finis :

> *« Nous partons du nœud initial, et nous lisons la chaîne en entrée en suivant au fur et à mesure dans l'automate les chemins qui traversent les nœuds dont les étiquettes sont identiques aux symboles lus. Si nous arrivons à un nœud terminal après avoir lu toute la séquence, la séquence est reconnue ; en revanche, si nous n'avons pas pu rejoindre un nœud terminal, ou si nous avons rejoint le nœud terminal après avoir lu toute la chaîne en entrée, celle-ci n'est pas reconnue. Au cas où la reconnaissance est réussie, nous associons à séquence reconnue toutes les informations rattachées au nœud terminal, […] »* [Silberztein, 1993] (voir section 1.5).

Ce programme de reconnaissance est inadéquat pour le traitement de la langue arabe. En effet, lors du traitement d'une forme telle qu'elle figure dans les textes courants, la lecture des différents symboles qui y sont présents ne serait pas suffisante pour atteindre un nœud terminal à cause de l'absence de certaines voyelles par rapport à la même forme qui aurait été intégrée dans le lexique.

5.3.1 Un nouvel algorithme d'analyse lexicale

Dans cette section, nous nous intéressons, principalement, au traitement des formes présentes dans les textes qui appartiennent au lexique sans qu'elles soient correctement représentées dans les textes et ce à cause de l'absence partielle ou totale des signes diacritiques. L'objectif de ce traitement est de restituer la graphie initiale de ces formes telles qu'elles apparaissent dans le dictionnaire électronique.

Traditionnellement, les méthodes utilisées pour aborder ce problème procède par :

- une étape de dévoyellation qui consiste à séparer les consonnes et les voyelles du mot à analyser ;
- une mémorisation de la position de chaque voyelle dans le mot ;
- une mise en conformité avec la hiérarchie du lexique.

Cette démarche semble être assez coûteuse en termes de temps d'analyse. L'utilisation de la technologie à états finis pour le stockage du lexique pourrait apporter une solution plus adéquate. En effet, pour remédier à un tel coût, nous proposons un nouvel algorithme de parcours des transducteurs de stockage pour le moteur d'analyse lexicale dans NooJ. Cet algorithme est utilisé pour toutes les langues sémitiques dans NooJ y compris l'arabe : Il part du constat que toute consonne d'une forme voyellée, dans n'importe quelle langue sémitique, est suivie d'une voyelle brève ou un signe de gémination (la Šadda pour l'arabe ou le Daguesh pour l'hébreu).

En effet, lors de la consultation des dictionnaires pour la vérification de l'appartenance d'une forme au lexique stocké sous forme de transducteurs à états finis, nous procédons au parcours de ces transducteurs comme suit :

- nous partons du nœud initial du transducteur ;
- nous lisons la chaîne en entrée caractère par caractère et nous avançons au fur et à mesure dans le transducteur par l'intermédiaire de transitions dont les étiquettes sont identiques aux symboles lus. Trois cas peuvent se présenter :
  - ✓ Le symbole lu est une consonne ou un signe de gémination. Dans ce cas, nous passons à la lecture du symbole suivant qui devrait être une voyelle ou un signe de gémination :
    - o si celui-ci est une voyelle, nous avançons dans le transducteur tant que possible ;
    - o si celui-ci est un signe de gémination, nous passons à la lecture du symbole suivant sachant qu'il devrait être une voyelle ;
    - o si celui-ci est une consonne, nous avançons sur tous les chemins accessibles par l'intermédiaire de transitions étiquetées par n'importe quel signe diacritique.
  - ✓ Le symbole lu est une voyelle longue : Celui-ci peut être suivi d'une voyelle ou non selon son comportement à l'intérieur de la forme (en tant que consonne ou glide) ;
  - ✓ Le symbole lu est une voyelle : Nous poursuivons le parcours du transducteur sans aucune intervention.
- Si nous arrivons à un nœud terminal après avoir lu toute la forme en entrée, cette séquence est attestée reconnue. Le cas échéant, cette séquence est considérée comme non reconnue. Au cas où la reconnaissance a réussi, nous associons à la séquence reconnue toutes les informations rattachées au nœud terminal.

Cette approche ne demande aucun temps de calcul supplémentaire pour restituer les voyelles d'une forme non voyellée ou partiellement voyellée. Elle nous permet de traiter toutes les formes,

indépendamment de leurs états de vocalisation, en un temps linéaire qui dépend uniquement de la longueur de celles-ci, soit $O(n)$ où n est la taille de la forme à analyser.

Notons que la présence d'une voyelle brève ou d'une šadda dans la forme en entrée est automatiquement prise en compte pour éliminer des chemins qui étaient attestés acceptables. En effet, le parcours d'un chemin du transducteur est abandonné à la première discordance avec un symbole lu du mot en entrée. Ainsi, nous évitons les parcours inutiles et nous aboutissant à un énorme gain en temps de traitement tout en évitant la génération de mauvaises analyses. Par exemple :

- Si le mot en entrée est كتب (*ktb*), en l'absence de toute contrainte vocalique, notre analyseur fournit 15 voyellations potentielles représentant une forme simple (voir section 2.5.1) telles que كَتَبَ (*kataba* – il a écrit), كُتِبَ (*kutiba* - il a été écrit), كَتْبٌ (*katbun* – un écrit), كُتُبٌ (*kutubun* – des livres), etc ;
- Si le mot en entrée est كَتب (*ka_tb*), en présence de la voyelle qui suit la première consonne, notre analyseur évite toutes les discordances et ne fournit que les voyellations potentielles respectant cette contrainte telles que كَتَبَ (*ka_taba* – il a écrit) et كَتْبٌ (*ka_tbun* – un écrit) et il écarte les voyellations telles que كُتُبَ (*kutiba* - il a été écrit) et كُتُبٌ (*kutubun* – des livres) ;
- Si le mot en entrée est كتبٌ (*ktbun*), en présence de la voyelle casuelle, notre analyseur évite toutes discordances et ne fournit que les deux seules voyellations potentielles respectant cette contrainte, soit كَتْبٌ (*katbun* – un écrit) et كُتُبٌ (*kutubun* – des livres) ; il écarte les autres voyellations telles que كَتَبَ (*kataba* – il a écrit) et كُتِبَ (*kutiba* - il a été écrit), etc.

5.3.2 Démarche de restitution automatique des voyelles

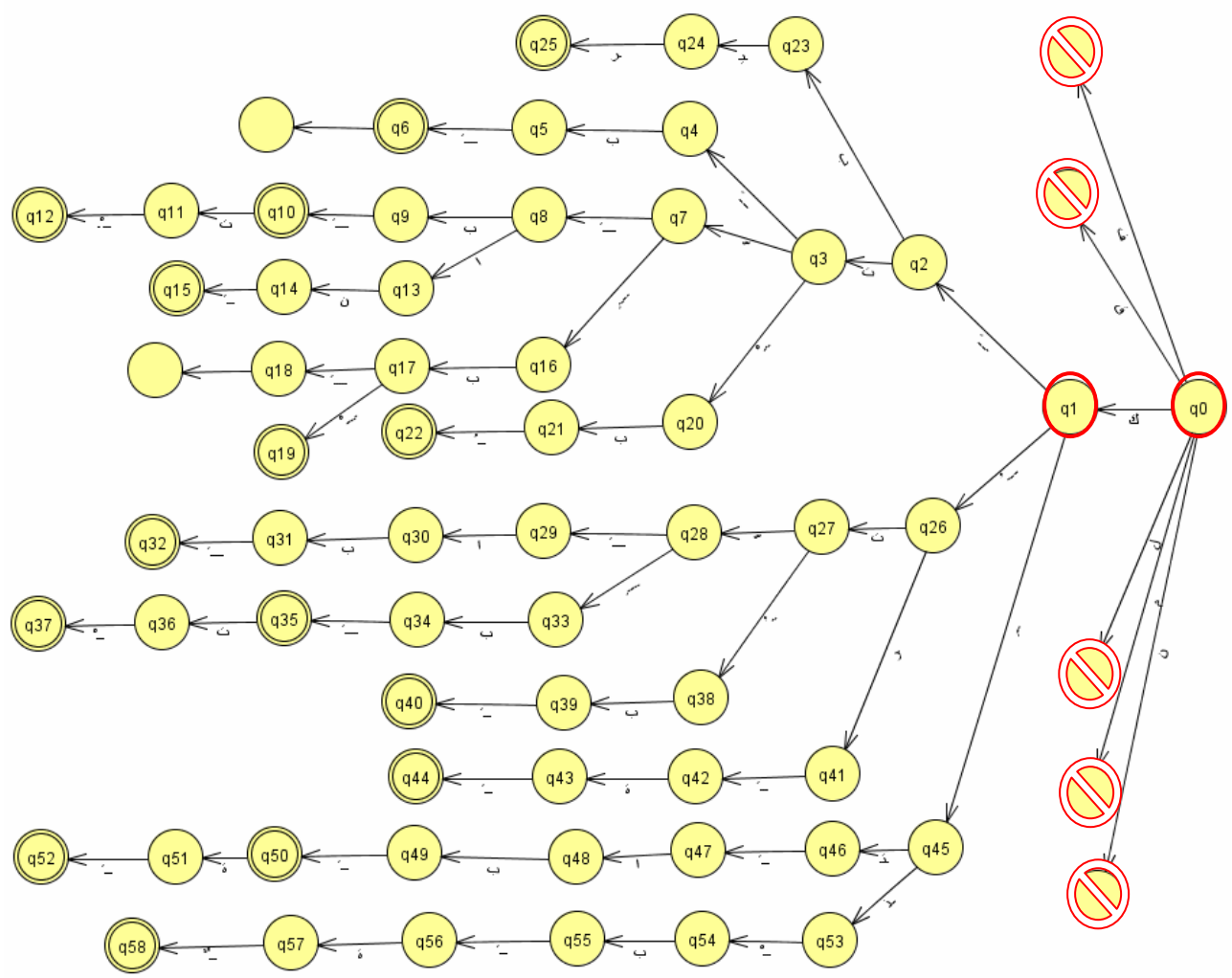
Pour illustrer notre démarche de restitution des voyelles, nous détaillons le cheminement suivi pour attester l'appartenance de la forme "كتّبَ" (*kt_tb_a*) au lexique et fournir les informations morpho-syntaxiques qui y sont rattachées. Nous allons décrire l'exemple d'une forme partiellement voyellée contenant une Šadda ainsi qu'une voyelle brève (la voyelle finale) pour deux raisons :

- montrer les répercutions de la présence de voyelles en termes de réduction du nombre des analyses fournies
- éviter la description d'un processus aboutissant à 15 voyellations différentes listées à la section 2.5.1 et qui auraient été fournies si cette forme était complètement non voyellée.

Pour faciliter la lecture et la visibilité des nœuds de cet exemple, nous avons opté pour une simplification de l'automate représenté ; celui-ci est restreint aux seize formes, commençant par la lettre ك (*k*), données comme exemple d'entrée lexicale et jugées significatives pour la description de la démarche de restitution automatique des voyelles. Nous allons lire cette entrée partiellement voyellée "كتّبَ" (*kt_tb_a*), lettre par lettre, afin de valider, à chaque étape, le chemin traversé dans l'automate :

Etape 1 :

Lecture de la première lettre de la chaîne en entrée : "ك" (*k*) → Départ du nœud initial[32] q_0 et validation de la première transition (q_0 → q_1)

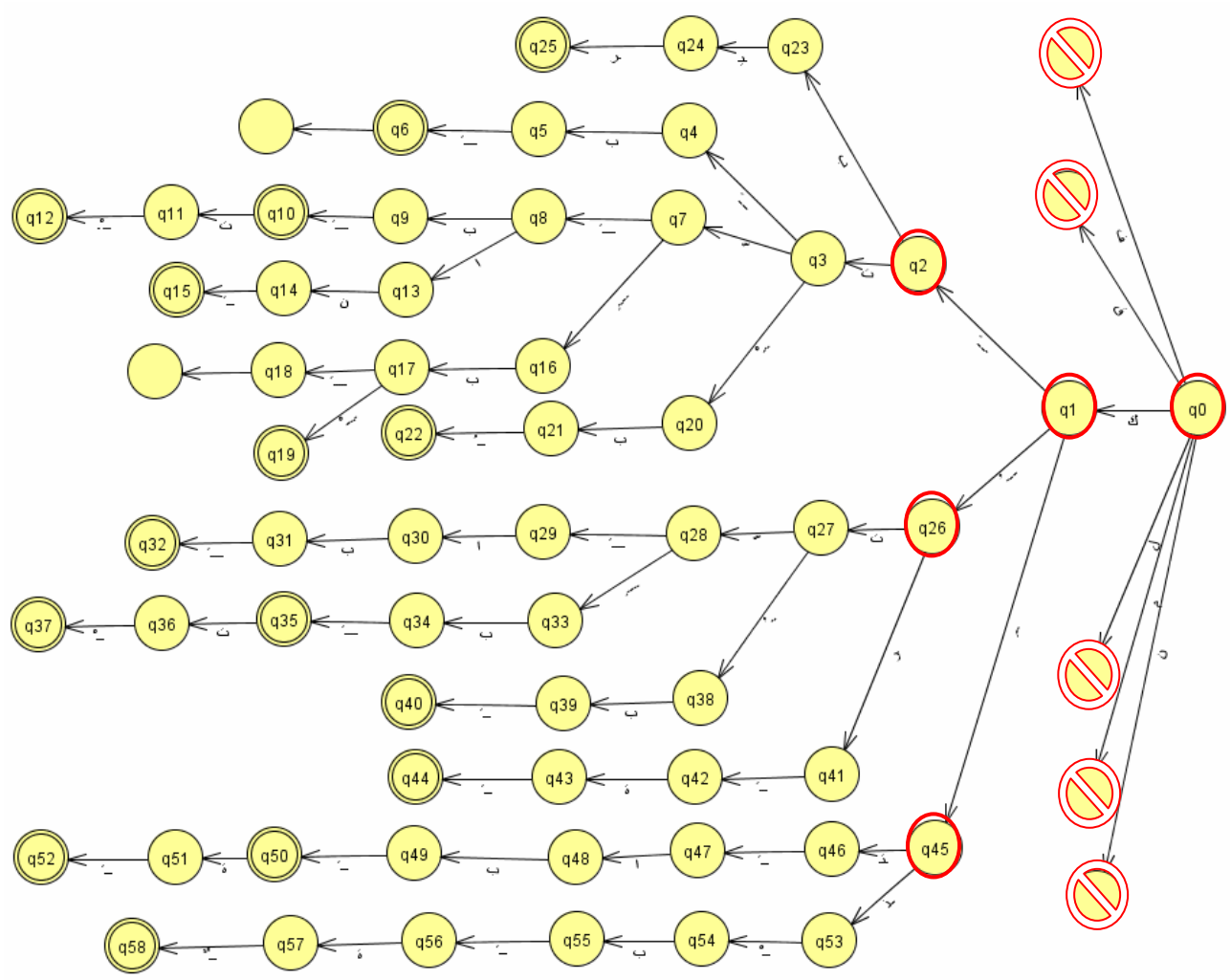


[32] Chaque nœud d'un chemin valide est entouré en rouge.

Etape 2 :

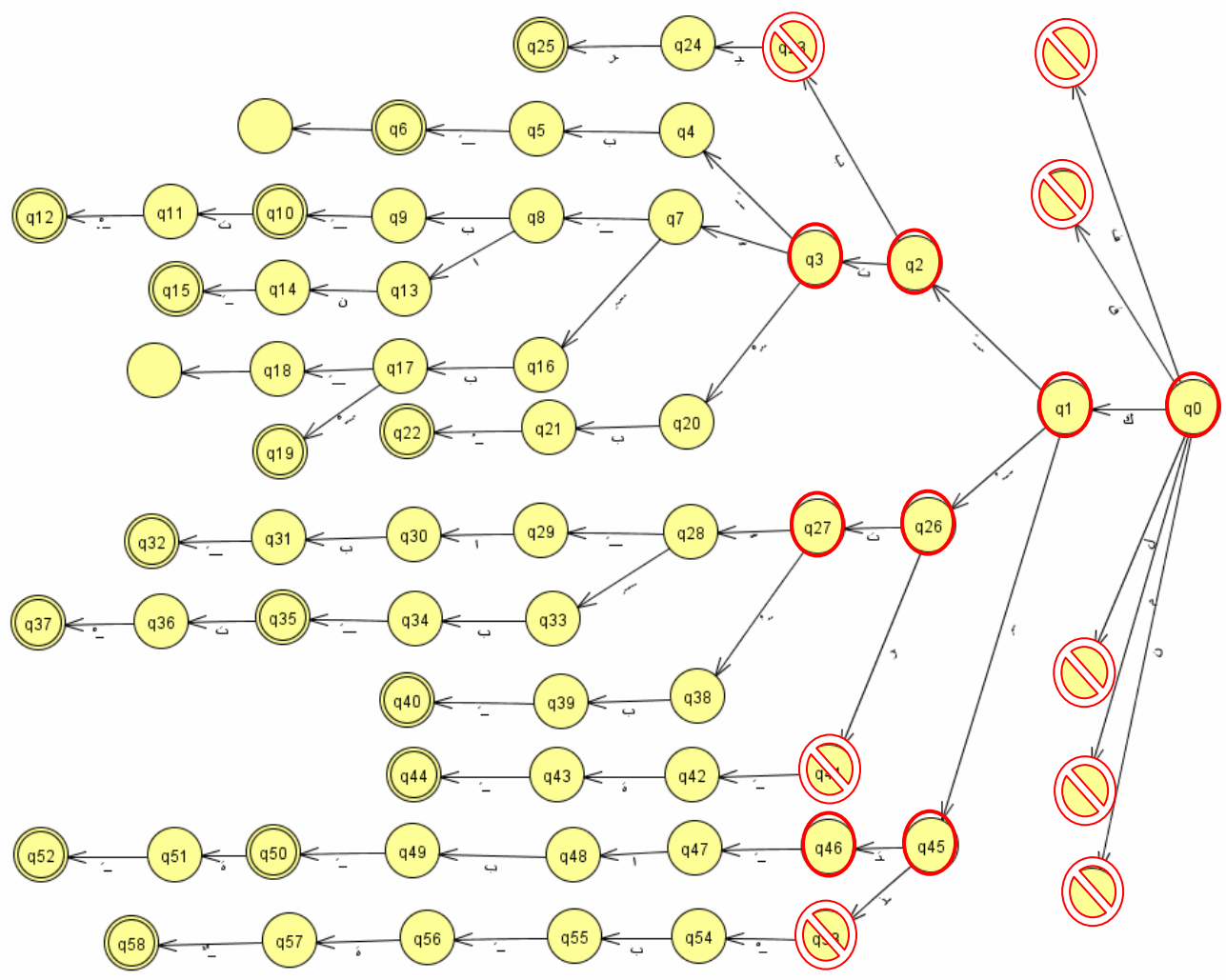
Lecture de la deuxième lettre "ت" (*t*). Cette dernière est la deuxième consonne de suite rencontrée, nous acceptons tous les arcs étiquetés par les différentes voyelles possibles tels que :

- q1 → q2 : "ـَ – فَتْحَة" (*a* - *fatḥä*) : ce chemin peut mener à : كَتَبَ (*kataba* - écrire), كَتَّبَ (*kattaba* – enseigner l'écriture), كَتِّبْ (*kattib* – enseigne l'écriture !), كَتْب (*katb* – un écrit), كَتَّبَتْ (*kattabat* – elle a enseigné l'écriture), كَتَّان (*kattaān* – lin, plante), كَبِير (*kabiyr* – grand), etc.
- q1 → q26 : "ـُ – ضَمَّة" (*u* - *ḍammä*): ce chemin peut mener à : كُتِبَ (*kutiba* – a été écrit), كُتِّبَ (*kuttiba* – il a été enseigné l'écriture), كُتُب (*kutub* – des livres), كُتِّبَتْ (*kuttibat* – elle a été enseignée l'écriture), كُتَّاب (*kuttāb* – école coranique), كُرَة (*kurat* – une balle), etc.
- q1 → q45 : "ـِ – كَسْرَة" (*i* - *kasrä*): ce chemin peut mener à : كِتَاب (*kataāba* - écrire), كِتَابَة (*kattaābat* –écriture), كِذْبَة (*kiḏbat* – un mensonge), etc.



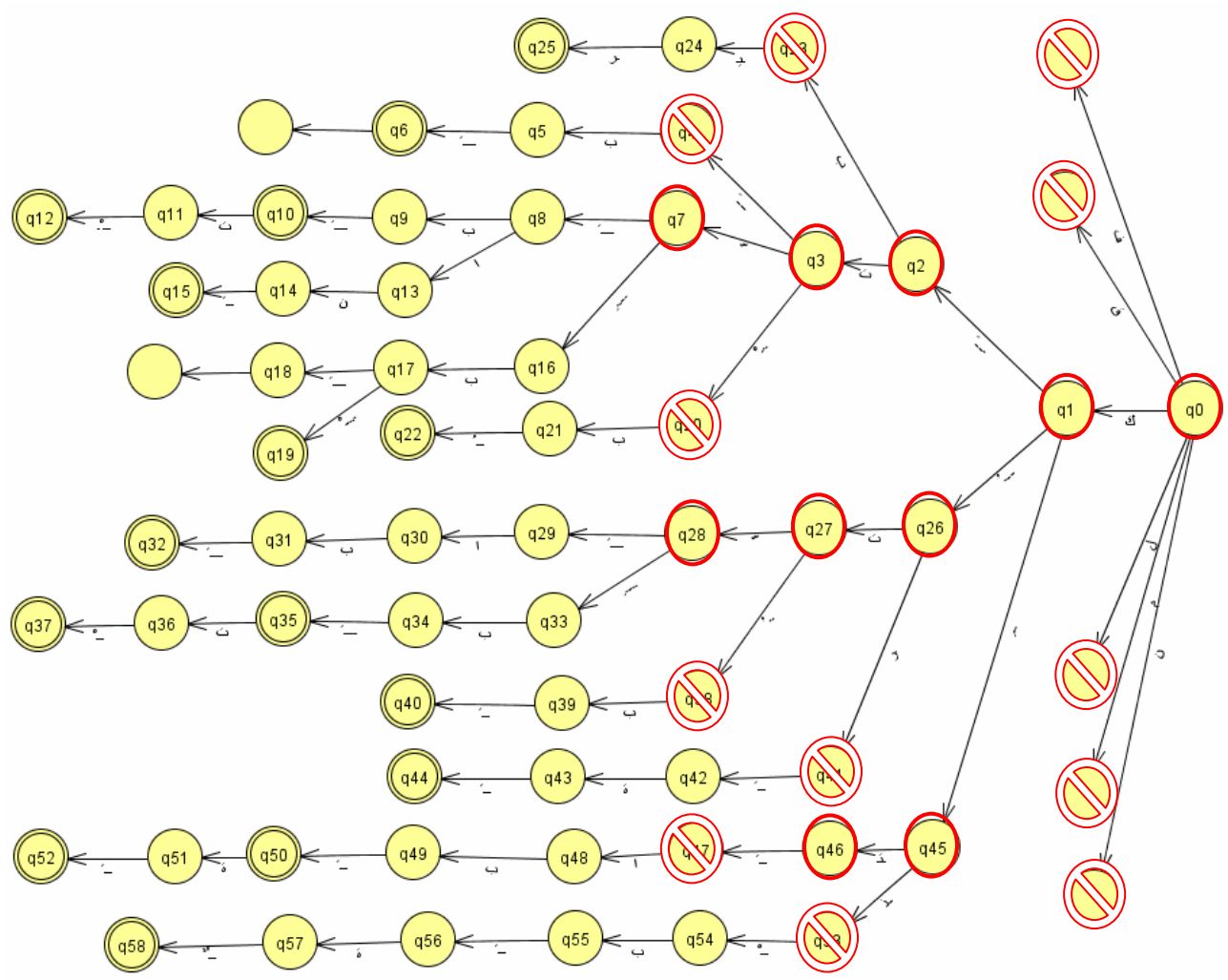
Etape 2' :

Poursuite du parcours de l'automate en ne validant que les différentes transitions ayant comme étiquette le symbole déjà lu "ت". Nous validons, alors, les transitions : q2 → q3, q26 → q27 et q45 → q46. Toutefois, nous écartons, automatiquement, tous les chemins de l'automate qui conduisaient à des discordances au niveau de la deuxième consonne. En conséquence, nous éliminons les transitions : q2 → q23, q26 → q41 et q45 → q53 et nous excluons les formes: كَبِير (*kabiyr* – grand), كُرَة (*kurat* – une balle) ou كِذبَة (*kidbat* – un mensonge).



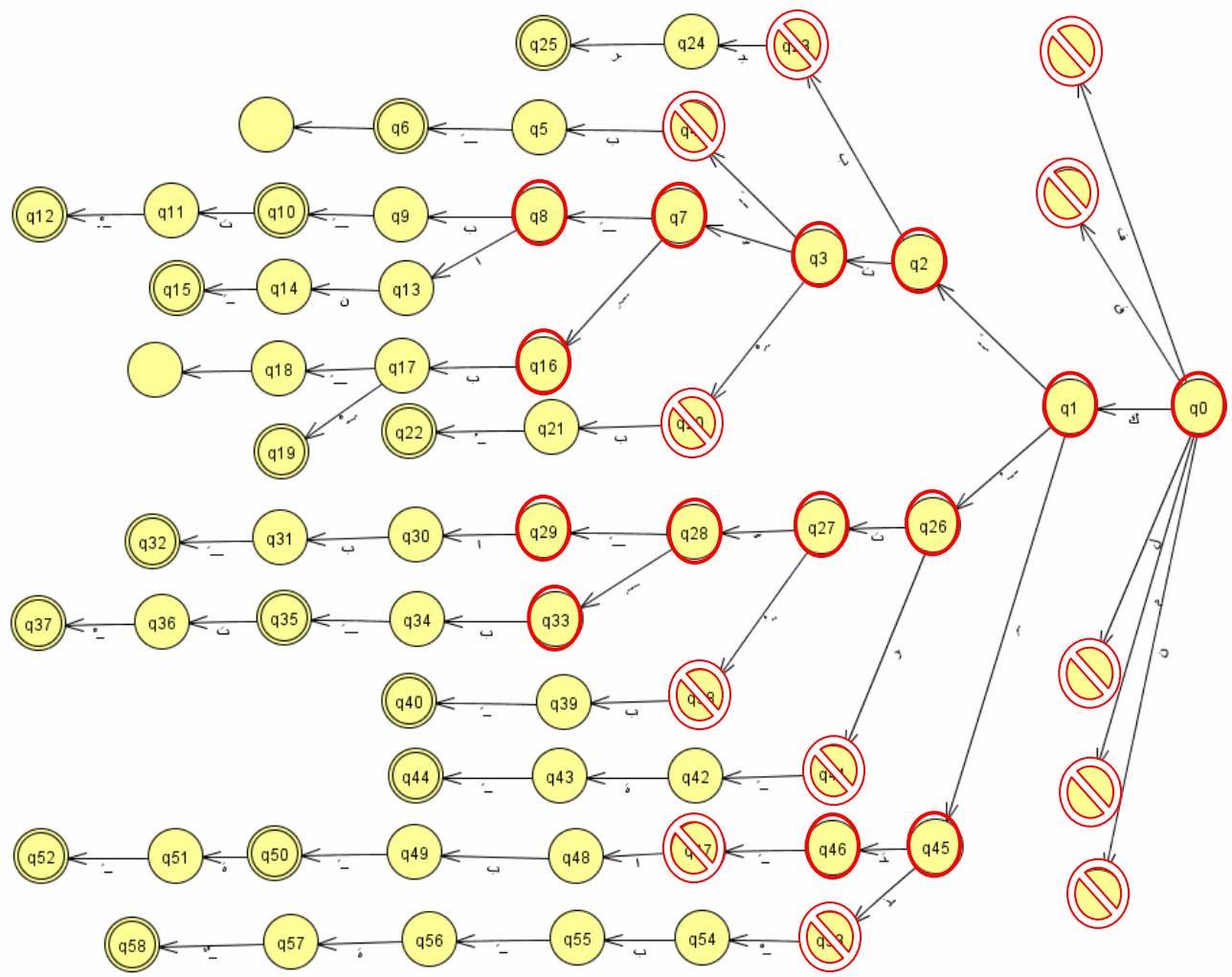
Etape 3 :

Lecture du caractère suivant, la Šadda : "ّ". Cette voyelle nous contraint à abandonner tous les chemins qui ne donnent pas la possibilité d'avoir une šadda à la suite de la deuxième consonne "ت" (*t*), ceci est équivaut au dédoublement de la lettre *t* dans les formes translittérées. Donc, la lecture de cette lettre nous contraint à exclure plusieurs chemins : d'une part, ceux qui commencent par la succession : q0 → q1 → q45 → q46 et ce à cause de l'absence de formes stockées dans l'automate ayant une "ـِ– كَسْرَة" (*i - kasrä*) sur la première consonne et une *Šadda* sur la deuxième. D'autre part, nous écartons tous les itinéraires qui devraient accepter des formes sans redoublement telles que : كَتَبَ (*kataba* - écrire), كَتْب (*katb* – un écrit), كُتِبَ (*kutiba* – a été écrit), كُتُب (*kutub* – des livres), كِتَاب (*kataāba* - écrire), كِتَابَة (*kattaābat* –écriture), كِذْبَة (*kidbat* – un mensonge). Ainsi, nous ne gardons que les chemins qui mènent aux formes : كَتَّبَ (*kattaba* – enseigner l'écriture), كَتِّبْ (*kattib* – enseigne l'écriture!), كَتَّان (*kattaān* – lin, plante), كُتِّبَ (*kuttiba* –a été enseigné l'écriture), كُتَّاب (*kuttuāb* – école coranique), etc.



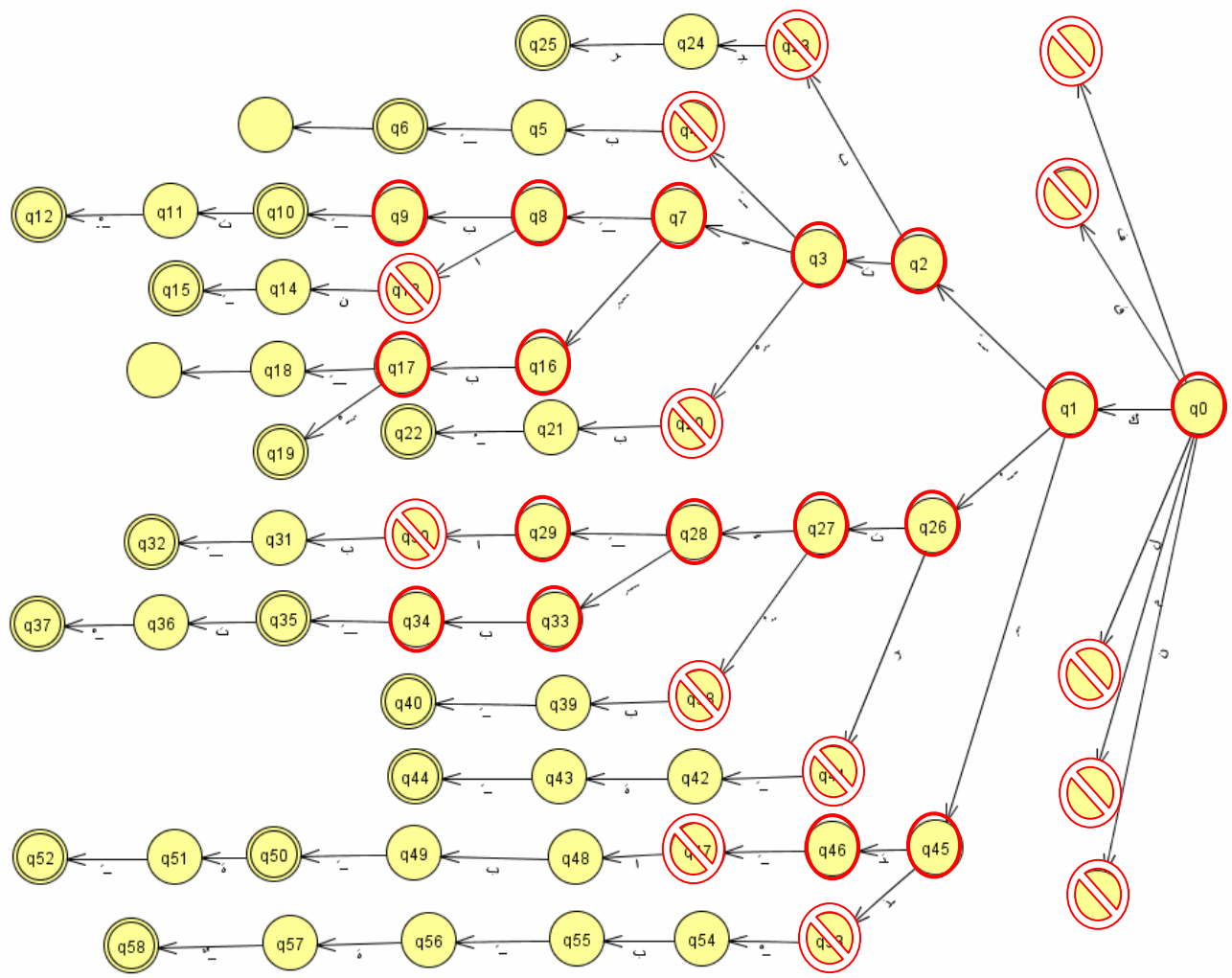
Etape 4 :

Lecture de la troisième consonne "ب" (*b*). En l'absence de la voyelle nécessaire entre la šadda et cette troisième consonne, nous acceptons toutes les transitions étiquetées par une voyelle et partant des états q7 et q28. Nous atteignons, ainsi, les états q8, q16, q29 et q33.



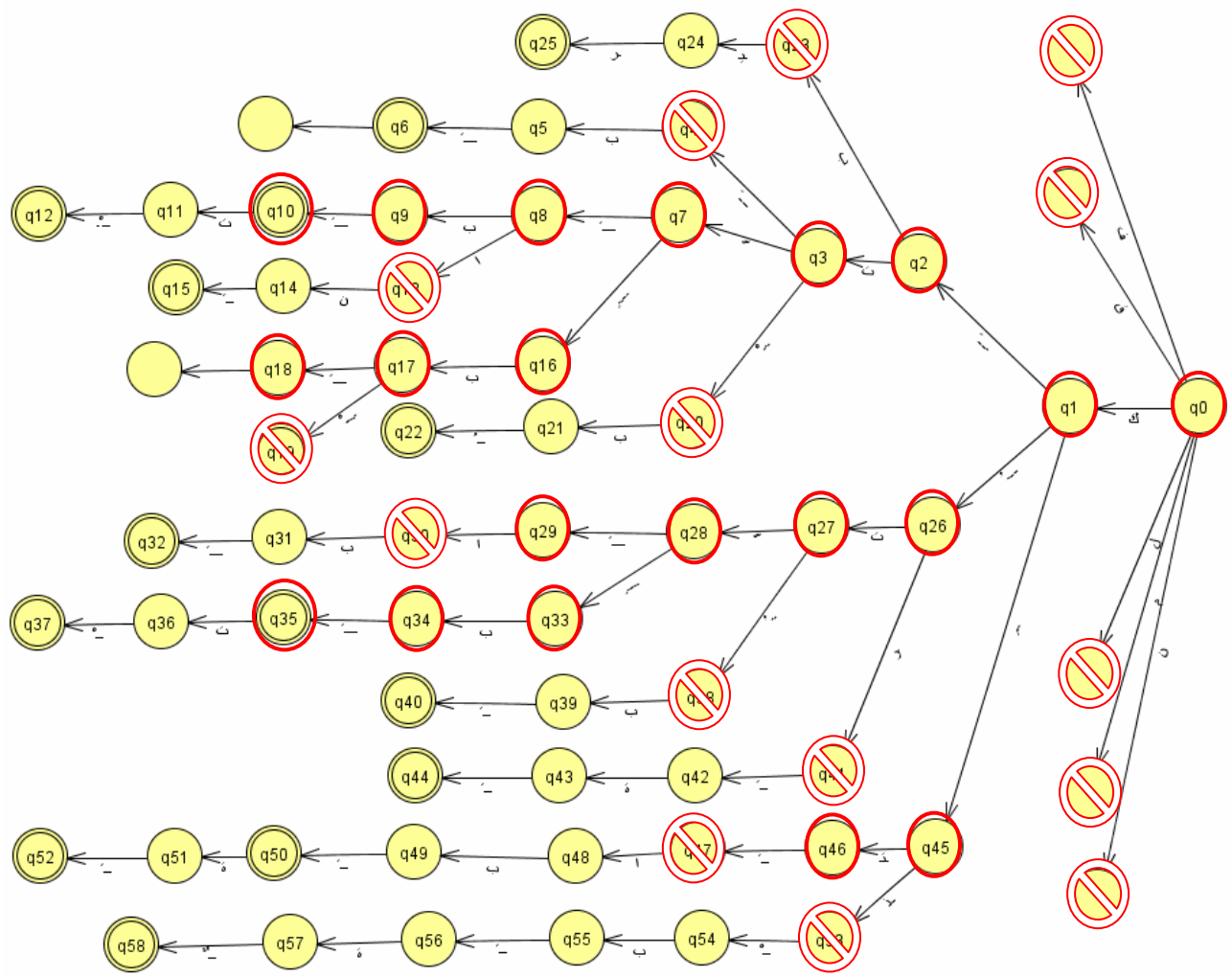
Etape 4' :

Restriction de la validation aux chemins donnant lieu à la lettre "ب" (*b*) à la suite des transitions précédemment validées. Ainsi, nous excluons les deux transitions : q8 → q13 débouchant sur la forme كَتَّان (*kattaān* – lin, plante) et q29 → q30 de la forme كُتَّاب (*kuttuāb* – école coranique).



Etape 5 :

Le symbole dernièrement lu est le dernier caractère dans la forme en entrée. Ce symbole étant une consonne, nous procédons à la validation de toutes les transitions étiquetées par une voyelle et atteignant un état final, représenté par un double-cercle, soit les chemins reconnaissant les deux formes كَتَّبَ (*kattaba* – enseigner l'écriture) et كُتِّبَ (*kuttiba* – a été enseigné l'écriture). Cependant, en dépit de la validité de la transition q17 → q18 (étiquetée par la voyelle "ـَ" (a)), le chemin traversé ne peut être accepté car l'état q18 n'est pas un état final.



A l'issue de ce parcours de l'automate, nous renvoyons la liste de tous les mots ayant une structure morphologique ressemblante à celle de la forme en entrée et nous fournissons pour chacune l'ensemble des informations morpho-syntaxiques qui s'y rattachent. Dans la section suivante, nous décrivons une phase de désambiguïsation prévue pour la réduction du nombre d'analyses proposées.

5.3.3 Réduction de l'ambiguïté lexicale liée à la restitution des voyelles

L'ambiguïté est l'un des problèmes centraux d'une analyse morpho-syntaxique. Les analyseurs se trouvent fréquemment confrontés à des situations d'ambiguïté à tous les niveaux de l'analyse que ce soit au niveau lexical, syntaxique, voire sémantique :

- **au niveau lexical :** l'ambiguïté est liée à la segmentation en unités lexicales et surtout à l'homographie polycatégorielle que le processus d'analyse se heurte ;
- **au niveau syntaxique :** l'ambiguïté est plutôt liée à la richesse des constructions syntaxiques et à leurs interprétations multiples ;
- **au niveau sémantique :** l'ambiguité est plutôt liée à la possibilité de faire correspondre au moins deux sens distincts à la même forme.

Ces problèmes d'ambiguïtés sont engendrés, majoritairement, par les phénomènes morpho-syntaxiques spécifiques à la langue arabe tels que la voyellation, la flexion et la dérivation, l'agglutination (voir section 2.5). Dans cette section, nous nous intéressons plus particulièrement à l'ambiguïté lexicale liée à la restitution automatique des voyelles.

A l'issue de l'analyse lexicale, nous attribuons à chaque mot du texte traité l'ensemble des interprétations liées aux voyellations potentielles. Une étude menée sur notre corpus nous a montré que les mots qui le constituent sont très ambigus. De nombreuses ambiguïtés sont gênantes car elles affectent des mots fréquents. Dans certains cas, la simple exploration des contextes des formes ambigües pourrait résoudre le problème. Nous avons, alors, opté pour aborder le problème de la levée d'ambiguïté à l'aide de grammaires locales NooJ de désambiguïsation. Ces grammaires locales sont fondées sur le fait que la langue est redondante [Silberztein, 1993]. En effet, il suffit de rencontrer une forme relativement ambiguë dont le contexte, droit ou gauche, pourrait imposer des contraintes pour déduire certaines informations précises et réduire l'ambiguïté. En conséquence, toutes nos grammaires locales de désambiguïsation développées sont construites à partir de règles linguistiques faisant intervenir le ou les mots cibles en contexte. Elles agissent par rapport au contexte d'apparition d'un mot du texte pour lui attribuer l'étiquette la plus adéquate et réduire ainsi le nombre d'annotations qui lui ont été associées par analyse morpho-lexicale.

Par exemple, il suffit de reconnaître la séquence morpho-syntaxique <PREP> <N> (une préposition <PREP> suivie d'un nom <N>), pour déduire que la flexion casuelle de ce nom doit être réduite au génitif. Ceci peut s'écrire sous forme d'une expression rationnelle :

```
<PREP> <N>/<N+i>
```

Cette expression rationnelle est utilisée pour signaler que parmi les annotations des formes nominales (<N>) qui sont précédées par une préposition (<PREP>) nous ne retenons que celles avec une flexion casuelle au génitif (<N+i>). Cette expression peut, aussi, s'écrire sous forme d'une grammaire où :

- les symboles à l'intérieur des nœuds désignent les étiquettes des formes en entrée ;
- les symboles en-dessous des nœuds représentent les annotations qui servent à réduire l'ambiguité en éliminant toutes celles qui n'y sont pas compatibles.

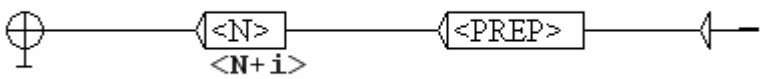


Figure 28 : Grammaire de désambiguïsation des formes nominales : Préposition suivie d'un nom

En se basant sur le même procédé de désambiguïsation, nous avons construit des grammaires locales pour la réduction de l'ambiguïté de quelques constructions autour des formes verbales.

Dans ce qui suit, nous illustrons notre démarche par le biais de l'exploration contextuelle de la forme non voyellée "أخرج" (*'ahrğ*).

Dans son état de non vocalisation, la forme "أخرج" (*'ahrğ*) peut accepter une trentaine d'analyses différentes :

- 22 formes simples fléchies : ces annotations sont attribuées directement par consultation du dictionnaire. Nous en donnons quelques-unes dans le tableau ci-après :

Voyellation	Translittération	Traduction
أَخْرُجُ	*'ahruğu*	Je sors (1ère personne du singulier, inaccompli, voix active)
أَخْرُجَ	*'ahruğa*	Je sors (1ère personne du singulier, subjonctif, voix active)
أَخْرُجْ	*'ahruğ*	Je sors (1ère personne du singulier, apocopé, voix active)
أُخْرَجُ	*'uhrağu*	On me fait sortir (1ère personne du singulier, inaccompli, voix passive)
أُخْرَجَ	*'uhrağa*	On me fait sortir (1ère personne du singulier, subjonctif, voix passive)
أُخْرَجْ	*'uhrağ*	On me fait sortir (1ère personne du singulier, apocopé, voix passive)
أَخْرَجَ	*'ahrağa*	Il a fait sortir (3ère personne du singulier, accompli, voix active)
أُخْرِجَ	*'uhriğa*	Il s'est fait sortir (3ère personne du singulier, accompli, voix passive)
أُخْرِجُ	*'uhriğu*	Je fais sortir (1ère personne du singulier, inaccompli, voix active)
أُخْرِجَ	*'uhriğa*	Je fais sortir (1ère personne du singulier, subjonctif, voix active)
أُخْرِجْ	*'uhriğ*	Je fais sortir (1ère personne du singulier, apocopé, voix active)
أُخْرَجُ	*'uhrağu*	Je me suis fait sortir (1ère personne du singulier, inaccompli, voix passive)
أُخْرَجَ	*'uhrağa*	Je me suis fait sortir (1ère personne du singulier, subjonctif, voix passive)
أُخْرَجْ	*'uhrağ*	Je me suis fait sortir (1ère personne du singulier, apocopé, voix active)

Tableau 19 : Extrait de la liste des analyses de la forme non voyellée "أخرج"

- 8 annotations en tant que forme agglutinée formée par l'adjonction de la particule d'interrogation "أ" (*á* – est-ce-que ?) et le morphème "خرج" (*harağa*). Ce dernier peut accepter 4 analyses en tant que noms et autant d'analyses en tant que verbe.

Sur la Figure 29, nous faisons figurer une partie de cette liste d'annotations sur la structure des annotations de textes (TAS : Text Annotation Structure) dans NooJ.

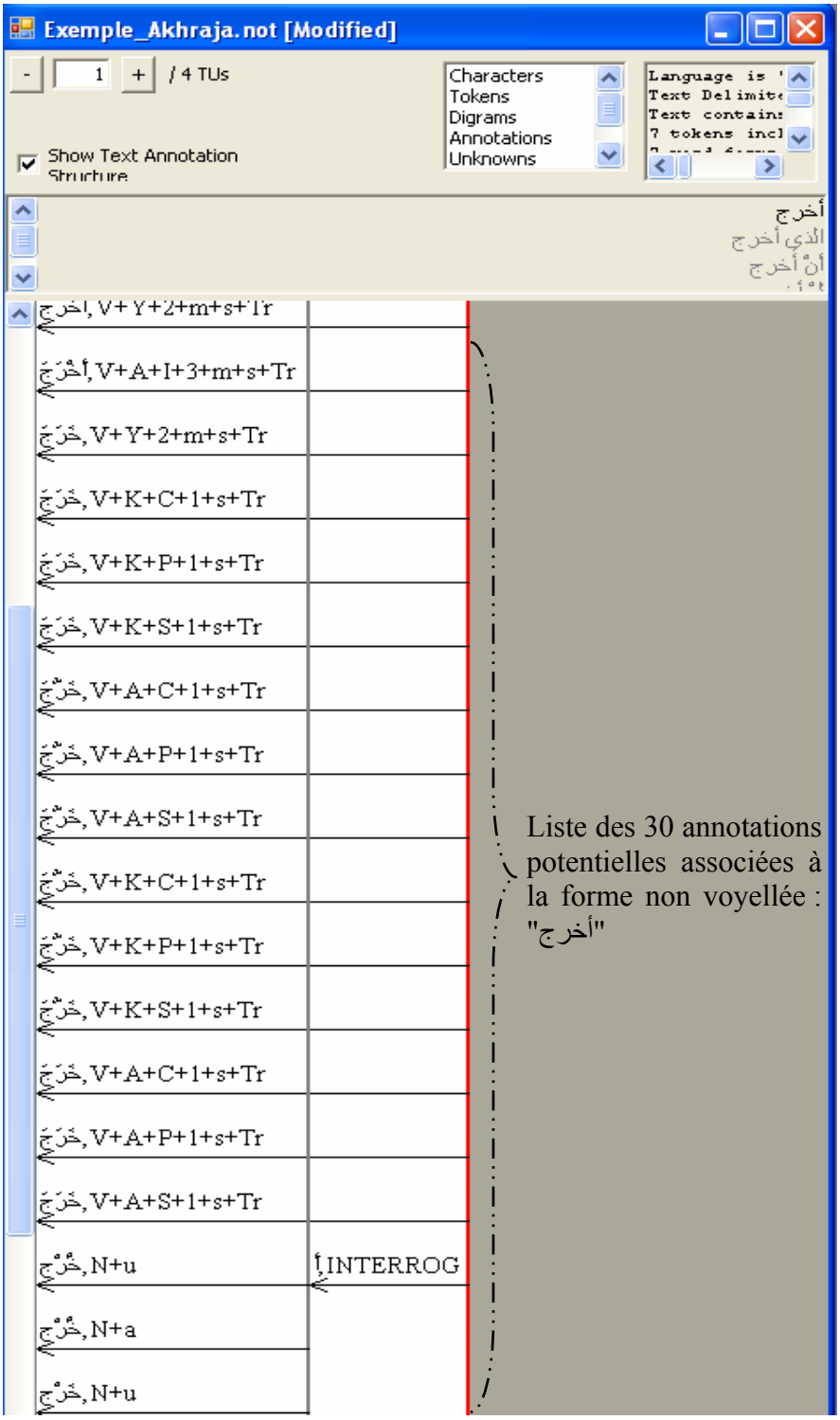


Figure 29 : Analyses de la forme non voyellée "أخرج" avant désambiguïsation

L'ambiguïté de cette forme peut être réduite par exploration du contexte antérieur. Parmi les situations les plus fréquentes dans la langue, nous pouvons citer le cas où :

- **le verbe est précédé par un relatif :**
  Les relatifs tels que"الذي" (*alladiy* – qui + masculin) et "التي" (*allatiy* – qui + féminin) sont assez contraignants car ils n'apparaissent qu'en position sujet et précisément avant le verbe. Dans ce cas, le verbe ne peut être conjugué qu'au présent de l'indicatif ou à l'accompli. Ainsi, toutes les annotations relatives à des annotations en tant que verbe au subjonctif (+S), apocopé (+C) ou impératif (+Y) doivent être écartées.

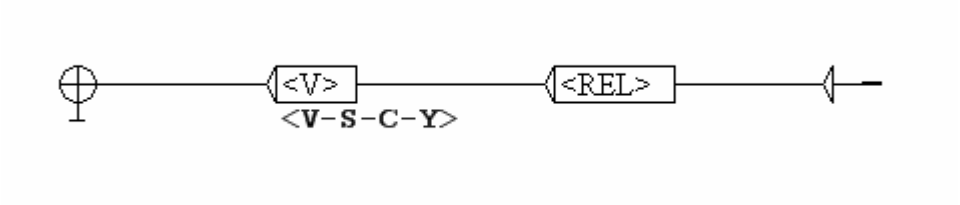


Figure 30 : Grammaire de désambiguïsation des formes verbales : Utilisation des relatifs

A l'issue de l'application de cette grammaire, nous signalons que pour chaque verbe (<V>) qui suit un relatif, désigné par le symbole <REL>, nous écartons toutes les annotations représentant une conjugaison du verbe au subjonctif (S), apocopé (C) et impératif (Y) et ce à l'aide du symbole <V-S-C-Y>. Nous ne retenons, alors, que celles représentant une conjugaison à l'accompli (+I) ou à l'inaccompli (+P).

A l'issue de l'application de cette règle de désambiguïsation, nous remarquons que nous avons réduit l'ambiguïté à plus que 73,3 % des cas. La fenêtre affichée, ci-dessous, montre qu'il y a eu suppression de 22 annotations parmi les 30 annotations potentielles initialement présentes.

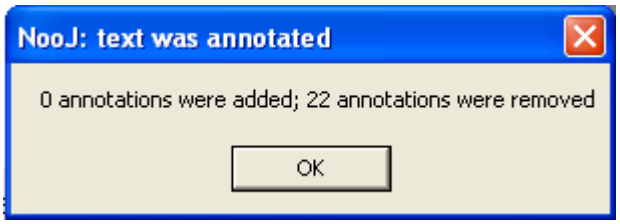


Figure 31 : Nombre d'annotations supprimées par la grammaire de désambiguïsation

Ainsi, nous ne retenons que 8 annotations (voir Figure 32), à partir des 30 annotations affichées dans la Figure 29.

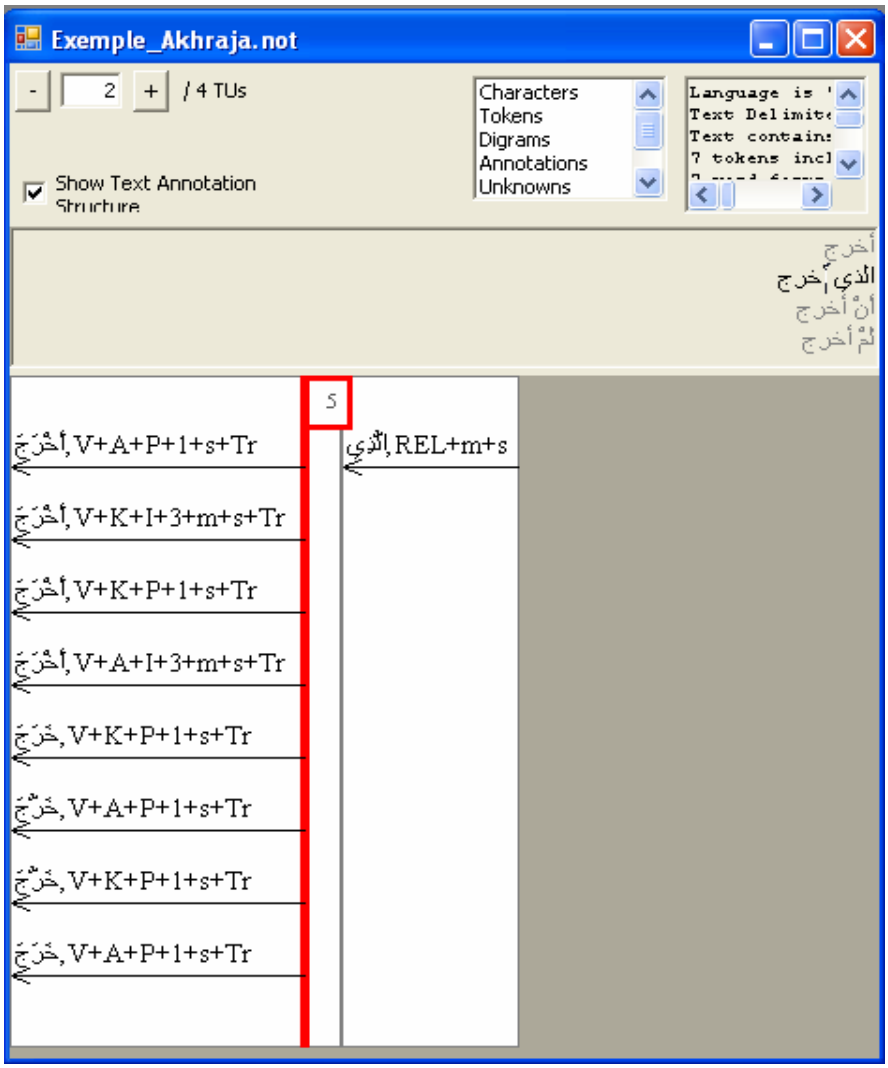


Figure 32 : Analyses de la forme non voyellée " أخرج " après désambiguïsation (1)

- **Un verbe est précédé par une particule :**
  Dans cette section, nous signalons que certaines particules qui précédent les verbes conjugués, tels que la particule du subjonctif "أنْ" (*áan* – que) ou la particule de l'apocopé "لمْ" (*lam* – ne…pas), permettent de contraindre le mode de conjugaison de ceux-ci. Pour illustrer ce propos, nous considérons les deux séquences suivantes :

  أنْ أخرج
  لَمْ أخرج

  Dans le premier exemple, le verbe est précédé par la particule de subjonctif "أنْ" (*áan* – que), il ne peut donc être conjugué qu'au subjonctif ou à l'accompli. Nous pouvons, donc, écarter toutes les conjugaisons à l'inaccompli (+P), à l'apocopé (+C) ou à l'impératif (+Y). En plus, en se basant sur une incompatibilité syntaxique des deux particules les empêchant de se suivre, nous excluons les huit segmentations potentielles faisant apparaître la particule d'interrogation, "أ" (*á* – est-ce-que ?), entre la particule du subjonctif et le verbe conjugué. Nous remarquons, ainsi, que, par la simple exploration du contexte antérieur du verbe, nous avons réussi à réduire l'ambiguïté à 73,3% des cas en passant de 30 annotations potentielles à uniquement 8 qui sont acceptables. De la même façon, cette désambiguïsation reste valable avec les trois autres

particules du subjonctif "لَنْ" (*lan* – ne…pas), "كَيْ" (*kay* – pour que) ou "حَتَّى"(*ḥattaq* – pour que). Cette situation se traduit par le graphe suivant :

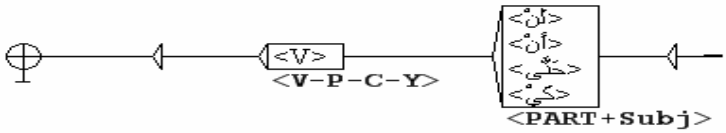


Figure 33 : Grammaire locale de désambiguïsation des formes verbales : Utilisation des particules du subjonctif

Dans cette grammaire, l'écriture du symbole <V-P-C-Y> en-dessous du symbole désignant le verbe (<V>) nous permet d'exclure toutes les annotations désignant une conjugaison à l'inaccompli (-P), à l'apocopé (-C) et à l'impératif (-Y). Ainsi, nous ne retenons que les conjugaisons du verbe au subjonctif (+S) ou à l'accompli (+I).

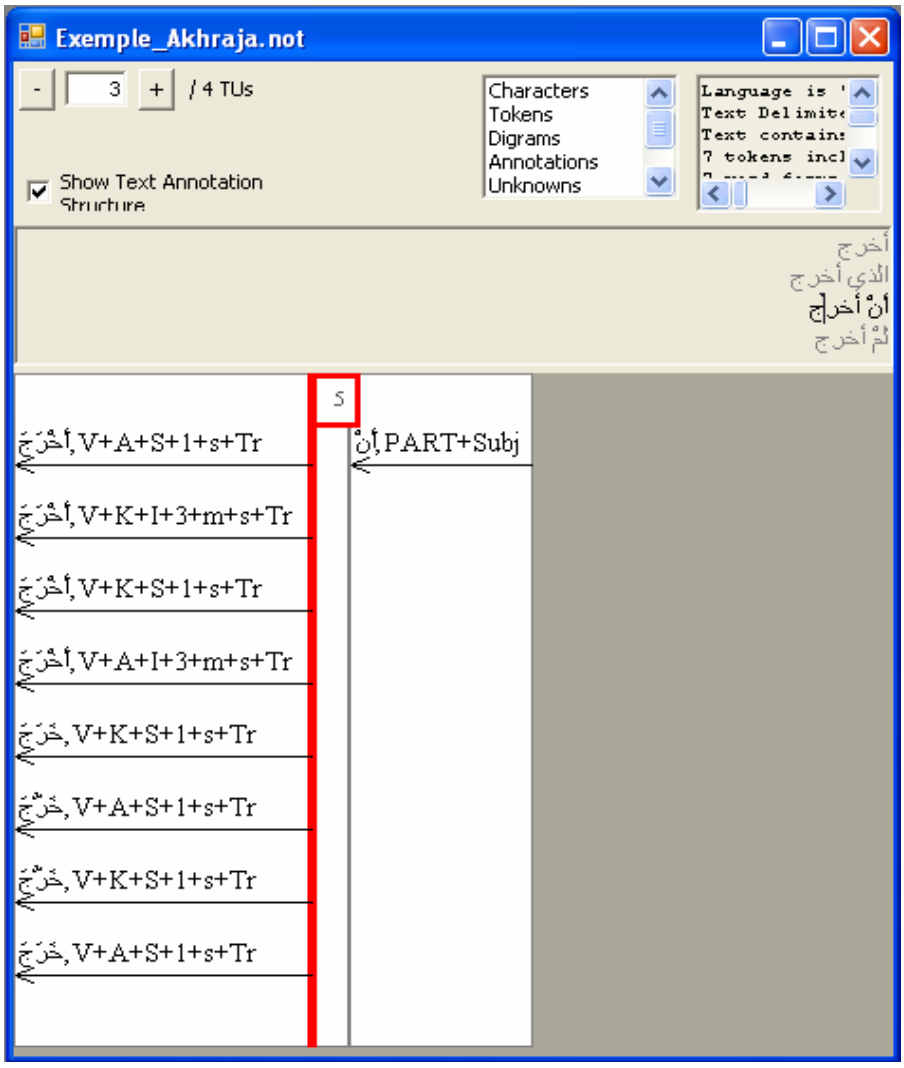


Figure 34 : Analyses de la forme non voyellée "أخرج" après désambiguïsation (2)

Pour le second exemple, le verbe est précédé par la particule de l'apocopé "لَمْ" (*lam* – ne…pas), il ne peut donc être conjugué qu'à l'apocopé. Ainsi, nous réduisons l'ambiguïté de plus que 80% en limitant le nombre d'annotations à 6 au lieu des 30 annotations possibles attribuées par analyse morpho-lexicale. Cette situation se traduit par la grammaire de désambiguïsation suivante :

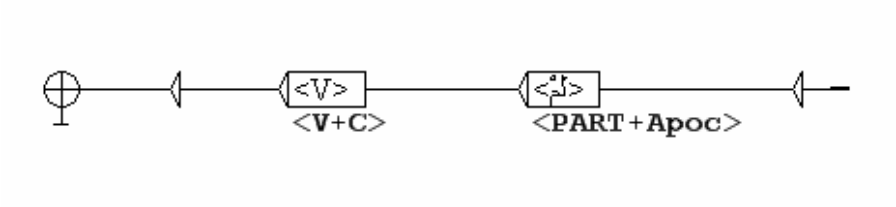


Figure 35 : Grammaire locale de désambiguïsation des formes verbales : Utilisation des particules de l'apocopé

Dans cet exemple, le symbole <V+C> désigne les annotations du verbe à l'apocopé. De la même façon que pour les particules du subjonctif, nous écartons toutes les tokenisations faisant apparaître une particule préfixée à la forme fléchie telle que la particule d'interrogation, "أ" (*á* – est-ce-que ?), la conjonction de coordination, "و" (*wa* – et) ou la particule de corroboration.

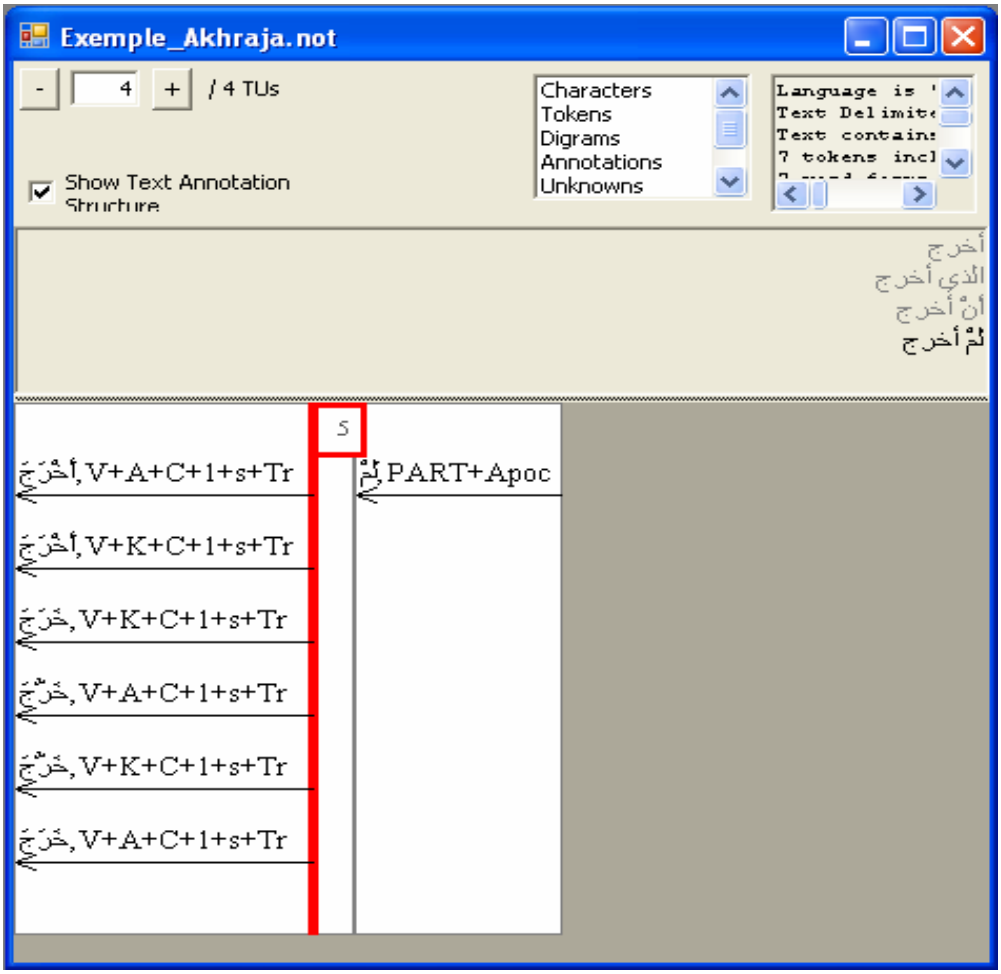


Figure 36 : Analyses de la forme non voyellée "أخرج" après désambiguïsation (3)

A l'issue de ces essais de réduction de l'ambiguïté lexicale liée à la restitution automatique des voyelles, nous convenons de signaler que cette ambiguïté ne constitue pas en soi un obstacle insurmontable du point de vue informatique. En effet, les stratégies d'analyse non-déterministes, se basant sur des réseaux de transitions récursifs (RTNs) et augmentés (ATNs), sont capables d'explorer systématiquement toutes les lectures des mots ainsi que toutes les possibilités de syntagmes spécifiées dans la grammaire. C'est plutôt la maîtrise de la combinatoire engendrée par l'ambiguïté qui pose des difficultés pour les analyseurs. Il faut aussi noter que ce n'est pas la difficulté d'analyser un langage ambigu qui constitue le problème, mais c'est plutôt la difficulté de l'analyser de façon réaliste [Ouersighni, 2002].

5.4 Correction automatique de quelques erreurs typographiques

Comme déjà signalé à la fin de la Section 5.2, plusieurs raisons peuvent amener l'analyseur automatique à réfuter un mot en entrée :

- Le mot comporte une erreur d'orthographe et il devient méconnaissable par le système ;
- Le mot n'existe pas dans le lexique.

Pour le premier cas, il serait convenable de proposer des heuristiques pour la correction de certaines erreurs très fréquentes.

Cependant, dans le second cas où un mot ne serait pas présent dans le lexique, nous pouvons difficilement lui attribuer une analyse et une catégorie de la grammaire au hasard. A priori, un mot inconnu peut en effet être aussi bien un nom, un adjectif, un verbe, … Et, si nous créons une catégorie spécifique pour les mots inconnus, il faudra l'introduire dans trop de règles, ce qui revient à multiplier dangereusement la taille de la grammaire et les possibilités d'analyse.

Compte tenu des obstacles que posent les erreurs d'orthographe au niveau de la robustesse de l'analyse automatique, les travaux ont commencé depuis les années 90 par [Ben Hamadou, 1993] et se sont poursuivis avec [Ben Othmane, 1998] et [Ouersighni, 2002]. Dans la majorité de ces travaux, la complexité des problèmes de la vérification, l'identification et la correction orthographiques ont été directement reliées à la topologie complexe des erreurs ainsi qu'à la description des modalités d'analyse. Actuellement, les systèmes de correction orthographique se sont étendus à différentes approches dont la plus populaire est celle qui inclut des ressources lexicales ainsi que des règles de repérage linguistique [Ouersighni, 2002].

Pour notre part, afin de participer à la résolution de ce problème et de remédier à une grande partie des anomalies d'analyse constatées, nous proposons :

- un ensemble d'heuristiques pour analyser des formes mal-orthographiées les plus fréquentes. Les règles de correction sont écrites sous forme de grammaires morphologiques ordonnées par le biais du système d'attribution des priorités aux différentes ressources linguistiques dans NooJ.
- un système de reconnaissance d'entités nommées pour attribuer les annotations adéquates aux noms propres non reconnus en fonction du contexte de leur apparition dans les phrases (voir chapitre 6).

A l'issue de ces deux démarches, nous avons envisagé une étape d'alimentation des dictionnaires électroniques à partir des mots restant inconnus.

Par ailleurs, après application des dictionnaires électroniques et des grammaires morphologiques de tokenisation, nous supposons que le mot inconnu pourrait être un mot mal orthographié. Cette section est consacrée à l'application de certaines heuristiques pour le traitement des erreurs orthographiques les plus fréquentes telles que la présence de la Kashida, la confusion "ا" *álif* / "أ" *hamzä* au début d'un mot, la confusion "ي" *yâ'* / "ى" *álif maqṣûrä* à la fin d'un mot, etc.

5.4.1 Traitement de la kashida ou Tatwil

La Kashida ou encore "حرف تطويل" (*ḥarf taṭwiyl*) est introduite dans les mots en arabe pour augmenter la distance entre certains caractères à l'aide d'un trait "–". Elle est souvent rajoutée pour des raisons purement esthétiques. De nos jours, avec les textes électroniques, les utilitaires de traitement de textes ont recours à la Kashida pour l'ajustement des paragraphes arabes par le biais d'une augmentation des espaces entre les caractères en vue d'une meilleure lisibilité.

Exemple : prenons le mot تتشبثون (*tatašabbaṯuwna* – vous tenez à qqchose ou qqu'un), la séquence est plus lisible quand elle est notée ainsi تـتـشـبـثـون (*ta_ta_ša_bba_tu_wna*)

L'emplacement de la Kashida dans le mot est très aléatoire, du moins selon les textes actuels. Or, ce caractère ne figure pas dans l'alphabet arabe et n'est pas pris en compte dans la construction des mots dans un dictionnaire. Les méthodes traditionnelles commencent par parcourir le mot pour éliminer la Kashida avant d'entamer l'analyse morphologique proprement dite [Abbes, 2004]. Dans notre application, grâce au stockage du lexique sous forme de machines à états finis, le traitement de la Kashida ne demande aucun traitement supplémentaire. En effet, lors de la consultation du transducteur représentant le dictionnaire, nous lisons le mot caractère par caractère et nous essayons de poursuivre le parcours de ce transducteur en tenant compte de l'étiquette, le caractère lu et les possibilités de transitions partant de l'état en cours (voir section 5.3.2). Lorsque le caractère lu est une Kashida, il suffit d'envisager une ε-transition de l'état en cours vers lui-même. En d'autres termes, nous passons à la lecture du caractère suivant sans avancer dans le parcours du transducteur.

5.4.2 Traitement de la confusion "ا" *álif* / "أ" *hamzä*

Lors de nos expérimentations et de nos tests d'évaluation de la couverture des ressources linguistiques développées, nous avons constaté que, dans la majorité des textes, il y a une confusion entre les deux lettres, le *álif* ou *hamzä wasliyya* "ا" et la *hamzä qaṭ'iyya* ("أ", "إ"), quand elles occupent la première consonne de la forme de base. En effet, la *hamzä wasliyya* "ا" prend souvent la place de la *hamzä qaṭ'iyya* initiale. Cette situation ne relève d'aucune règle en arabe et ne repose sur aucune tradition. Toutefois, ceci est une erreur d'orthographe et tout système de traitement automatique considère que chaque mot incluant une telle confusion est erroné et ne peut le traiter comme un mot arabe existant dans le lexique.

Puisqu'il est inconcevable que nous introduisions de telles informations dans le lexique, nous avons construit une grammaire morphologique à faible priorité qui permet de remplacer, au sein des mots inconnus, le *álif* par l'une ou l'autre des *hamzäs* pour essayer de retrouver l'orthographe telle qu'elle devrait l'être dans le lexique.

Ceci garantit un maximum de reconnaissance dans le corpus. Par exemple, nous trouvons souvent le mot "ان" inconnu dans une analyse automatique à moins de remplacer le *álif* par une *hamzä*. Mais est-ce que c'est "أن" (*án* - que), "إن" (*ín* - si), "أنَّ" (*ánna* - que), "إنَّ" (*ínna*), le verbe "أنَّ" (*ánna* - gémir) ou l'un de ses allomorphes ? La question reste entièrement ouverte surtout que les quatre premières formes sont très fréquentes dans la langue. Actuellement, nous acceptons toutes analyses potentielles quitte à pouvoir désambiguïser par la suite et éliminer celles qui ne correspondent pas. A ce propos, nous notons qu'au cas où la forme est partiellement voyellée, la présence d'une voyelle à la suite du *álif* peut contraindre le choix précis de la *hamzä* adéquate :

- Si le *álif* est suivi par une "ـَ – فَتْحَة" (*a* - *fatḥä*), la *hamzä* prend la forme "أ" (*á* - *hamzä*). Par exemple, la forme erronée اَبناء (*āabnaā'*) sera transformée en أبناء (*áabnaā'* – les fils de) ;
- Si le *álif* est suivi par une "ـُ – ضَمَّة" (*u* - *ḍammä*), la *hamzä* prend la forme "أ" (*á* - *hamzä*). Par exemple, la forme erronée اُخُوّة (*āuḫuwwat*) sera transformée en أُخُوَّة (*áuḫuwwat* –fraternité) ;
- Si le *álif* est suivi par une "ـِ – كَسْرَة" (*i* - *kasrä*), la *hamzä* prend la forme "إ" (*í* – *hamzä*). Par exemple, la forme erronée اِقامة (*āiqaāmat*) sera transformée en إقَـامَة (*íiqaāmat* –résidence).

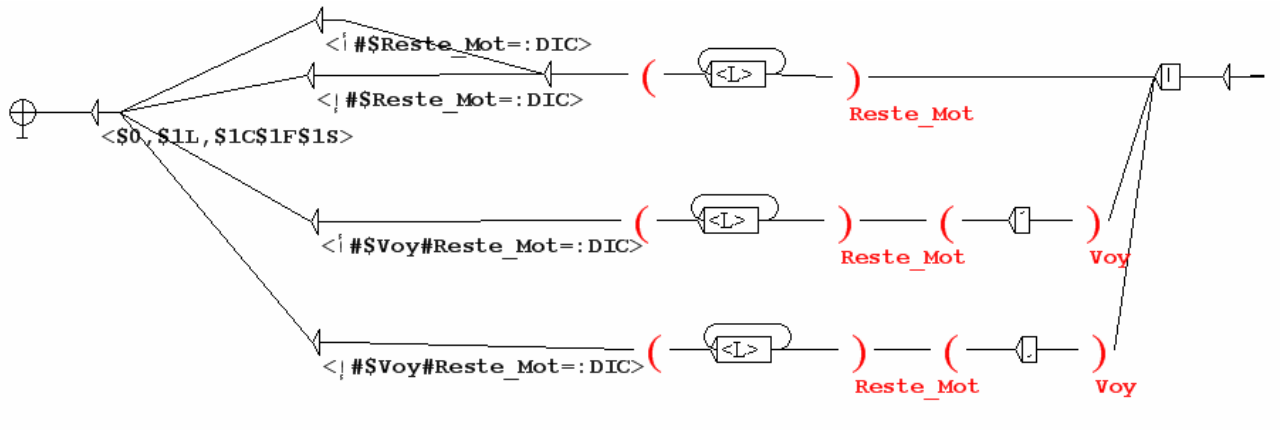


Figure 37 : Grammaire morphologique de correction orthographique : Traitement de la confusion "ا" álif / "أ" hamzä

Dans cette grammaire de correction orthographique, nous faisons usage des contraintes lexicale de type <أ#$Reste_Mot= :DIC> où <DIC> désigne toutes les formes pouvant appartenir aux dictionnaires électroniques. Ce symbole nous permet de contourner le listage de tous les codes grammaticaux tels que <N>, <V>, <PREP>, etc.

En l'occurrence, la forme "الى" (*ilaq* – jusqu'à) est présente 10 000 fois dans notre corpus ; elle doit être considérée comme une erreur fréquente afin de l'analyser comme étant la forme "إلى" et lui attribuer sa catégorie grammaticale en tant que « Préposition ». Ceci nous permettra de limiter les répercutions du refus de l'analyse d'une telle forme fréquente sur les tâches d'analyse automatique en cours.

5.4.3 Traitement de la confusion "ي" *yâ'* / "ى" *álif maqṣûrä*

Une autre confusion très similaire à la précédente concerne la totale confusion du "ي" (*y - yâ'*) et du "ى" (*q - álif maqṣûrä*) lorsqu'ils figurent à la fin d'un mot.

Nous avons remarqué que cette confusion ne peut figurer que dans un seul sens, celui du remplacement du "ي" (*y -yâ'*) par un "ى" (*q - álif maqṣûrä*). En effet, la présence du "ي" (*y - yâ'*) comme lettre terminale garantit que c'est l'orthographe voulue, excepté dans quelques éditions égyptiennes où cette confusion serait permise. Toutefois, la présence du "ى" (*q - álif maqṣûrä*) maintient une ambiguïté totale sur le mot surtout que les mots avec "ي" ont souvent un homographe avec "ى".

Pour éviter l'introduction de nouveaux cas d'ambiguïté, le système explore les possibilités d'écriture du mot avec "ي" (*y - yâ'*) uniquement si celle avec "ى" (*q - álif maqṣûrä*) ne donne pas de résultats.

Dès lors, dans le cas de l'existence de solutions avec "ى" (*q - álif maqṣûrä*), le système ne procède pas à une deuxième tentative en remplaçant la dernière lettre par "ي" (*y - yâ'*) afin d'éviter l'introduction d'ambiguïtés supplémentaires. Par exemple, pour le mot يبقى (*yabqaq* – il subsiste), analysable en tant que forme fléchie du verbe بَقِيَ (*baqiya* – subsister), nous n'appliquerons pas nos grammaires de correction bien qu'il suffirait de substituer sa dernière lettre, "ى" (*q - álif maqṣûrä*), par un "ي" (*y - yâ'*) pour obtenir une deuxième forme verbale يُبْقِي (*yubqiy* – il fait subsister).

5.4.4 Restitution de la lettre "ة" (*tâ'*) finale

Une troisième confusion très similaire aux précédentes concerne la substitution de la lettre "ة" (*ħ - tâ' marbûṭä*) avec la lettre "ه" (*h - hâ'*) lorsque celle-ci figure à la fin d'un mot.
De façon analogue au cas précédent, la commutation des deux lettres n'est permise que dans un seul sens. En effet, tandis que la rencontre de la lettre "ة" (*ħ - tâ' marbûṭä*) comme lettre terminale garantit que c'est l'orthographe voulue, la rencontre la lettre "ه" (*h - hâ'*) maintient une erreur d'orthographe potentielle sur le mot surtout si son analyse n'aboutit pas.

Toujours par souci d'introduction de nouveaux cas d'ambiguïté, le système n'explore les possibilités orthographiques du mot avec une "ة" (*ħ - tâ' marbûṭä*) qu'au cas où la forme avec une lettre "ه" (*h - hâ'*) ne donne pas de résultats.

5.4.5 Correction du Tanwin

Cette heuristique concerne la vocalisation " ً" (*an - double-fatha / tanwîn*) notée sur le "ا" (*ā - álif*) terminal plutôt que sur la dernière consonne. Par exemple, dans les textes courants, nous pouvons trouver "أَبَداً" au lieu de "أَبَدًا". Le nettoyage du mot consiste à remplacer les séquences "ـاً" par "ـًا".

Sur le plan pratique, éviter une telle confusion n'est pas vraiment compliqué. En effet, il suffit que la forme en question reste inconnue par analyse morphologique et qu'elle se termine par "ا" (*ā - álif*) suivi de la voyelle " ً" (*an - double-fatha / tanwîn*) pour lui appliquer une grammaire morphologique permettant l'inversion de ces deux derniers caractères.

Par conséquent, si nous reprenons l'exemple de la forme "الى" (voir paragraphe 5.4.2), nous pouvons affirmer que si nous ne procédons pas à ces ajustements, la proportion des mots de la langue qui resteraient non analysables risque d'être discriminante vis-à-vis des performances des étapes d'analyse suivantes telles que l'analyse syntaxique, la reconnaissance des entités nommées, la levée d'ambiguïté, etc.

Remarque : Les problèmes d'orthographes, cités ci-dessus, ont toujours été contraignants aux moteurs d'analyse automatique de l'arabe notamment lors de la recherche. Par exemple, nous trouvons l'annonce suivante sur le site d'ASSA (Arabic Social Science Research)[33] :

« You can search the titles and descriptions of ASSR resources for a word or phrase in Arabic or English. Exact match is needed in Arabic (including Hamza, final Taa' and Kashida) ».

"يمكن البحث عن كلمة أو شبه جملة في عناوين أو وصف موارد هذه البوابة باللغتين العربية أو الإنجليزية، مع ملاحظة وجوب التطابق الإملائي التام بالعربية : الهمزة، التاء المربوطة، الكشيدة، الخ."

5.5 Annotation automatique de textes

5.5.1 Analyse textuelle et annotation automatique

Après la phase de formalisation du lexique et de développement de grammaires morphologiques à large couverture, nous confrontons nos ressources linguistiques à des textes écrits en arabe standard. Cette confrontation nous permet de réviser, corriger et compléter toutes nos ressources. En effet, lors de l'application des ressources linguistiques à un corpus, NooJ fournit deux listes : celle des mots reconnus nommée « Annotations » et celle des mots non-reconnus désignée par « Unknows ».

[33] http://www.assr.org/search.asp

Par ailleurs, après toute analyse morpho-lexicale d'un texte, chacun des mots reconnus est associé aux informations qui s'y rattachent indépendamment de son contexte. Ces informations lexicales, affichés sur la liste des « Annotations », sont attribuées de deux façons différentes :

- le programme d'identification de formes simples et composées par consultation de la liste des formes fléchies possibles rattachées aux entrées du dictionnaire « El-DicAr » ;
- la routine d'application de grammaires morphologiques permettant de segmenter les formes agglutinées, repérer les différents composants de ces dernières et attester leurs appartenances à la langue.

Dans cette section, nous nous intéressons à l'analyse linguistique d'une phrase et nous fournissons la liste des annotations contenant toutes les analyses qui auront été associées aux différentes formes reconnues par consultation de dictionnaires ou par tokenisation à l'aide de grammaires morphologiques. Dans l'exemple détaillé de la Figure 38, nous proposons la phrase :

فَقَدَ الوَزِير صَوَابَهُ فِي مُدَاوَلَات مَجْلس النُّواب
faqada al-waziyr ṣawaābahu fiy modaāwalaāti mağlis en-nuwweāb
Le ministre a perdu son sang-froid lors des délibérations de la Chambre des Représentants

Cette phrase contient:

- des formes simples :
  - ✓ le verbe "فَقَدَ" (*faqada* – il a perdu) ;
  - ✓ la préposition "فِي" (*fiy* – dans, lors de) ;
  - ✓ les deux formes nominales : "مُداولات" (*modaāwalaāti* – les délibérations) et "مَجْلس" (*mağlis* – conseil / Chambre).
- trois formes agglutinées :
  - ✓ deux formes préfixées : "الوَزير" (*al-waziyr* – le ministre) → "ال" (*el* – le) + "وَزير" (*waziyr* – ministre) et " النُّواب " (*al-nuwweāb* – les représentants) → "ال" (*el* – les) + " نُواب " (*nuwweāb* – représentants) ;
  - ✓ une forme suffixée "صَوَابَهُ" (*ṣawaābahu* – son sang-froid) → "صَواب" (*ṣawaāba* – sang-froid) + "هُ" (hu – son).
- un mot composé : " مَجْلس النُّواب" (*mağlis en-nuwweāb* – Chambre des Représentants) ;
- une expression figée discontinue : "فَقَدَ .... صَوَابَهُ" (*faqada ... ṣawaābahu* – perdre son sang-froid).

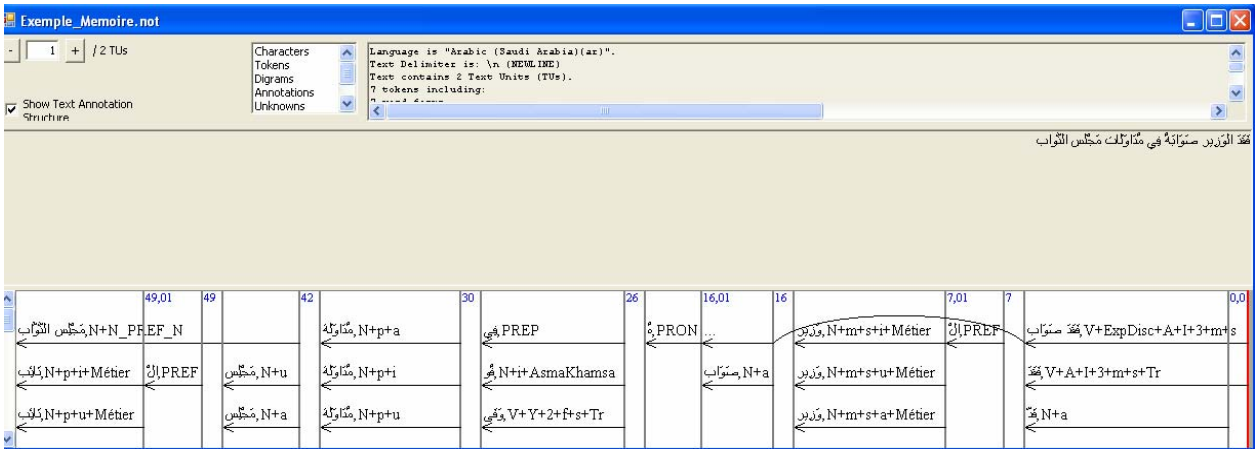


Figure 38 : Fenêtre de la Structure des Annotations d'un Texte : Text Annotation Structure - TAS

Sur le plan pratique, chaque annotation attribuée à un morphème, à un mot simple ou composé est représentée par une flèche (voir Figure 38). Toute flèche est étiquetée de façon à afficher le lemme, la catégorie grammaticale ainsi que toutes les informations flexionnelles, morphologiques et syntactico-sémantiques qui sont rattachés aux différents constituants des phrases.

Nous notons que, par rapport à l'étiquetage de cette phrase, nous rencontrons :

- des agglutinations : elles sont représentées par une succession de deux flèches tel est le cas de la forme "الوَزير" (*al-waziyr* – le ministre) décomposée en deux parties pour distinguer ces deux morphèmes : "ال" (*el* – le) en tant que préfixe (PREF) et "وَزير" (*waziyr* – ministre) en tant que nom masculin singulier d'un métier (N+m+s+Métier) avec les différentes flexions casuelles potentielles : « +a » pour l'accusatif, « +u » pour le nominatif et « +i » pour le génitif. Pareillement, la forme agglutinée " النُّواب " (*al-nuwweāb* – les représentants) dissociée en deux : l'article défini "ال" (*el* – les) et la forme nominale " نُّواب " (*nuwweāb* – représentants). Cette forme nominale est analysée en tant que participe actif (N+PA) dérivé à partir du verbe "نَابَ" (*naāba* - représenter) décliné au masculin, pluriel (+m+p) ;
- un certain nombre d'ambiguïtés :
  - ✓ le mot "فِي" est ambigu, il est reconnu comme étant :
    - une préposition : فِي,PREP (*fiy* - dans) ;
    - un nom : فُو,فِي,N+i+AsmaKhamsa (*fiy* – bouche, génitif);
    - un verbe conjugué à l'impératif : فِي,وَفَى,V+Y+2+f+s+Tr (*fiy* – sois confiant !);
  - ✓ la forme"فَقَدَ" est ambigue, elle est reconnue comme étant :
    - un verbe conjugué à l'accompli: فَقَدَ ,فَقَدَ,V+P+A+3+m+s+Tr (*faqada* – a perdu);
    - un adverbe : فَقَدْ,ADV (*faqad* – alors);
    - une forme agglutinée : فَـ + قَدَّ (*fa* + *qadda*, et + il a coupé) : conjonction de coordination + verbe à l'accompli ;
    - une forme agglutinée : فَـ + قَدُّ (*fa* + *qaddu*, et + le pouvoir) : conjonction de coordination + un nom ;
- un mot composé : il se présente comme alternative à la séquence de deux mots simples correspondantes "مَجْلس النُّواب " (*mağlis al-nuwweāb* – Chambre des Représentants) ;
- une expression figée telle que l'expression discontinue "فَقَدَ .... صَوَابَهُ" (*faqada ... ṣawaābahu* – perdre son sang-froid) : elle est représentée en tenant compte de leur discontinuité par le biais d'une flèche oblique sous forme d'un arc reliant ses deux parties.

La même liste d'annotations est fournie sous forme d'une liste facilement exploitable, comme le montre la figure ci-dessous :

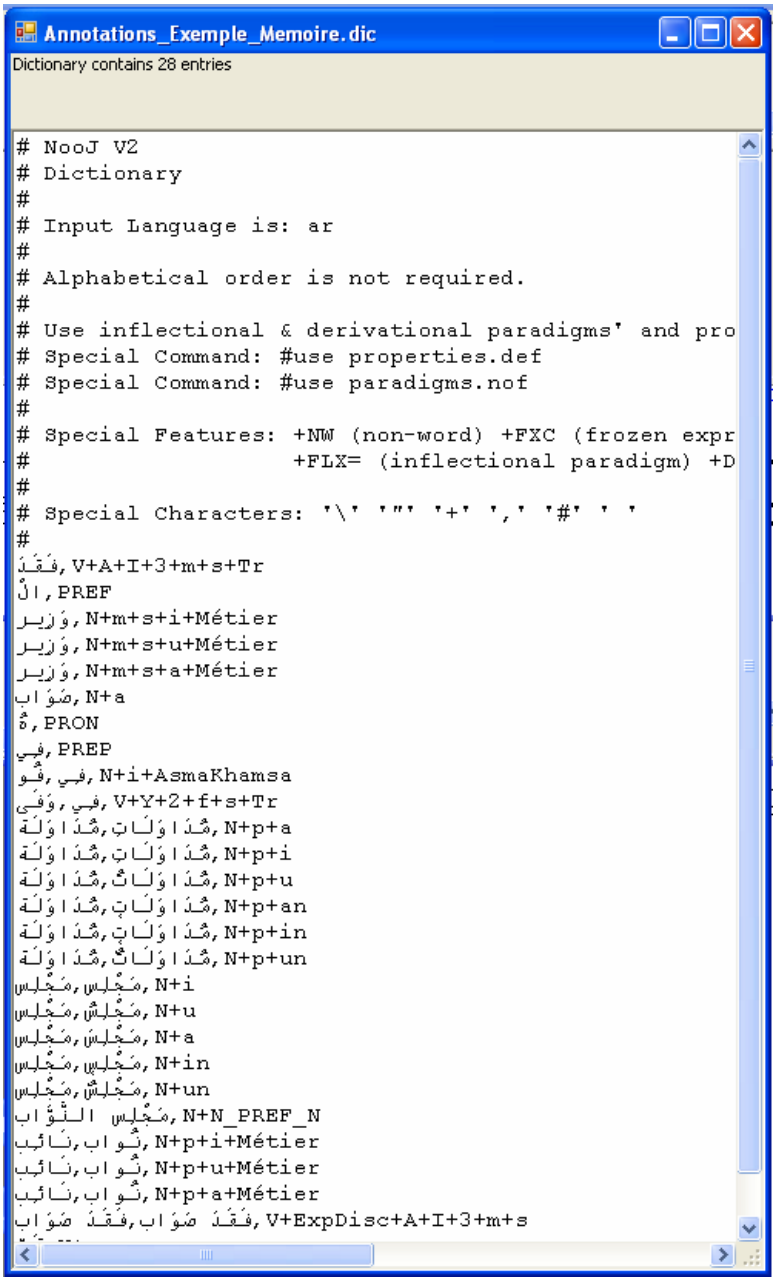


Figure 39 : Liste des annotations

Ces annotations peuvent être utilisées dans de nombreuses applications du traitement de la langue :

- Elles peuvent être fournies comme entrées d'un analyseur syntaxique. Pour cette raison, nous n'avons écarté aucun trait et nous avons gardé toutes les analyses des particules (proclitiques et enclitiques) [Abbes, 2004] ;
- Elles peuvent être fournies comme entrées à un correcteur orthographique [Ouersighni, 2002] ;
- Elles peuvent servir pour l'évaluation de ressources lexicales, c'est pourquoi nous renvoyons tous les résultats y compris les mots non reconnus ;
- Elles peuvent être utilisées dans l'enseignement des langues en fournissant les analyses potentielles à l'apprenant [Zaafrani, 2001] ;
- Elles peuvent être intégrées dans des requêtes, sous forme de symboles grammaticaux dans une application de concordance ainsi qu'un concordancier en ligne dont la réalisation sera détaillée dans le Chapitre 7.

Notons qu'aux annotations morpho-lexicales ainsi obtenues, peuvent s'ajouter des annotations syntactico-sémantiques attribuées par confrontation du texte à des grammaires locales. Ce type d'annotations sera détaillé dans le chapitre 6.

5.5.2 Recherches linguistiques et concordances

A ce niveau de traitement, tous les mots reconnus par l'ensemble des ressources lexicales et morphologiques sont associés à un ensemble d'informations linguistiques. Ces informations sont attribuées directement à partir d'une consultation du dictionnaire électronique, d'une tokenisation par le biais de grammaires morphologiques ou d'une correction orthographique à l'aide des heuristiques présentées dans la section 5.4. Dès lors, il est possible d'exploiter ces informations pour explorer les phénomènes linguistiques locaux au moyen d'expressions rationnelles ou de graphes.

En effet, outre les recherches morphologiques de surface par le bais d'expressions rationnelles (voir section 7.5.1) et la recherche des formes fléchies telles qu'elles figurent dans les textes, nous pouvons utiliser les informations morpho-syntaxiques recensées dans le dictionnaire. Celles-ci sont représentées par le biais de symboles grammaticaux ou flexionnels écrits entre des crochets (« < » et « > »).

Par ailleurs, nous utilisons la description formelle des entrées lexicales introduites dans le dictionnaire ainsi que celles générées automatiquement afin de construire nos symboles grammaticaux :

- Les formes fléchies identifiables grâce à l'écriture de leurs formes de base ou lemmes entre crochets. Par exemple :
  - ✓ le symbole <كَتَبَ> (<*kataba*>) représente toutes les formes conjuguées du verbe كَتَبَ (*kataba* – écrire), comme كَتَبَتْ (*katabat* – elle a écrit), كَتَبُوا (*katabuwā* – ils ont écrit), يَكْتُبَان (*yaktubaāni* – ils écrivent, forme duale), أُكْتُبْ (*áuktub* – écris ! forme impérative au masculin, singulier), أُكْتُبِي (*áuktubiy* – écris ! forme impérative au féminin, singulier), etc. ainsi que celles qui y sont dérivées telles que كَاتِب (*kaātib* – celui qui écrit, écrivain), كاتبات (*kaātibaāt* – celles qui écrivent, écrivains au féminin, pluriel), مكتوب (*maktuwb* – ce qui a été écrit, un écrit), مكتوبَيْن (*maktuwbayni* – ce qui a été écrit, deux écrits au masculin, dual à l'accusatif / génitif), etc. ;

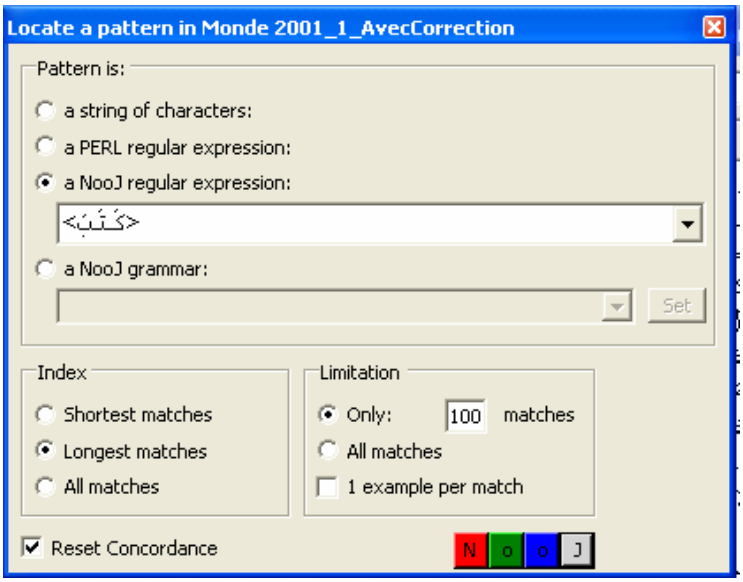


Figure 40 : Fenêtre de recherche linguistique : « Locate a pattern »

Figure 41 : Table de concordances de la requête : <كَتَبَ>

- ✓ le symbole <كِتَاب> (*<kitaāb>*) représente toutes les formes nominales qui y découlent, comme كِتَابُ (*kitaābu* – un livre au nominatif, défini), كِتَابًا (*kitaāban* – un livre à l'accusatif, indéfini), كِتَابَيْنِ (*kitaābayni* – deux livres à l'accusatif ou génitif, défini), كُتُبٌ (*kutubun* – des livres au nominatif, indéfini), etc.

Figure 42 : Table de concordances de la requête : <كِتَاب>

A ce propos, nous soulignons notre prise en compte de l'ambiguïté de certaines formes, c'est-à-dire qu'elles représentent plusieurs emplois distincts. Par exemple, le mot ذهَب est ambigu : il représente soit le substantif ذهَب (*ḏahab* – or), soit le verbe ذهَب (*ḏahab* – aller). Dans ce cas, le symbole <ذهَب> (<*ḏahab*>) représente aussi bien les formes fléchies du nom ذهَب (*ḏahab* – or) telles que ذَهَبًا (*ḏahaban* – or à l'accusatif, indéfini), أذْهَابٌ (*áḏhaābun* – ors : forme plurielle au nominatif, indéfinie), ذُهُوبٍ (*ḏuhuwbin* – ors : forme plurielle au génitif, indéfinie), que les formes conjuguées du verbe ذهَب (*ḏahab* – aller) telles que ذهَبُوا(*ḏahabuwā* – ils sont allés), تَذْهَبُ (*taḏhabu* – tu vas), اِذْهَبَا (*íḏhabaā* – allez !, forme impérative au dual), et les formes dérivées telles que ذاهب (*ḏaāhib* – celui qui va, participe actif au masculin, singulier), ذاهبون (*ḏaāhibuwn* – ceux qui vont, participe actif au masculin, pluriel), etc.

Ainsi, nous montrons que ces symboles sont utilisés pour bénéficier de la relation qu'existe entre les différentes formes fléchies d'une même entrée lexicale et éviter l'énumération de celles-ci dans une requête en cas de besoin.

- Les formes associées à une catégorie grammaticale donnée peuvent être toutes reconnues par simple écriture de ce code morpho-syntaxique entre crochets. Par exemple, le symbole <V> représente toutes les formes verbales dans le texte ou le corpus, le symbole <ADV> symbolise le sous-dictionnaire des adverbes, etc. En fait, tous les codes qui apparaissent dans le dictionnaire (voir Tableau 16) peuvent être utilisés dans de telles requêtes. Par ailleurs, le programme de recherche n'attache aucune signification aux codes utilisés : une forme ne sera reconnue comme <ADV> que si elle fait partie des mots ayant été préalablement associées au code morpho-syntaxique ADV dans une entrée du dictionnaire électronique ou à partir d'une annotation attribuée automatiquement par le biais d'une grammaire morphologique ou syntaxique.

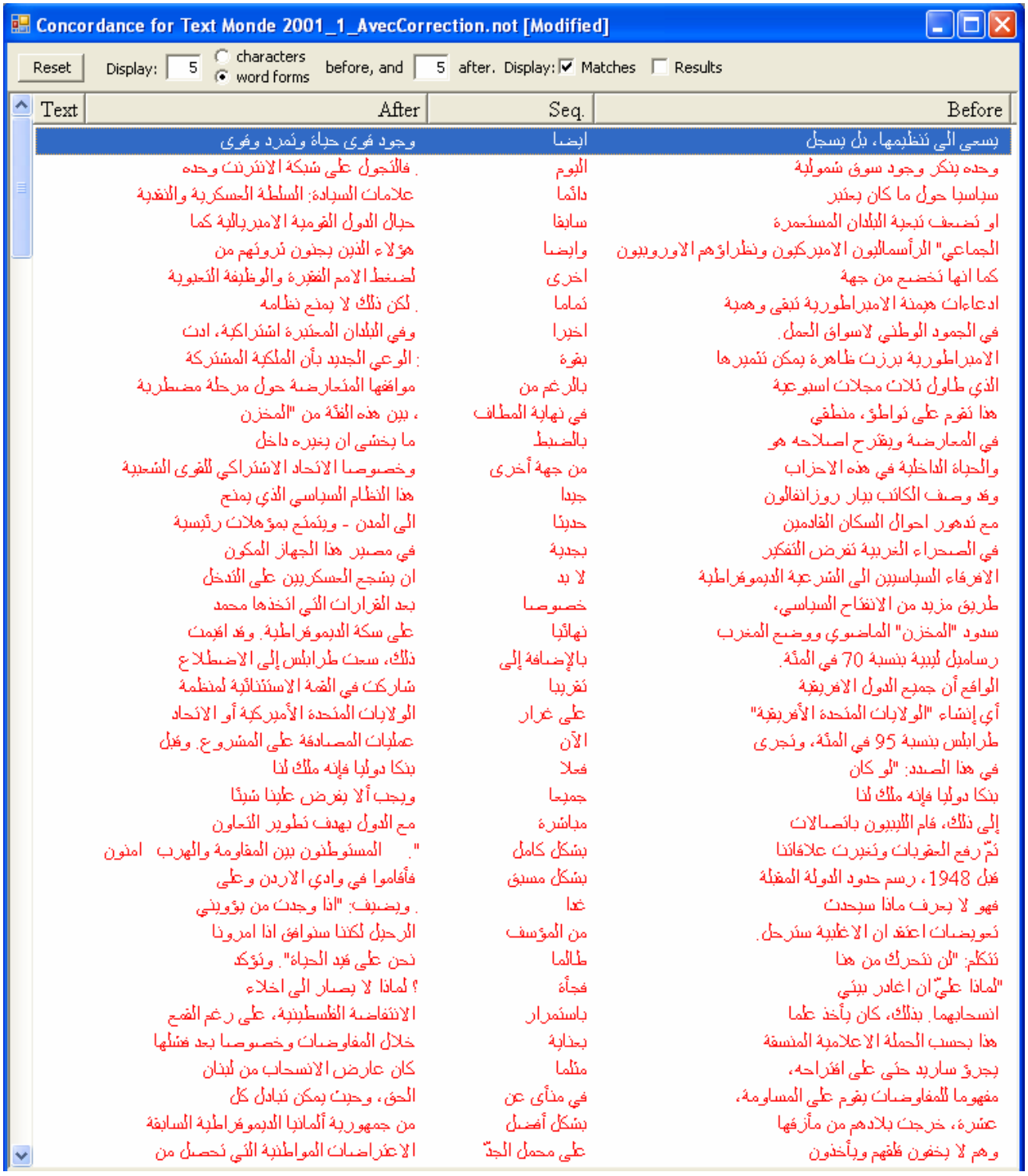


Figure 43 : Table de concordances de la requête : <ADV>

Les mêmes symboles peuvent aussi être utilisés pour représenter des séquences grammaticales. Par exemple, les séquences constituées d'un verbe suivi d'un adverbe peuvent être représentées dans notre système par l'expression rationnelle : <V><ADV>

Figure 44 : Table de concordances de la requête : <V><ADV>

- De plus, nous pouvons combiner les symboles, cités ci-dessus, avec une séquence de codes flexionnels (voir le Tableau 18). Par exemple, nous pouvons écrire :
  - ✓ le symbole <ذَهَب,V> (<*dahaba*,V>) pour représenter uniquement les formes verbales du verbe ذَهَب (*dahaba* – aller) et exclure les formes nominales ;
  - ✓ le symbole <V+P> pour représenter toutes les formes verbales conjuguées au présent de l'indicatif ;
  - ✓ le symbole <ذَهَب,V+1+s> (<*dahaba*,V+1+s>) pour représenter toutes les formes verbales (V) du verbe ذَهَب (*dahaba* – aller) conjuguées à la première personne du singulier (+1+s).

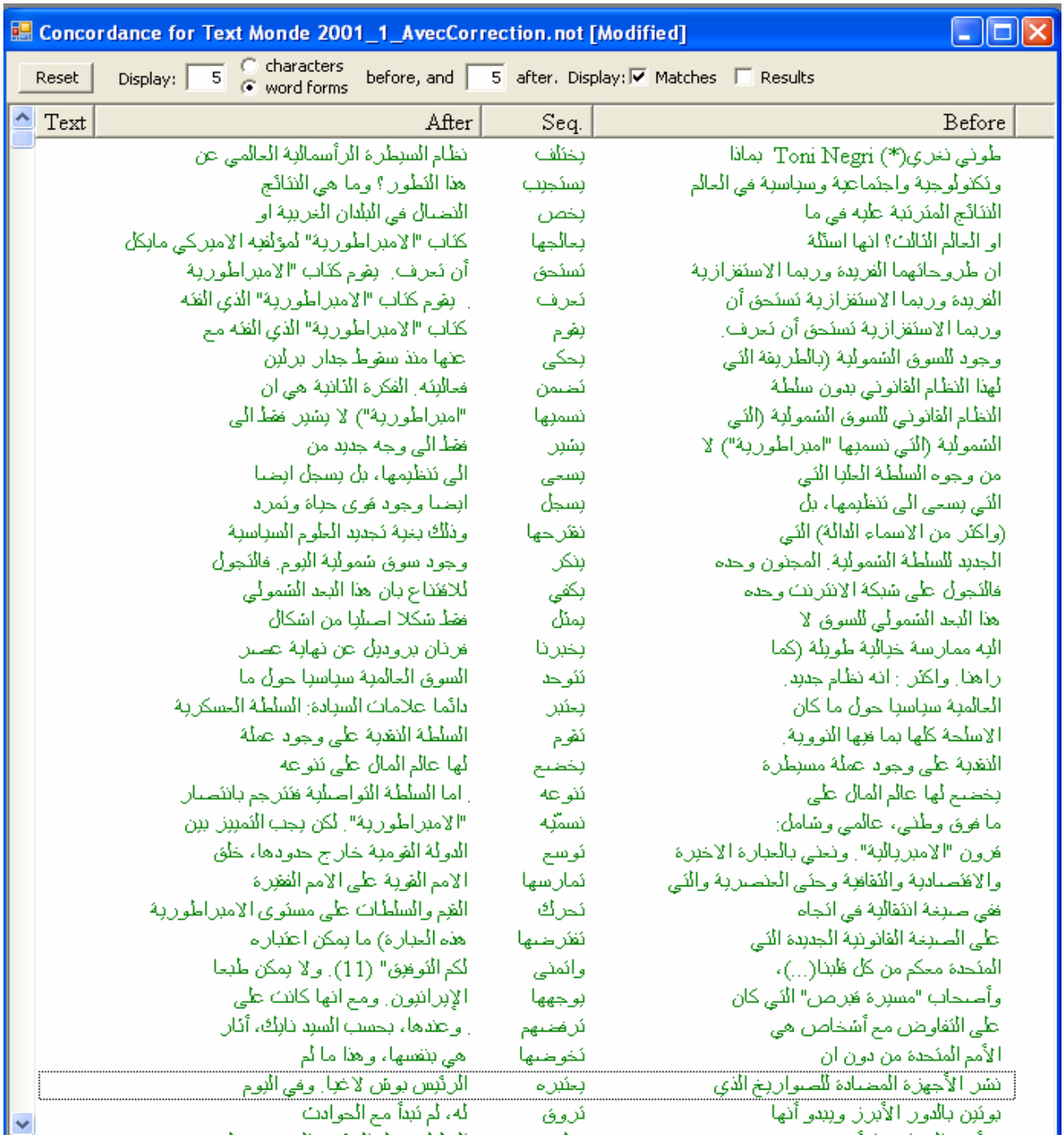


Figure 45 : Table de concordances de la requête :
<V+P>

Les fenêtres de concordances, que nous utilisons pour montrer les résultats des requêtes décrites, offrent plusieurs possibilités de traitement. Parmi lesquels, nous pouvons citer les possibilités de :

- nettoyer les concordances afin d'exclure le bruit des résultats affichés ;
- ajouter des annotations au texte. Cette fonctionnalité permet une annotation interactive du texte ou du corpus en cours de traitement ;
- colorier les séquences retenues dans le texte ;
- exporter les concordances en XML ;
- exporter les indexes des formes ou expressions retenues ;
- construire un rapport statistique des concordances.

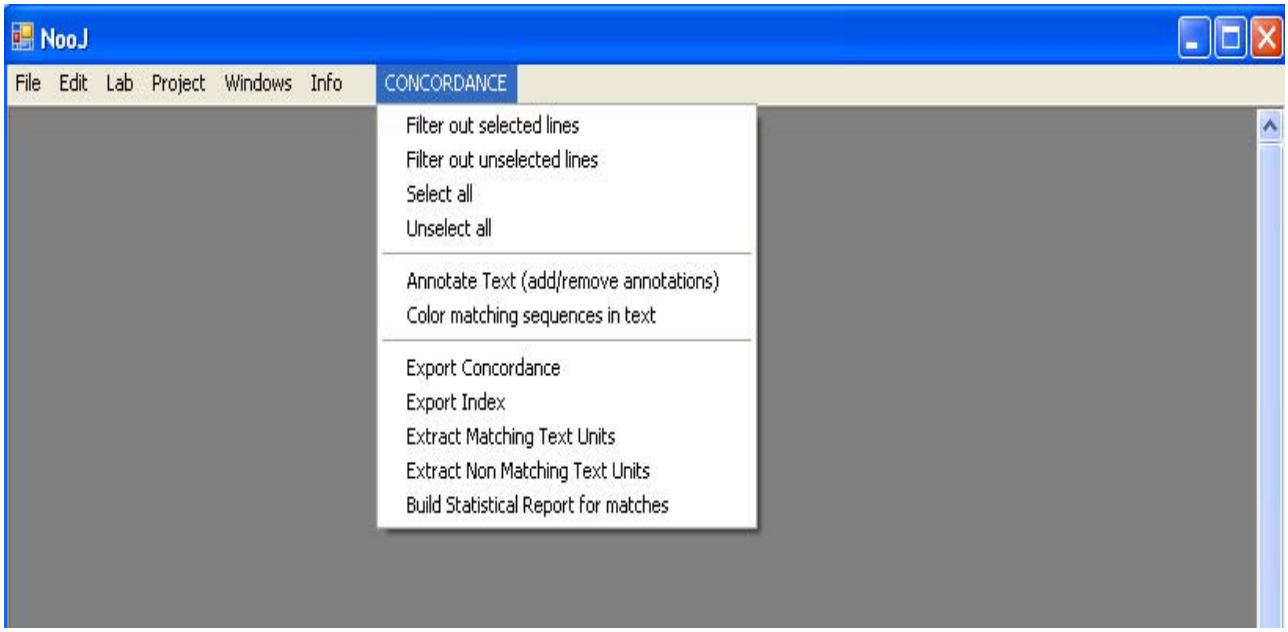


Figure 46 : Menu relatif aux tables de concordances

5.6 Expérimentations et évaluations

5.6.1 Caractéristiques générales du corpus étudié

Le corpus étudié est constitué d'un ensemble d'articles journalistiques publiés par le journal « Le Monde Diplomatique » dans sa version arabe[34]. Les sujets qui y sont abordés sont assez généralistes et traitent les divers thèmes de l'actualité mondiale politique, économique, culturelle, sportive, etc. Le recours à une grande variété de thèmes et de domaines abordés a pour objectif d'avoir une large couverture des mots de la langue. Plusieurs rubriques du journal pendant 6 ans, de l'année 2001 à l'année 2006, ont été recensées. Les journalistes écrivains et les traducteurs sont natifs de divers pays arabes. Ce corpus contient 1 009 articles (voir Figure 47), cumulant un total de 2 006 631 mots graphiques [35] regroupées sous 125 306 formes différentes hors signes de ponctuation, chiffres et mots en alphabet latin.

Nous signalons qu'outre les formes écrites en arabe, ce corpus inclut :

- **des signes de ponctuation :** Dans notre corpus nous avons énuméré 25 séquences de ponctuations différentes, cumulant 60 000 occurrences ;
- **des nombres :** Nous avons dégagé 2 676 séquences de chiffres arabes ou indiens, pour un ensemble de 34 463 occurrences. Pour reconnaître tous ces nombres à l'intérieur des grammaires construites, nous utilisons le symbole <NB> ;
- **des mots non arabes :** ce sont les mots écrits en caractères latins. Dans notre corpus, nous avons dénombré 10517 formes différentes, avec un ensemble total de 38404 occurrences.

[34] Site web: http://www.mondiploar.com .

[35] Nous désignons par *mot graphique* toute séquence de caractères arabes délimitée par deux séparateurs (blanc ou autre marqueur de séparation, tel que la ponctuation) ; si une mot graphique se répète plus qu'une seule fois, il sera compté autant de fois.

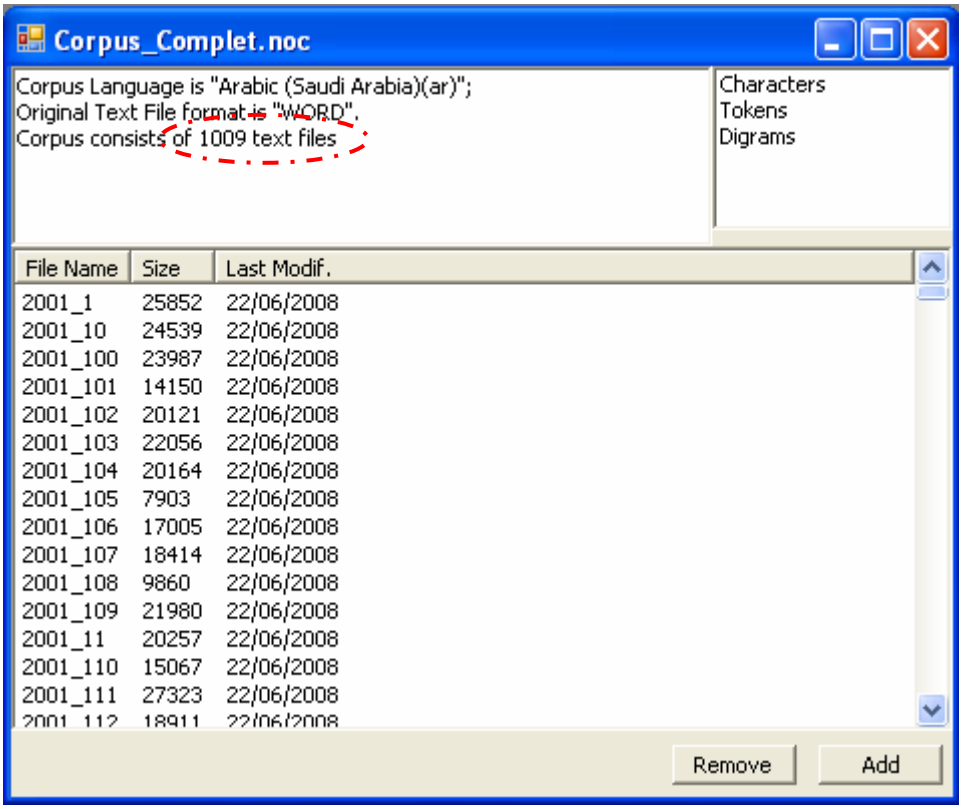


Figure 47 : Construction du corpus de test

Avant toute application de ressources linguistiques, nous fournissons quelques informations générales sur le corpus construit. Outre la liste de caractères et de digrammes[36], nous affichons la liste de toutes les formes accompagnées par leurs fréquences d'apparition (voir Figure 48). Grâce à la technologie à états finis qu'utilise NooJ, cette table de fréquences a été calculée en seulement 10 secondes pour la totalité du corpus. Cette table réalise différentes fonctionnalités :

- Trier les tokens (mots du texte) selon l'ordre alphabétique ;
- Trier suivant la fréquence d'apparition des tokens ;
- Exporter la liste totale ou partielle des tokens ;
- Afficher la table de concordances d'un ou plusieurs tokens au choix.

[36] Un digramme est un assemblage de deux unités graphiques séparées par un blanc.

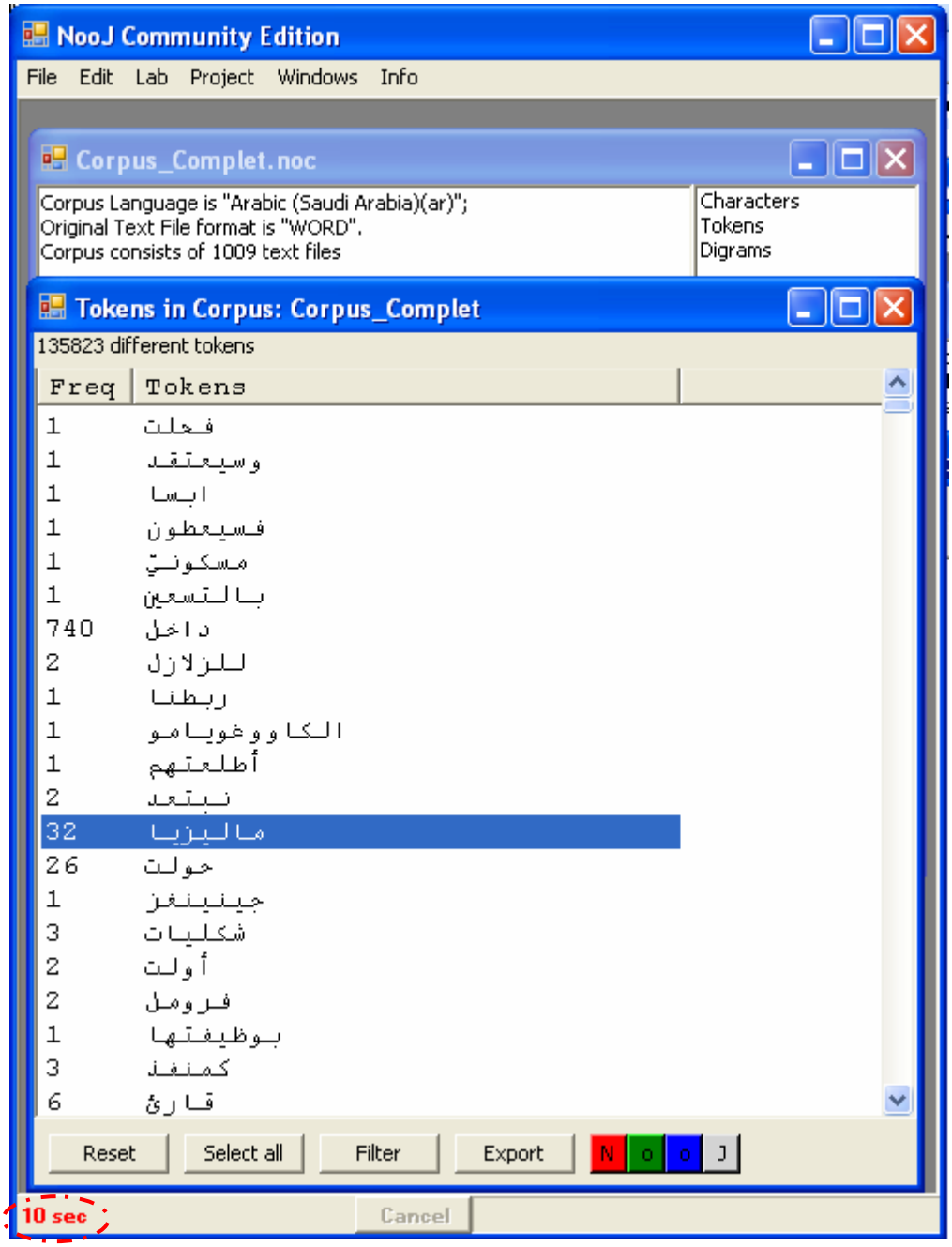


Figure 48 : Liste des formes dans le corpus[37]

En l'occurrence, cette table montre que le token sélectionné "ماليزيا" (*malyziya* - Malaisie) apparaît 32 fois dans notre corpus. Sa sélection permet d'afficher, sous forme d'une table de concordances toutes les occurrences de ce token entourées de leurs contextes droit et gauche selon le modèle KWIC (Key Word In Context), cf. la Figure 49.

[37] La valeur 135 823 affichée au-dessus de cette liste inclut 125 306 formes écrites en arabe et 10 517 formes écrites en caractères latins.

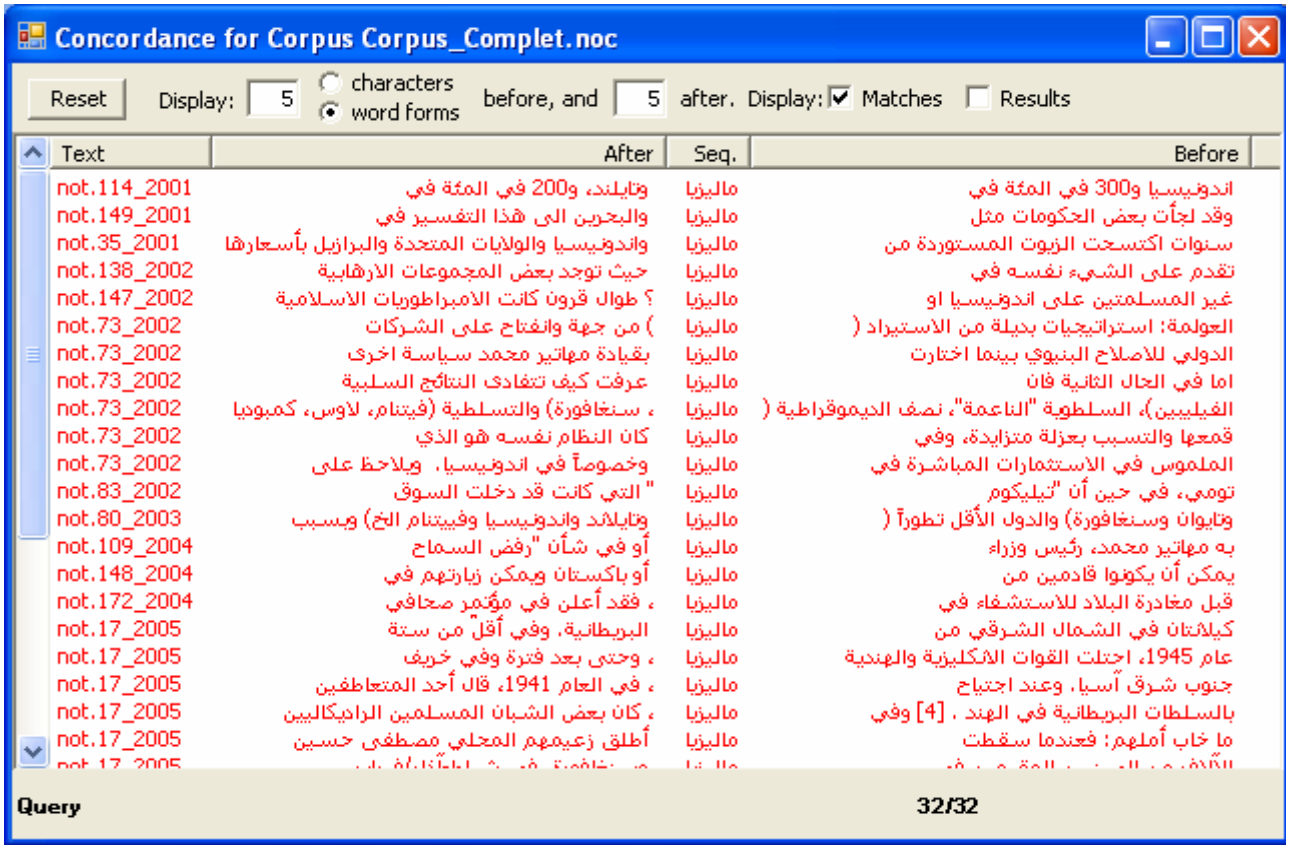


Figure 49 : Table de concordances du token sélectionné "ماليزيا"

La première colonne de cette table exhibe le nom du texte source. Nous notons qu'un double–click sur l'une des lignes de cette table permet d'afficher l'occurrence en surbrillance directement à l'intérieur de son texte source. En l'occurrence, un double-click sur la première ligne provoque l'affichage de la première apparition du token "ماليزيا" (*malyziya* - Malaisie) dans son texte source comme affiché sur la Figure 50.

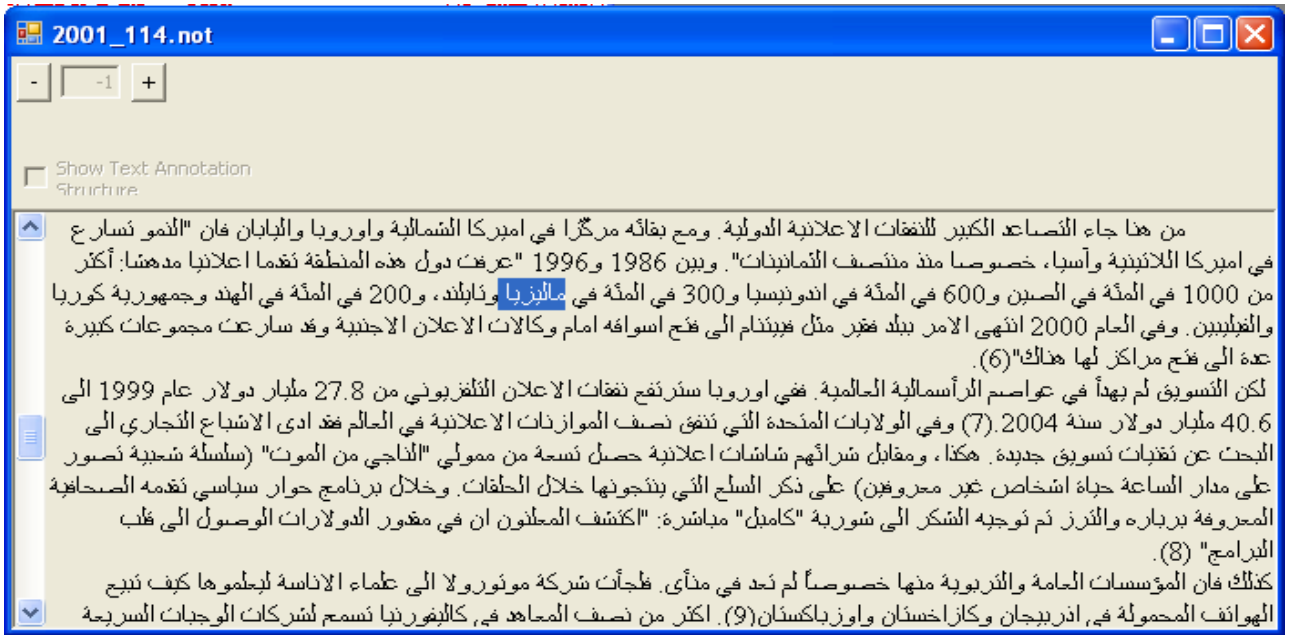


Figure 50 : Affichage du token sélectionné dans son texte source

5.6.2 Evaluation et couverture lexicale

Lors de la construction de nos ressources linguistiques, nous avons eu recours à des travaux de test et de validation à maintes reprises. En effet, avant de passer à une étape ultérieure, nous procédons à la vérification des résultats de l'étape en cours. Dans le tableau ci-dessous, nous montrons les résultats des expérimentations, liées à la couverture lexicale, obtenus à l'issue de chacune des étapes suivantes :

- **1ère étape :** tokenisation et application des dictionnaires électroniques ;
- **2ème étape :** ajout des grammaires de correction des erreurs les plus fréquentes.

	Nombre de formes reconnues		**Nombre de formes non reconnues**	
	Nombre	**%**	**Nombre**	**%**
1ère étape	107 914	86,12 %	17 392	13,88 %
2ème étape	115 946	92,53 %	9 360	7,47 %

Tableau 20 : Evaluation de la couverture lexicale des ressources linguistiques

A l'issue de chaque étape, nous appliquons les ressources linguistiques déjà construites sur l'ensemble des textes de notre corpus. Les résultats affichés nous permettent de réviser, corriger et compléter toutes les ressources mises en jeu.

- La liste des « Annotations » nous permet de réviser aussi bien les entrées de notre dictionnaire et les annotations attribuées aux différents morphèmes (obtenues par le biais des grammaires morphologiques), que celles générées par la collection des grammaires locales construites.
- La liste des « Unknows » est un bon indice de déficience des ressources développées. Ses entrées permettent aussi bien la mise à jour des dictionnaires (par l'ajout de nouvelles entrées lexicales ou par la description de nouvelles règles de déclinaison ou de dérivation), que l'enrichissement des grammaires (par écriture de nouvelles règles de tokenisation ou de nouveaux types de contraintes lexicales).

L'analyse automatique du corpus s'est, en effet, heurtée à plusieurs obstacles relevant pour la plupart des spécificités de l'écrit journalistique arabe contemporain. Comme indiqué sur le tableau d'évaluation, les deux étapes d'application des ressources linguistiques ont aboutit à une couverture lexicale d'environ 93%. La liste des « Unknowns » comporte :

- 7914 entités nommées transcrites : noms propres de personne tels que « شِيرَاك » (šîraāk - Chirac) avec certaines formes dérivées telles que « شِيرَاكِيَّة » (šîraākiyyat - Chiraquisme), des noms de ville tels que « مَرْسِيلِيَا » (marsîliyaā - Marseille) et des organisations telles que « مَايكْرُوسُوفِتْ » (maāyikruwsuwfit –Microsoft) ;
- environ 800 mots étrangers ou emprunts tels que « مِيتَافِيزِيقَا » (*miytaāfiyziyqaā* - métaphysique) ;
- 600 erreurs d'orthographe : ces erreurs typographiques dépassent les heuristiques décrites dans la section 5.4. Parmi ces erreurs, nous signalons l'omission de l'espace entre les mots telle que "عددكبير" (*'adadkabiyr* - ungrandnombre) qui normalement doit s'écrire "عدد كبير" (*'adad kabiyr* – un grand nombre).
- Quelques **lettres isolées :**ces lettres interviennent dans deux cas différents :
  - ✓ Dans des **abréviations** : soit en tant que sigle tel que آ.ف.ب (*â.f.b* – A.F.P., Agence France Presse), ou en tant que signes représentant un mot entier tels que : م (*m*) au lieu du mot ميلادي (*miylaādiy* – du calendrier grégorien) ou encore ه (*h*) au lieu de هجري (*hiğriy* – de l'Hégire), trouvés lors de l'écriture des expressions de date, etc. ;

- ✓ Comme **proclitiques prépositionnels** comme كَ (*ka* - comme), لـ (li – *pour*), بِ (*bi* – à) ou **des proclitiques de coordination** tels que فَ (*fa* – alors), وَ (*wa* - et). Ils apparaissent, généralement, accompagnés d'un mot étranger ou d'un nom propre tel que "لـ "ألكاتال (*li-alkaātaāl* – pour Alcatel), etc.

Cette expérimentation prouve que 85% de formes non reconnues sont des noms propres (noms de personne, d'organisations ou de lieux), contre seulement 8,4% de mots étrangers, 6,4% d'erreurs typographiques et 0,2% de lettres isolées. La plupart de ces formes sont des constituants d'entités nommées (ENs). Celles-ci encapsulent des informations importantes utiles pour l'interprétation sémantique des textes, mais elles ne figurent que rarement dans des dictionnaires. Citons cependant le dictionnaire électronique relationnel multilingue de noms propres français « *Prolexbase* »[38] qui contient 54 774 noms propres, 730 alias et 20 614 formes dérivées [Maurel, 2008].

Pour l'arabe, nous n'avons pas accès à des bases de données lexicales énumérant un nombre significatif de noms propres, et gérant toutes leurs variantes et les divers types d'ambiguïté qui en résultent. Il faut donc construire un système de reconnaissance automatique d'entités nommées sur des bases non-lexicales : nous utiliserons des techniques de reconnaissance morphologiques et syntaxiques. Ce système est le sujet de la section suivante. Il se base sur des règles représentées sous forme de grammaires locales construites à l'aide du module syntaxique dans NooJ.

[38] Le dictionnaire « Prolexbase » est construit dans le cadre du projet Prolex. Ce projet s'est d'abord focalisé sur la création d'une base de données de toponymes et de leurs dérivés puis a été étendu à une base relationnelle multilingue de noms propres [Tran, 2006].

Chapitre 6

Reconnaissance automatique des entités nommées

6.1 Introduction à l'analyse syntactico-sémantique automatique

La reconnaissance automatique des entités nommées rentre dans le cadre d'une analyse syntactico-sémantique, une étape qui suit l'analyse morpho-lexicale lors du traitement automatique d'un texte ou d'un corpus. En réalité, cette phase d'analyse consiste à exhiber certains concepts grammaticaux ou structures syntaxiques, à contrôler leur validité et à attester leur appartenance à des classes grammaticales particulières telles que « les mots composés », « les expressions figées », « les entités nommées », etc. Souvent, les méthodes employées pour réaliser un tel traitement se basent sur des grammaires non-contextuelles[39]. Ces grammaires sont utilisées au sein de notre plateforme de développement linguistique NooJ pour la tâche en cours : elles sont baptisées grammaires locales.

Elles servent à localiser des phénomènes locaux de manière très précise dans les textes, comme les dates [Maurel, 1990], les déterminants numéraux [Chrobot, 2000 ; Silberztein, 2003], les entités nommées [Friburger, 2002], etc. Ces grammaires sont des graphes lexicalisés [Gross, 1997 ; Silberztein, 1993] qui font appel à des dictionnaires de mots simples et composés. Elles sont équivalentes à des réseaux récursifs de transitions (RTN) voir même des réseaux de transitions augmentés (ATNs) [Woods, 1970]. En pratique, les grammaires locales sont des graphes qui peuvent appeler des sous-graphes indépendants. Parmi les avantages d'une telle structure, nous citons l'efficacité de son application directe aux textes, la reconnaissance de concepts linguistiques complexes ainsi que l'analyse transformationnelle et la production d'annotations.

Plus concrètement, la description de certains phénomènes linguistiques nécessite typiquement la construction de plusieurs dizaines de graphes. A l'intérieur de ces graphes, de nombreux phénomènes élémentaires plus ou moins complexes interviennent dans différents contextes. En l'occurrence, la liste des noms de jour "الإثنين", … et "الأحد" (*al-íṯnayn, ..., al-áḥad* – lundi … dimanche) peut être utilisée dans un graphe pour décrire aussi bien des compléments de date[40] que des compléments de durée[41]. Ainsi, l'avantage des réseaux de transitions augmentés (ATNs) apparaît clairement. En effet, certaines séquences peuvent se retrouver plusieurs fois dans la séquence à reconnaître ; nous allons donc leur associer un seul graphe, qui sera appelé plusieurs fois dans la séquence globale.

Sur le plan pratique, dans NooJ, les grammaires locales sont représentées par des graphes. Chaque graphe est représenté sous forme d'un ensemble de nœuds connectés et incluant un nœud initial et un nœud terminal. A l'intérieur de ces graphes :

- *Le nœud initial* : il est représenté par une flèche droite ;
- *Le nœud terminal* : il est représenté par une croix dans un rond ;
- *Les nœuds intermédiaires* : ce sont des boîtes qui contiennent les étiquettes en entrée du transducteur. Ils peuvent contenir :
  - ✓ *Des symboles grammaticaux* : ces symboles représentent, dans la majorité des cas, des mots au sens linguistique et sont donc très variés. Ces symboles sont utilisés afin de limiter le nombre de transitions dans les graphes. Parmi ces symboles, nous citons :
    - o <N> : désigne n'importe quelle unité lexicale ayant été associée à la catégorie grammaticale N (noms simples ou composés) ;
    - o *<V+3+m+s>* : désigne n'importe quel verbe conjugué à la troisième personne, masculin, singulier ;

[39] Les grammaires non-contextuelles représentent les grammaires de type 3 selon la hiérarchie de Chomsky.

[40] Par exemple, nous utilisons un nom de jour pour décrire la date : 2006 نُوفَمْبَر 27 يَوْم الِثْنَيْن (*yawm al-iṯnayn 27 nuwfambar 2006* – le lundi 27 novembre 2006).

[41] Par exemple, nous utilisons deux noms de jour pour désigner une période : من الإثنين إلى الأحد (*min al-íṯnayn ilaą al-áhad* – de lundi à dimanche).

- <كَتَبَ> : désigne toutes les formes fléchies du verbe "كَتَبَ" (*kataba* – écrire). Ce symbole remplace la disjonction d'environ 170 formes fléchies rattachées à ce lemme : 122 formes verbales, 48 participes actifs et passifs ;
- <NB> : symbolise n'importe quel nombre ou séquence de chiffres.

✓ *Des appels à des sous-graphes* : ces appels sont assurés par le biais du symbole « : » (n'apparaissant pas sur les graphes) qui précède le nom du sous-graphe à appeler. Par exemple, les nœuds qui apparaissent coloriés dans la Figure 55 (Centaines, Unités et Dizaines) sont des appels aux sous-graphes correspondants. Par exemple, la boîte « :Centaines » du graphe des Milliers de la Figure 59 fait appel au graphe décrivant les centaines de la Figure 58 ; le même graphe est utilisé à d'autres reprises pour la reconnaissance des Millions et des Milliards. Ainsi, la représentation de la grammaire de reconnaissance des déterminants numériques, détaillée dans la section 6.5.1, devient beaucoup plus compacte.

- *Des étiquettes sous les nœuds* : elles représentent les sorties produites. Par exemple, dans la Figure 52, nous avons ajouté l'étiquette <ADV+Temps> en-dessous du nœud initial pour indiquer que toute expression reconnue par cette grammaire sera annotée en tant qu'adverbe de temps ;

Ainsi, outre la possibilité de description des règles de flexion et de dérivation (voir section 4.4) et d'analyse morphologique (voir section 5.2), les graphes de NooJ peuvent être utilisés dans d'autres applications liées au traitement automatique des langues [Constant, 2003]. Parmi ces applications, nous citons :

- **Reconnaissance et étiquetage automatiques:**

Les grammaires locales lexicalisées permettent de reconnaître des séquences composées et d'étiqueter ces dernières de manière très satisfaisante. Par exemple, la grammaire ci-dessous permet de représenter une classe de mots composés sémantiquement proches de "عِيد" (*ʿiyd* – fête) [Silberztein, 2000]. Elle reconnaît les expressions qui y sont liées telles que "عيد الأضحى" (*ʿiyd al-aḍḥaq* – Aid El Ad'ha), "عيد الفطر" (*ʿiyd al-fitr* – Aid El Fitr), "عيد الأمّهات" (*ʿiyd al-ommahaā*t – Fête des Mères), etc.

Figure 51 : Grammaire locale de reconnaissance :
Les noms de fêtes

Outre la reconnaissance, il est possible que ces grammaires produisent des informations en sortie [Silberztein, 1999a]. Généralement, ces sorties jouent le rôle d'étiquette syntatico-sémantique pour la séquence identifiée. Par exemple, le graphe de la Figure 52, décrivant des adverbes de temps tels que "في وضح النهار" (*fiy waḍḥi an-nahaār* – en pleine journée) ou "بشكل شبه يومي" (*bišaklin šibh yawmiyy* – quasi-quotidiennement), peut servir pour étiqueter ce type d'adverbes. Sur le plan de la formalisation, cet étiquetage est effectué par le biais des informations de sortie écrites en gras sous les boites du graphe. En l'occurrence, les étiquettes écrites en-dessous des nœuds du graphe pour annoter chaque séquence reconnue par <ADV+Temps>. Toutefois, lorsqu'aucune étiquette n'est représentée en-dessous des nœuds du graphe, les sorties seront vides et les grammaires ne joueront plus le rôle d'étiqueteur.

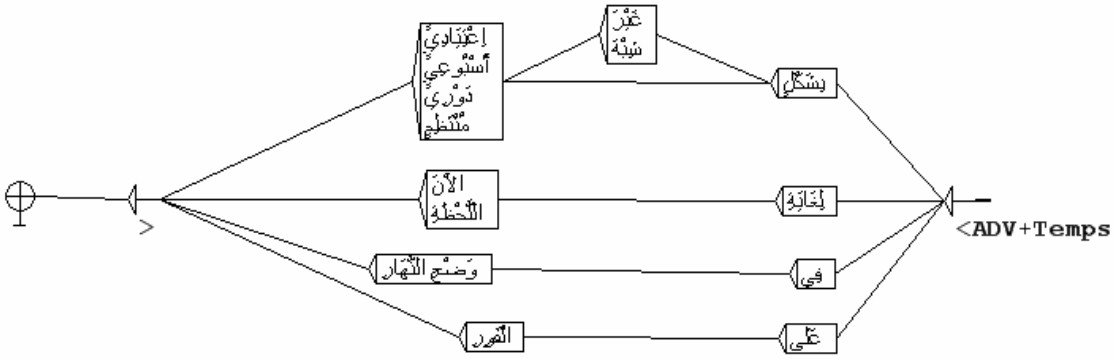


Figure 52 : Grammaire locale de reconnaissance :
Etiquetage de locutions adverbiales de temps

Par exemple, après l'application de cette grammaire à la phrase "وقعت السرقة في وضح النهار", celle-ci peut être étiquetée comme suit :

<\ADV> في وضح النهار<ADV+Temps> وقعت السرقة
waqa'at al-sariqat <ADV+Temps> fiy waḍḥi an-nahaār </ADV>
Le vol s'est déroulé <ADV+Temps> en pleine journée </ADV>

Cette fonctionnalité sera détaillée dans la section suivante lors de la description de la reconnaissance et l'annotation des déterminants numériques en arabe.

- **Extraction de données :**

  L'extraction d'information est l'un des domaines les plus en vogue ces dernières années, il s'agit notamment de l'extraction de noms propres. De nombreuses études ont été menées sur le français pour extraire automatiquement des noms de personnalités en leur associant une fonction politique ou professionnelle à l'aide de grammaires sous forme de graphes [Senellart, 1998] ou bien extraire des noms propres de personne à l'aide de cascades de transducteurs [Friburger et Maurel, 2004] ou encore extraire les noms de gènes dans les corpus en génomique [Poibeau, 2001]. Tout au long de ce chapitre, nous détaillerons l'utilisation des grammaires locales pour l'extraction d'entités nommées.

- **Ambiguïté :**

  La levée d'ambiguïté est fondamentale pour le traitement automatique de textes. De nombreuses études pointues ont montré l'intérêt de l'utilisation des cascades de transducteurs pour la désambiguïsation [Silberztein, 2005a]. Dans la section 5.3.3, nous décrivons quelques grammaires locales de levée d'ambiguïté.

En outre, les grammaires locales peuvent être utilisées pour le filtrage d'information, notamment les travaux sur la distribution des dépêches AFP[42] [Balvet, 2000]. elles peuvent aussi servir comme outil d'aide à la traduction automatique [Mesfar, 2006].

6.2 Typologie des entités nommées

Les noms propres sont très fréquents dans les textes électroniques, notamment les articles journalistiques (10% en anglais selon [Coates-Stephens, 1993]). Mais, malgré la fréquence de leurs apparitions et l'importance des informations qu'ils encapsulent notamment pour l'interprétation sémantique des textes, les noms propres restent insuffisamment représentés dans les ressources

[42] AFP : Agence France Presse (site web : http://www.afp.com/)

lexicales électroniques et leur extraction automatique ne représente qu'un domaine relativement jeune [Friburger, 2002], [Piton et al., 1996].

En effet, c'est à partir de 1995 que les organisateurs de la conférence MUC[43] introduisaient la notion d'extraction de noms propres et créaient la tâche de reconnaissance et de catégorisation automatique de ce qu'ils appellent *entités nommées*. Dès le départ, cette appellation a été élargie pour inclure d'autres types d'expressions. Selon MUC, nous distinguons, au moins, trois types d'entités à reconnaître et à classer par catégorie [Poibeau, 2005]:

- **ENAMEX :** cette classe regroupe les noms propres. Nous distinguons au moins trois sous-catégories :
  - ✓ **Personnes :** noms de personne. Par exemple, "جُون كنيدي" (*ğuwn kinydy* – John Kennedy) ;
  - ✓ **Organisations :** sociétés, banques, associations, universités… Par exemple, إيرْبُوس (*iyrbuws* – Airbus), يُونِيسْكُو (*yuwniyskuw* – Unesco), etc. ;
  - ✓ **Localisations :** toponymes tels que les noms de pays, villes, états, mers, océans, montagnes, fleuves… Par exemple, فرَنْسَا (*firansaā* - France), بَاريس (*baāriys* - Paris), البحر الأبيض المتوسّط (*el batr eláabya*ḍ *elmutawassiṭ* – La mer méditerranée)…
- **NUMEX :** expressions numériques de pourcentage, taille, expressions monétaires, etc.
- **TIMEX :** expressions temporelles de date ou de durée.

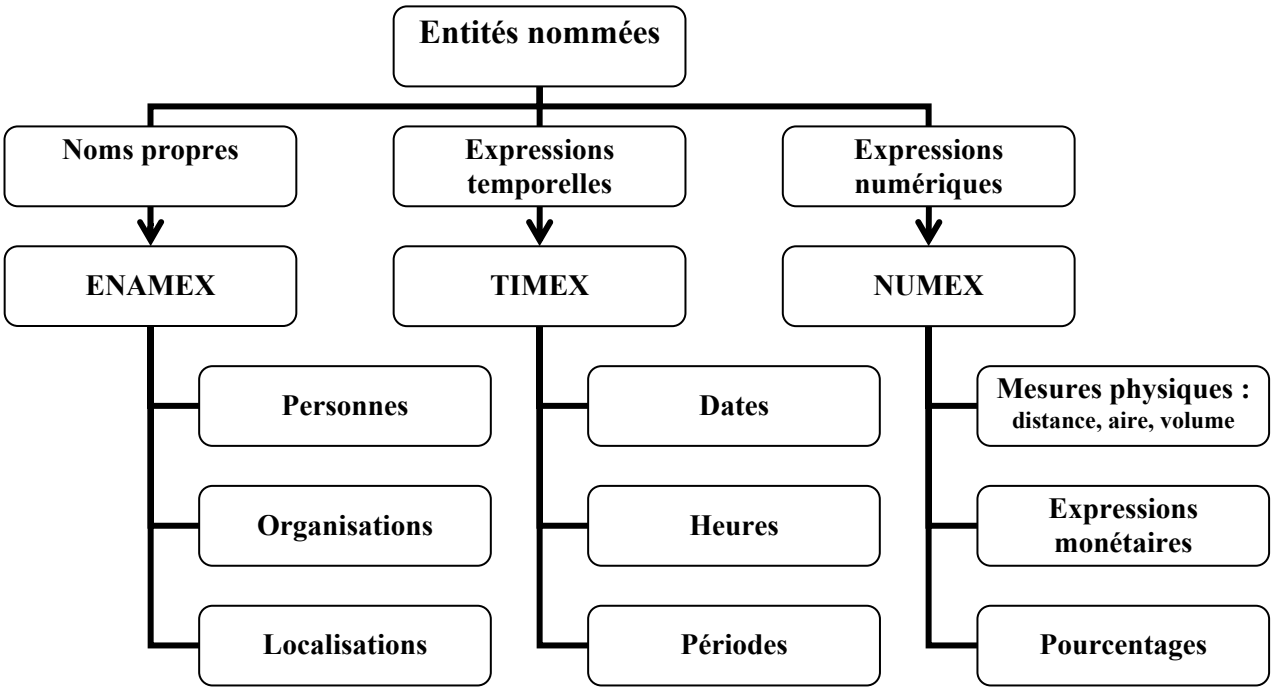


Figure 53 : Typologie des entités nommées

Dès le départ, la mise en œuvre de la solution lexicographique, qui consiste en une simple consultation de dictionnaires électroniques dénombrant toutes les entités nommées, s'est avérée impossible. Ceci est dû particulièrement aux problèmes de la graphie multiple et l'absence de normes d'écriture ou de transcription des noms propres notamment ceux d'origine étrangère. En effet, il est impossible d'énumérer tous les noms propres dans des listes, de rassembler et maintenir ces listes, de traiter toutes les variantes orthographiques et finalement de résoudre l'ambiguïté qui en résulte.

[43] http://en.wikipedia.org/wiki/Message_Understanding_Conference

Par ailleurs, il existe trois principaux types de systèmes pour extraire les noms propres. Un aperçu de ces trois systèmes a été donné dans [Poibeau, 1999] et [Friburger, 2002] :

- **Les systèmes à base de règles :** Dans la littérature, la majorité des systèmes utilisent cette approche. Les systèmes typiques à base de règles utilisent les preuves internes et externes, ainsi que des dictionnaires de mots déclencheurs pour l'aide au repérage. Les règles sont écrites à la main.
- **Les systèmes à apprentissage :** Ces systèmes construisent leurs connaissances automatiquement grâce à un apprentissage sur un corpus d'entraînement [Collins, Singer, 1999].
- **Les systèmes hybrides :** Ces systèmes se basent sur l'utilisation aussi bien de règles écrites à la main que d'autres qui sont construites automatiquement à l'aide d'informations syntaxiques et d'informations sur le discours tirées de données d'entraînement grâce à des algorithmes d'apprentissage et d'arbres de décision.

L'adéquation des systèmes à base de règles a été reconnue lors de la dernière conférence MUC 7. C'est cette même technique que nous préconisons pour le développement d'une composante d'extraction d'entités nommées. Cette composante est basée sur des règles écrites à la main et représentées sous forme de grammaires locales construites à l'aide du module syntaxique de NooJ. Ces règles ont été fondées sur des preuves internes et externes pour l'identification et la catégorisation des entités nommées [MacDonald, 1996] où :

- **Les preuves internes** sont fournies par les constituants de l'entité nommée. Elles peuvent être contenues dans des listes de mots déclencheurs ou de noms propres appelées gazetteers.
- **Les preuves externes** sont fournies par le contexte dans lequel une entité nommée apparaît. Elles se basent sur les relations syntaxiques au sein d'une phrase pour attribuer la catégorie d'une telle entité. Cette catégorisation utilise les informations morpho-syntaxiques fournies par l'étape précédente d'analyse morphologique.

L'usage de ces preuves est indispensable à cause de l'absence d'indices évidents permettant de détecter la présence d'un nom propre, comme la présence de majuscules à la tête de tels noms dans les langues romanes. Ceci nous impose une compréhension assez approfondie de la nature morphologique de chaque forme du texte, notamment sa catégorie grammaticale et ses informations distributionnelles (par exemple : +Humain, +Pays, +Monnaie, …).

6.3 Problèmes de la reconnaissance des entités nommées

Nous exposons dans cette partie les problèmes de la reconnaissance des entités nommées en arabe. Nous décrivons notamment la différence entre leurs divers types, la répercussion de l'absence de voyellation et le problème de délimitation de celles-ci.

6.3.1 TIMEX et NUMEX vs. ENAMEX

TIMEX & NUMEX	ENAMEX
✓ Indices structuraux ou contextuels suffisants ; ✓ Toutes les expressions temporelles et numériques sont identifiables à l'aide d'une liste de marqueurs lexicaux (noms de jours, noms de mois, monnaies, unités de mesure, etc.) ; ✓ Expressions basées sur des listes statiques.	✓ Absence de majuscules (indice naïf) ; ✓ Très peu d'indices structuraux ou contextuels ; ✓ Listes de mots déclencheurs (titres, places,…) ; ✓ Abondance des mots inconnus par analyse morpho-lexicale → absence d'informations linguistiques nécessaires pour la reconnaissance → indices insuffisants pour le typage des ENs ; ✓ Expressions basées sur des informations morphologiques générées dynamiquement par l'analyse lexicale.

En se basant sur ce tableau comparatif, nous signalons qu'en plus de l'absence d'indices évidents pour reconnaître la présence d'un nom propre (par exemple les lettres initiales en majuscules dans le cas des langues romanes), d'autres problèmes spécifiques à la reconnaissance des entités nommées en arabe surgissent aussi bien lors de l'identification et de la délimitation que lors du typage de celles-ci.

6.3.2 L'absence de voyelles

Le problème de la non vocalisation est lié, directement, au degré élevé d'ambiguïté qui découle du manque de voyelles courtes au sein des textes courants, voir section 2.5.3. Cette absence de signes diacritiques peut affecter les systèmes de reconnaissance des entités nommées. Ceci est principalement dû à l'ambiguïté sémantique qui découle de l'ensemble des vocalisations potentielles pouvant être attribuées à toute forme partiellement voyellée ou non voyellée. En effet, les vocalisations acceptées pour une forme quelconque d'un texte peuvent mener à différents sens pouvant désigner des mots déclencheurs qui introduisent différents types d'entités nommées. En l'occurrence, la forme non voyellée « مؤسسة » (*muẃass'sat*) peut accepter, entre autres, les deux vocalisations suivantes:

- « مُؤَسَّسَة » (*muẃassasat* - la compagnie) : mot déclencheur d'un nom d'organisation ;
- « مُؤَسِّسَة » (*muẃassisat* - la fondatrice) : mot déclencheur d'un nom de personne.

Cet exemple permet d'illustrer les répercussions de l'absence des voyelles dans les mots des textes courants sur l'étape de typage des entités nommées. En effet, il suffit d'avoir, dans un texte, ce même mot non voyellé suivi d'un nom propre, restant non reconnu après analyse morpho-lexicale, pour remarquer que l'entité en question est ambiguë : elle peut être analysée en tant que nom de société ou bien celui d'une personne.

6.3.3 Problèmes de délimitation et polysémie

Les problèmes de la délimitation des entités nommées sont, essentiellement, liés à l'absence d'informations qui découle de l'abondance des mots inconnus, par l'analyseur morpho-lexical, dans les entités nommées. Ces problèmes de délimitation se sont développés avec l'utilisation d'antonomases où des noms propres sont substitués par des séquences de mots ou réciproquement ainsi que l'emploi d'un certain nombre d'homonymes [44]. En effet, la présence de formes polysémiques dans une entité nommée augmente les difficultés de délimitation de celle-ci. En l'occurrence, nous citons certains prénoms qui pouvent jouer le rôle d'un nom ou d'un verbe ou d'un adjectif tels que :

- « أَشْرَف » (*áašrafa*) : cette forme peut désigner un prénom masculin, une forme verbale fléchie (il a supervisé) ainsi qu'un adjectif superlatif (le plus honnête) ;
- « أَحْمَد » (*áaḥmadu*) : cette forme peut désigner un prénom ainsi qu'une forme verbale fléchie (je loue).

Par ailleurs, nous signalons qu'il existe des cas d'ambiguïté où l'entité nommée peut être confondue avec un nom composé ou un fragment d'une phrase verbale. Par exemple, pour la séquence non voyellée "حافظ الأسد" (*ḥaāfiẓ al-ásad*), nous pouvons distinguer les trois analyses suivantes :

- *Le nom d'une personne politique* : "الرئيس السوريّ حافظ الأسد" (*al-raýiys al-suwriy ḥaāfiẓ al-ásad* – le Président Syrien Hafedh Al-Asad) ;
- *Un nom composé* : "نظّف حافظ الأسد القفص" (*naẓẓafa ḥaāfiẓ al-ásad al-qafaṣa* – Le gardien du lion a nettoyé la cage) ;
- *Un fragment d'une phrase verbale* : "حافظ الأسد على هيبته" (*ḥaāfaẓa al-ásad 'laą haybatihi* – Le lion a préservé sa dignité)

[44] Un homonyme est un mot qui a les mêmes formes orale et écrite qu'un autre mot, mais avec une signification différente.

Ainsi, nous essayons de baser la délimitation des entités nommées sur les informations fournies par notre analyseur morphologique pour décider l'appartenance d'une forme du texte à une entité nommée lors de la détermination des contextes [Maloney, 1998]. Bien que pour les grammairiens arabes, tout élément de la langue, simple ou complexe, peut être un nom propre, y compris les onomatopées, nous avons dénombré une liste de formes dont l'appartenance à un nom propre serait peut probable. Nous citons :

- ✓ Les mots invariables tels que les prépositions, les adverbes, etc. ;
- ✓ Les formes verbales fléchies telle que يَكْتُبُ (*yaktubu* - il écrit) à l'exception de la liste recensée des prénoms arabes comme يَزِيدُ (*yaziydu* - il ajoute) ;
- ✓ Certains éléments lexicaux tels que des verbes d'expression tels que قَالَ (*qaāla* – dire) ou كَلَّمَ (*kallama* – parler avec qqu'un) et les verbes de connaissance tels que عَرِفَ (*'arifa* - savoir).
- ✓ Les formes nominales suffixées comme كِتَابُهُ (*kitaābuhu* - son livre) ;
- ✓ Les formes verbales suffixées par un sujet ou un objet telles que يُكَاتِبُه (*yukaātibuhu* - il lui écrit).

6.4 Approche de reconnaissance des entités nommées

Afin de remédier à tous ces problèmes, nous construisons un système de reconnaissance d'entités nommées pour l'arabe. Nous procèdons en deux étapes :

- Tout d'abord, nous rassemblons le maximum d'informations pour toutes les formes du texte reconnues par analyse morphologique ou par consultation des dictionnaires électroniques ;
- Par la suite, ces informations seront employées dans des grammaires locales, syntactico-sémantiques, pour localiser les séquences pertinentes.

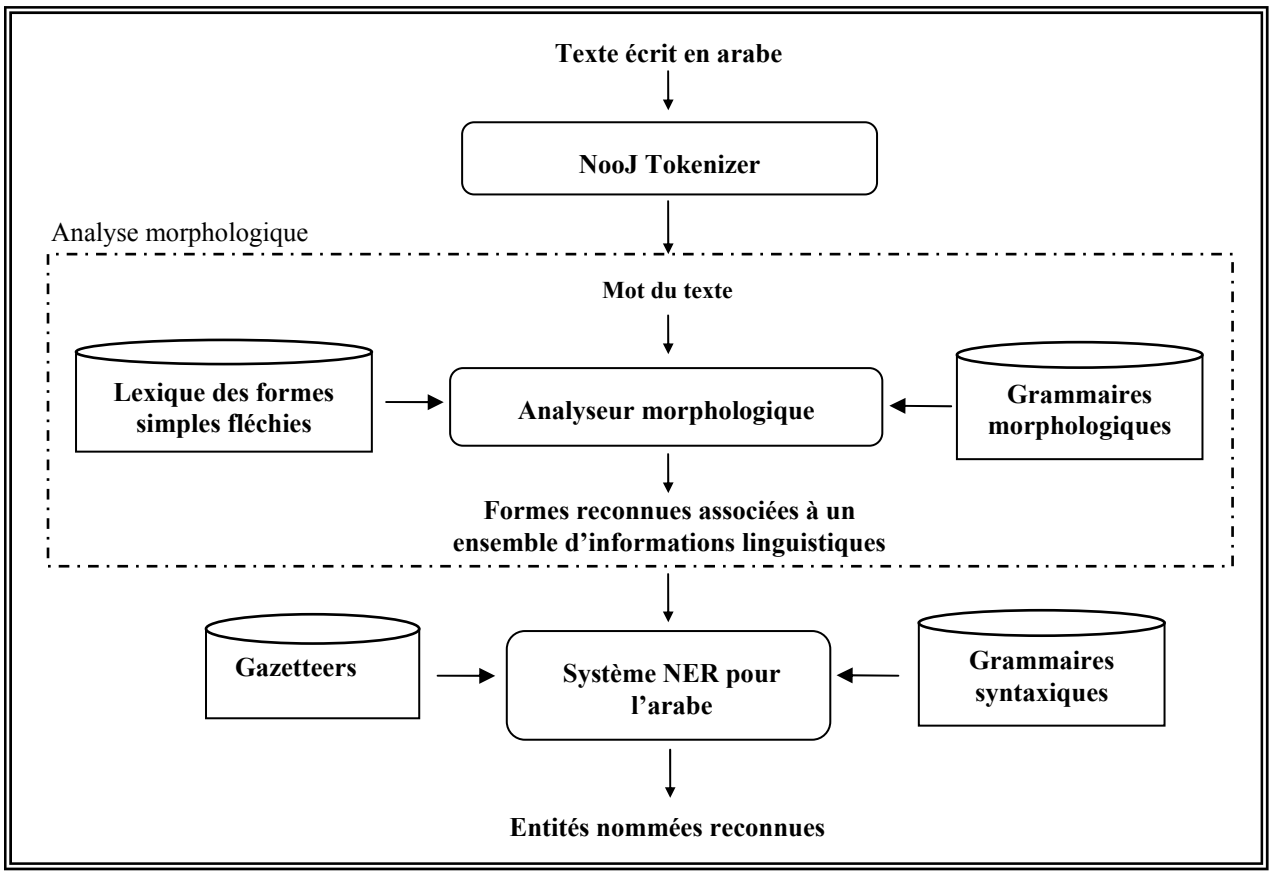


Figure 54 : Architecture générale du système de reconnaissance des entités nommées

Compte tenu de la structure agglutinante que présente la majorité des mots en arabe, notre analyseur morphologique, décrit à la section 5.2, permet de séparer et d'identifier les morphèmes des formes en entrée et leur associer l'ensemble d'informations nécessaires au traitement en cours. Ces formes sont décomposées pour reconnaître les affixes (conjonctions, prépositions, pronoms personnels, etc.) qui s'y rattachent. Ces possibilités morphologiques dans NooJ permettent de faciliter l'identification des mots déclencheurs, des noms de personnes ou de localités même lorsqu'ils sont agglutinés. En effet, chacune de ces formes est associée, par analyse morphologique, à un ensemble d'informations linguistiques utiles pour l'étape suivante: lemme, étiquette grammaticale, genre et nombre, information syntaxique (Ex : +Transitive), information distributionnelle (Ex : +Humain), etc. En conséquence, au lieu d'énumérer toutes les formes fléchies (singulier, duel, pluriel, masculin, féminin) des noms de professions considérés comme marqueurs lexicaux de noms de personnes (par exemple, "طبيب" (*tabiyb*, docteur)), nous utiliserons la syntaxe des expressions rationnelles de NooJ où le symbole grammatical <طبيب > (<*tabiyb*>, <docteur>) désigne toutes les formes potentielles fléchies vocalisées, partiellement vocalisées, ainsi que celle non vocalisées rattachées à ce lemme.

Ensuite, les informations fournies par analyse morphologique sont directement utilisées par notre système de reconnaissance d'entités nommées. Outre ses informations morpho-syntaxiques récolées, ce système se base sur l'utilisation de deux types de ressources linguistiques :

- **Gazetteers:** Ce sont les listes de marqueurs lexicaux antérieurement reconnus comme membres potentiels d'entités nommées et proprement classifiés. Parmi lesquels, nous distinguons :
  - ✓ Noms de personnes : 12 400 prénoms arabes et prénoms étrangers transcrits ;
  - ✓ Noms de lieux: 5 038 entrées: pays, villes, états, mers, océans, montagnes, fleuves, etc. ;
  - ✓ Noms d'organisations: 843 noms d'organisations : associations internationales, universités, télévisions, etc. ;
  - ✓ Expressions monétaires: 175 noms de monnaies et leurs subdivisions ;
  - ✓ Expressions temporelles: la liste des noms de jours, plusieurs listes de noms de mois[45], etc.
- **Grammaires locales :** Elles sont représentées, rappelons-le, sous forme de réseaux de transitions augmentés (Augmented Transition Network – ATN). Elles permettent de représenter des séquences de mots. Ces séquences sont décrites par le biais de règles manuellement écrites et de produire en résultat certaines informations linguistiques telles que le type de l'entité nommée identifiée (nom de personne, organisation, localisation, etc.).

Les règles décrites au sein de ces grammaires permettent de regrouper l'ensemble des éléments d'une même entité nommée. Elles sont généralement formées de mots déclencheurs, de formes provenant des Gazetteers, et, occasionnellement, de mots inconnus. Ces séquences de mots peuvent être parfaitement étiquetées lorsqu'elles apparaissent dans un certain contexte ou lorsqu'elles contiennent un mot déclencheur ou une entrée de nos gazetteers.

La prépondérance des formes inconnues dans les entités nommées implique un manque d'information qui, en plus des problèmes de détermination de mots d'arrêt qui permettent de décider où s'arrêter, augmente les probabilités d'erreurs de délimitation. Les grammaires syntaxiques de NooJ respectent certaines heuristiques lors de l'application des règles de reconnaissance. Elles

[45] Pour les noms de mois, nous avons dénombré 7 listes de noms de mois du calendrier:

- Hégire: مُحَرَّم, صَفَر, شَوَّال, شَعْبَان, رَمَضَان, رَجَب, رَبيع الأوَّل, رَبيع الثاني, ذُو القَعْدَة , ذُو الحَجَّة, جُمَادَى الأُولى,جُمَادَى الثاني.
- Arabe: يناير , فبراير, مارس, أبريل, مايو, يونيو, يوليو, أغسطس, سبتمبر, أكتوبر, نوفمبر, ديسمبر.
- Syrien: كانون الثاني , شباط, آذار, نيسان, أيار, حزيران, تموز, آب, أيلول, تشرين الأول, تشرين الثاني, كانون الأول.
- Tunisien/Algérien: جانفي , فيفري, مارس, أفريل, ماي, جوان, جويلية, أوت, سبتمبر, أكتوبر, نوفمبر, ديسمبر.
- Libyen: أين النار , النوار, الربيع, الطير, الماء, الصيف, ناصر, هانيبال, الفاتح, التمور, الحرث, الكانون.
- Mauritanien: يناير , فبراير, مارس, إبريل, مايو, يونيو, يوليو, أغشت, شتمبر, أكتوبر, نوفمبر, دجمبر.
- Marocain: يناير , فبراير, مارس, أبريل, ماي, يونيو, يوليوز, شتنبر, غشت, نونبر, أوكتوبر, دجمبر.

localisent le « Plus Long Patron / Longest Match » pour la seule grammaire et « Tous les Patrons / All Matches » pour la totalité des grammaires.

6.5 Reconnaissance des expressions numériques - NUMEX

6.5.1 Identification des déterminants numériques

Le problème de la reconnaissance automatique des déterminants numériques dans un texte fait partie des phénomènes linguistiques plus ou moins complexes. Généralement, ils ne peuvent pas être traités au niveau de l'analyse lexicale. Ils nécessitent des descriptions si redondantes qu'il serait très fastidieux, voire impossible, de les décrire manuellement dans des dictionnaires électroniques compilés sous forme d'automates finis.

En effet, si nous envisageons de reconnaître les déterminants numériques cardinaux au niveau lexical, cela nécessiterait le listage de tous les cardinaux, simples et composés, dont les valeurs varient de "واحد" (*waāḥid* – un, 1) jusqu'à "تسعة آلاف وتسع مائة وتسع وتسعون مليار" (*tis'at âlaāf wa-tis'a maāẏat wa-tis'a wa-tis'uwn milyaār* – neuf mille et neuf cent quatre-vingt dix-neuf milliards, 9 999 000 000 000) dans un dictionnaire électronique. Cependant, ce travail serait laborieux et sans intérêt d'autant plus que nous pouvons utiliser des grammaires locales sous forme de graphes NooJ pour décrire de tels phénomènes linguistiques et examiner leurs présences dans les textes.

Dès lors, nous nous intéressons, dans cette section, à l'identification des déterminants numériques écrits en toutes lettres[46] ainsi qu'à la détermination de leurs valeurs correspondantes. Par exemple, après reconnaissance de l'expression numérique "مائتان وأربع وَعِشْرُونَ" (*maāẏatayn wa-arba'a wa-'išruwna* - deux cent vingt quatre), nous donnons sa valeur numérique, soit 224. Ce travail se base sur la construction de grammaires locales de reconnaissances représentées sous forme de réseaux de transitions augmentés (ATNs)[47] à l'aide de l'éditeur graphique dans NooJ. Sur le plan pratique, ce type de représentation offre au moins trois avantages majeurs :

- La récursivité des appels entre graphes et sous-graphes ;
- La possibilité d'imposer des conditions sur les transitions ;
- La capacité d'associer des actions et des sorties aux transitions effectuées.

En effet, les mêmes séquences peuvent se retrouver plusieurs fois dans un nombre. Pour illustrer ce propos, nous considérons l'exemple "خمس آلاف وخمس مائة وخمس" (*ẖamsa âlaāf wa-ẖamsa maāẏat wa-ẖamsa* - cinq mille cinq cent cinq, 5505). Dans cet exemple, il suffit d'écrire un seul graphe et de l'appeler plusieurs dans la séquence globale.

[46] Dans ce travail, nous nous intéressons qu'à la reconnaissance des déterminants numériques représentant des nombres entiers et écrits en toutes lettres. A cet effet, nous signalons qu'en arabe les nombres décimaux ne semblent pas pouvoir être écrits en toutes lettres.

[47] Voir la Section 1.2.5 pour une description détaillée des réseaux de transitions augmentés ATNs

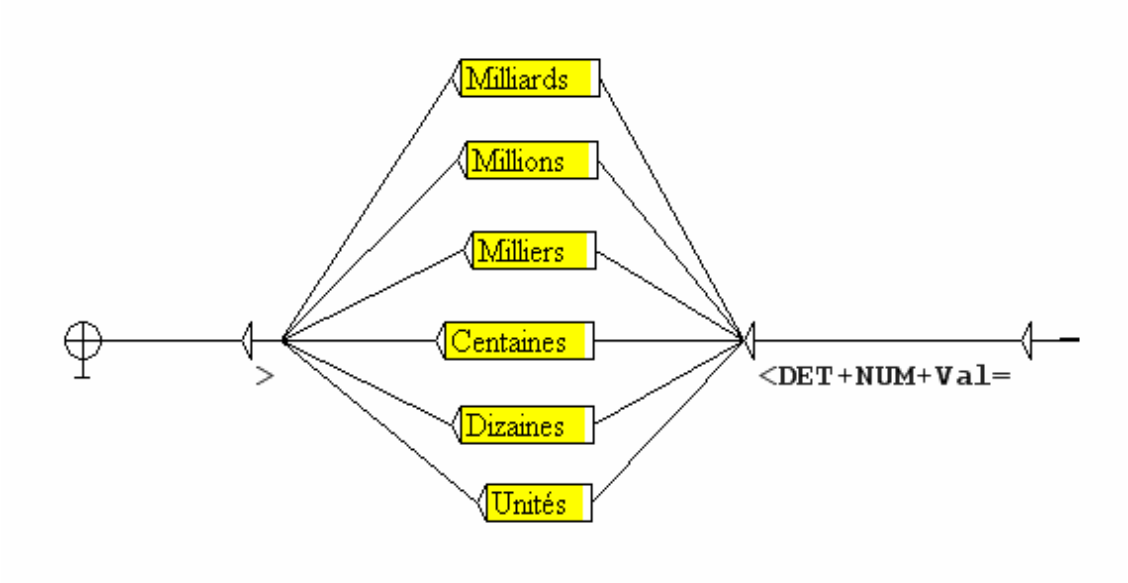


Figure 55 : Grammaire locale de reconnaissance des déterminants numériques : Graphe principal

Cette figure montre le graphe principal de reconnaissance. Celui-ci est restreint à des appels aux sous-graphes relatifs à l'identification des numéraux représentants des unités, dizaines, centaines et milliers. Comme sortie, nous attribuons à chaque syntagme retenu la catégorie grammaticale « DET » (un déterminant), l'information distributionnelle « +NUM » (numérique), ainsi que la valeur arithmétique qu'il représente « +Val= ». Ainsi, chaque numéral reconnu est accompagné de son équivalent écrit en chiffres. Ce type d'étiquettes est obtenu par l'introduction de symboles de sorties (productions ou outputs), marquées en dessous des nœuds des graphes. Toutefois, ceci a nécessité l'ajout de la valeur arithmétique aux entrées du dictionnaire représentant un entier naturel.

Nous notons que, les entrées introduites dans le dictionnaire sont restreintes aux nombres d'unité de "صِفْـر" (*ṣifr* – zéro, 0) à "تِـسْعَـة" (*tis'at* – neuf, 9), les nombres composés entre "أحَـد عَشَرَ" (*áaḥada 'ašara* – onze, 11) et "تِـسْعَـة عَشَرَ" (*tis'at 'ašara* – dix-neuf, 19) et les nombres de dizaines entre "عَشَرَة" (*'ašarat* – dix, 10) et "تِسعون" (*tis'uwna* – quatre-vingt, 90). A toutes ces entrées, nous avons ajouté une nouvelle propriété « +Val=… ». Parmi ces entrées, nous pouvons citer :

```
# Les nombres de 0 à 9
صِفْـر,N+Val=0+FLX=Flexion2
وَاحِـد,N+Val=1+FLX=Flexion2
اِثْـنَـان,N+Val=2+FLX=Flexion8
ثَـلَـاثَـة,N+Val=3+FLX=Numero1
أرْبَـعَـة,N+Val=4+FLX=Numero1
خَـمْـسَـة,N+Val=5+FLX=Numero1
سِـتَّـة,N+Val=6+FLX=Numero1
سَـبْـعَـة,N+Val=7+FLX=Numero1
ثَـمَـانِـيَـة,N+Val=8+FLX=Numero2
تِـسْـعَـة,N+Val=9+FLX=Numero1
```

```
# Les nombres de 11 à 19
أحَد عَشَرَ,N+Val=11+FLX=FlxOnze
اِثْـنَـا عَشَرَ,N+Val=12+FLX=FlxDouze
ثَـلَـاثَـة عَشَرَ,N+Val=13+FLX=FlxTreize
أرْبَـعَـة عَشَرَ,N+Val=14+FLX=FlxTreize
خَـمْسَة عَشَرَ,N+Val=15+FLX=FlxTreize
سِتَّـة عَشَرَ,N+Val=16+FLX=FlxTreize
سَـبْـعَـة عَشَرَ,N+Val=17+FLX=FlxTreize
ثَـمَـانِـيَـة عَشَرَ,N+Val=18+FLX=FlxTreize
تِـسْـعَـة عَشَرَ,N+Val=19+FLX=FlxTreize
```

Toutefois, en dépit de la facilité de formalisation sous forme de graphes NooJ, l'attribution de la valeur numérique correspondante à chaque déterminant reconnu ne peut qu'en augmenter la complexité de la description et la mise œuvre des règles de reconnaissance. Par exemple, compte tenu de l'omission du chiffre 0 (*ṣifr* – zéro,0) dans la lecture des cardinaux tel est le cas du chiffre des centaines dans le nombre 208 qui se lit "مَائَتَيْن وثَمَانِيَة" (*maāẏatayn wa-ṯamaāniyat* – deux cent huit) avec omission du zéro représentant le chiffre des dizaines, il faut donc introduire des ε-transitions au niveau des chiffres omis pour permettre l'insertion du « 0 » aux positions adéquates.

Afin de traiter ce phénomène dans ses plus fins détails, nous avons opté pour la création progressive des graphes. Dans un premier lieu, nous avons construit le graphe **Unités** donné ci-après. Il contient les cardinaux représentant les nombres unitaires compris entre 1 et 9. Le symbole <اِثْنَان > (<*iṯnaāni*>) représente toutes les formes rattachées à l'entrée "اِثْنَان " (*iṯnaāni* – deux, 2), soit "اِثْنَان" (*iṯnaāni* – deux, au nominatif, masculin), "اِثْنَتَان" (*iṯnataāni* – deux, au nominatif, féminin), "اِثْنَيْن" (*iṯnayni* – deux, accusatif, masculin) et "اِثْنَتَيْن" (*iṯnatayni* – deux, accusatif, féminin). Ce graphe sera appelé dans les graphes représentants les dizaines, centaines, milliers, millions et milliards.

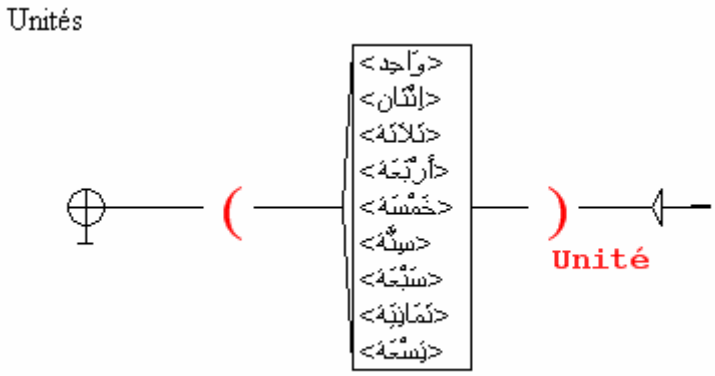


Figure 56 : Grammaire locale de reconnaissance des déterminants numériques : Graphe des Unités (cardinaux de 1 à 9)

Dans un second lieu, nous avons construit le graphe **Dizaines** qui décrit les déterminants compris entre 11 et 99. Il contient, entre autres, les séquences "ثَمَانِيَة عَشَرَ " (*ṯamaāniyat ʿašara* – dix-huit, 18), "أرْبع وَعِشْرُونَ" (*áarbaʿ wa-ʿišruwna* – vingt-quatre, 24), "اِثْنَين وَخَمْسِينَ " (*iṯnayni wa-hamsiyna* – cinquante-deux, 52), "تِسعون" (*tisʿuwna* – quatre-vingt-dix, 90), etc. Notons que, lorsqu'il s'agit d'un nombre de dizaine ayant un chiffre d'unités non nul, une conjonction de coordination <CONJ> assure la liaison entre les deux. Pour exprimer les nombres de dizaines composés, nous faisons appel au graphe **Unités** créé auparavant. La possibilité d'utilisation de variables au sein des grammaires locales, nous permet de regrouper les chiffres du même ordre

(unités, dizaines, centaines, etc.) et d'éviter le listage de toutes les formes potentielles ainsi que leurs valeurs numériques correspondantes de façon indépendantes dans un même nœud. Par exemple, au cas où la variable $Dix2 contient une forme fléchie de " ثَلاثَة عَشَرَ ", une simple consultation du dictionnaire à l'aide de la production $Dix2$Val permet de constituer la sortie correspondante, soit la valeur « 13 », à partir d'une lecture de la propriété « +Val= » ajoutée aux entrées du dictionnaire.

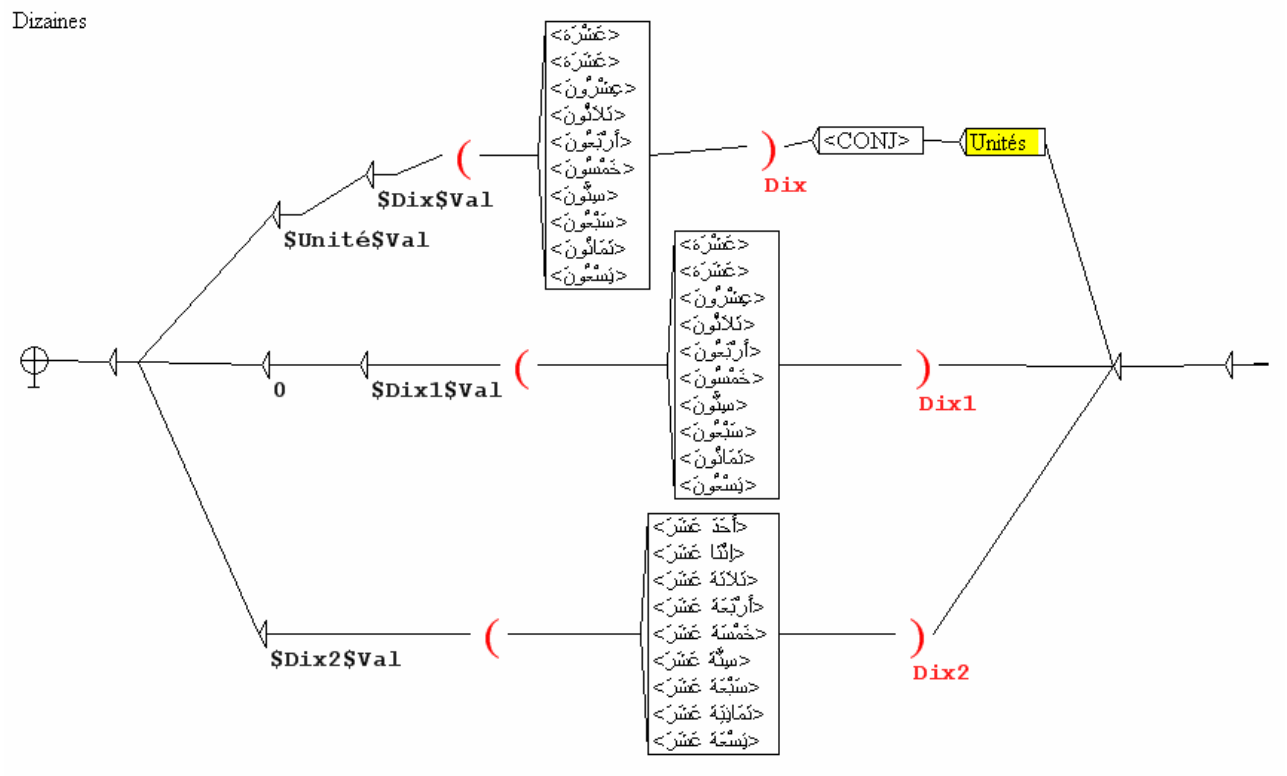


Figure 57 : Grammaire locale de reconnaissance des déterminants numériques : Graphe des Dizaines (cardinaux de 10 à 99)

Dans un troisième lieu, nous avons construit le graphe **Centaines** qui décrit les déterminants compris entre 100 et 999. Ce graphe fait appel aux deux graphes précédents pour l'identification des dizaines et des unités. Pour ce type de déterminants numériques, nous tenons compte des deux variantes orthographiques du mot désignant les centaines, soit "مَائَة" (*maāýat* – cent, 100) et "مِئَة" (*maýat* – cent, 100). Nous notons que lors de la construction de ce graphe, nous avons constaté que certains termes comme "مَائَة" (*maāýat* – cent, 100), "أَلْف" (*áalf* – un mille, 1000) et "مَلْيُون" (*malyuwn* – un million, 1 000 000) nécessitent l'utilisation des symboles grammaticaux suivants :

- <N+s> : désigne la forme au singulier. Cette forme est utilisée pour désigner des valeurs unitaires telles que 100, 1 000 ou 1 000 000
- <N+d> : désigne les formes duales telles que 200, 2 000 et 2 000 000
- <N+p> : désigne tous les multiples compris entre 3 et 10 tels que 3 000 ou 9 000 000

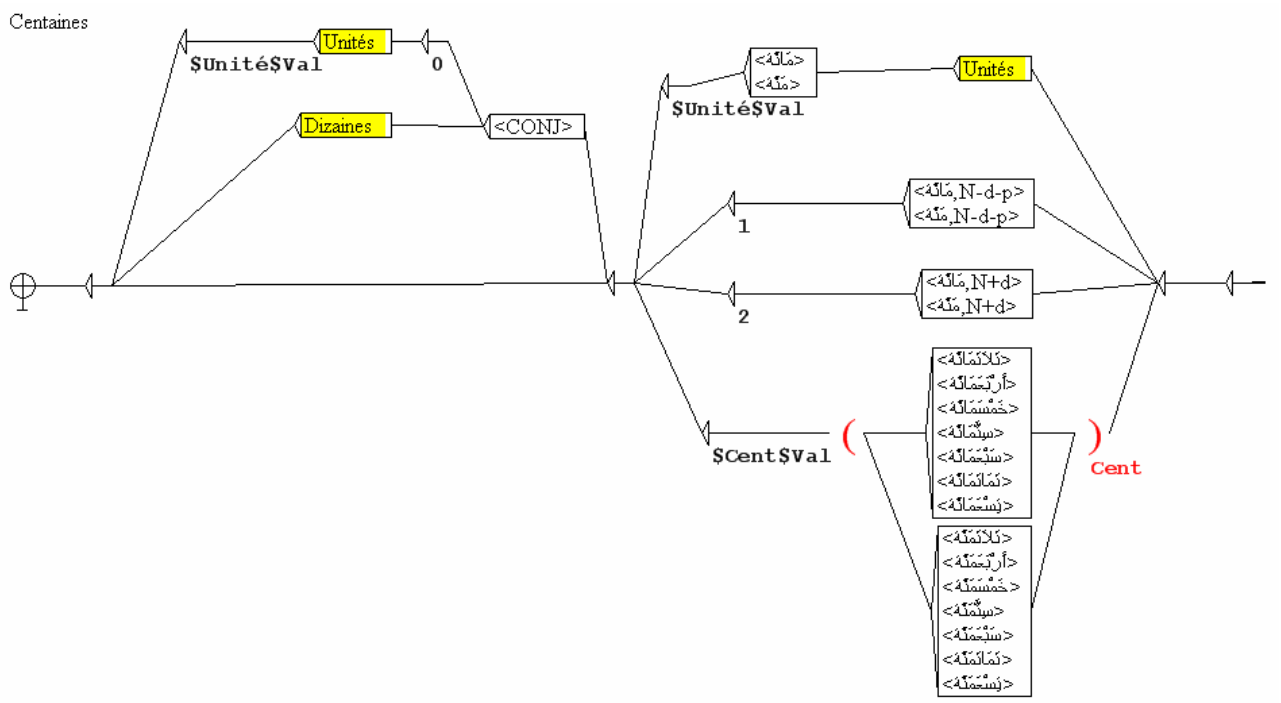


Figure 58 : Grammaire locale de reconnaissance des déterminants numériques : Graphe des Centaines (cardinaux de 100 à 999)

Dans un quatrième lieu, nous avons construit le graphe **Milliers** qui décrit les déterminants compris entre 1 000 et 999 999. Ce graphe fait appel aux trois graphes précédents pour l'identification de ses centaines, dizaines et unités.

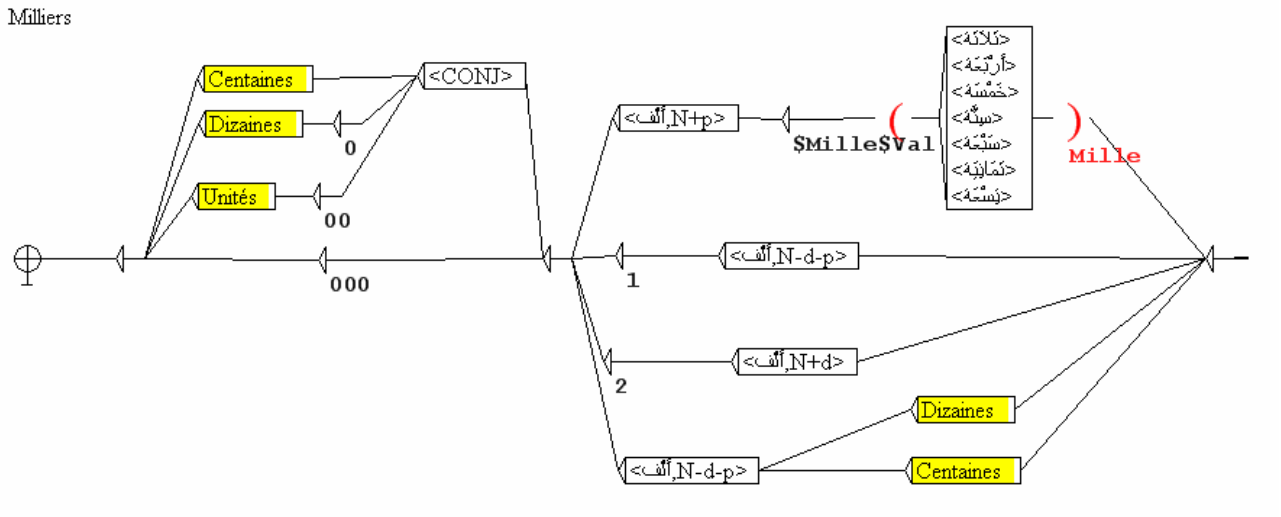


Figure 59 : Grammaire locale de reconnaissance des déterminants numériques : Graphe des Milliers (cardinaux de 1 000 à 999 999)

Et, finalement, nous avons construit les deux sous-graphes qui décrivent les déterminants numériques représentant les **Millions** et les **Milliards**. Nous notons que, le plus souvent, ces derniers types de déterminants apparaissent accompagnés par des symboles de devises dans des expressions monétaires, voir section 6.5.2.

La grammaire construite est ensuite appliquée sur un texte pour étudier sa couverture et vérifier la validité des valeurs numériques attribuées automatiquement par cette grammaire à variables. Le résultat de l'application de cette grammaire fournit la table de concordances suivante :

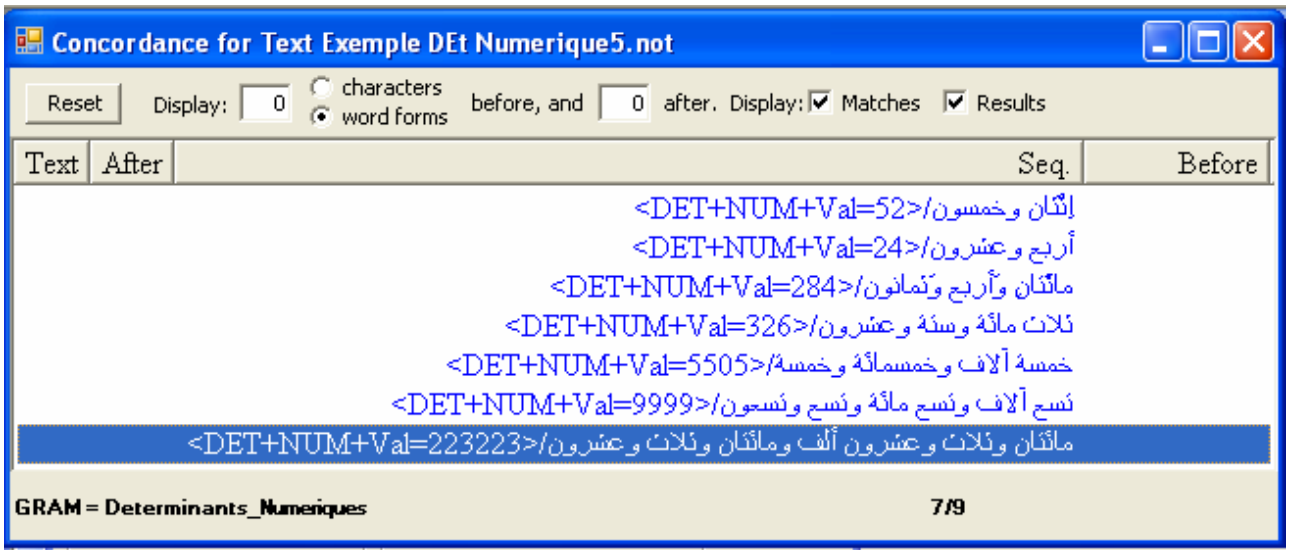


Figure 60 : Table des concordances de la grammaire locale de reconnaissance des déterminants numériques

Chaque ligne de cette table, rappelons-le, affiche la séquence retenue et sa valeur numérique assignée automatiquement entourées par les contextes droit et gauche. En l'occurrence, la première ligne atteste la reconnaissance du déterminant "إثنَان وخمسون" et affiche la valeur numérique correspondante, soit 52.

6.5.2 Identification des expressions numériques

De la même façon que pour les expressions temporelles, les expressions numériques sont généralement identifiables à l'aide de listes statiques de mots déclencheurs tels que les unités de distances et de poids, ainsi que les noms de devises et leurs subdivisions potentielles, etc.

L'étude que nous avons menée sur les expressions numériques, nous a rapidement montré leur structure syntaxique essentiellement formée d'un déterminant numérique, suivi d'une unité de mesure et éventuellement précédé d'un verbe déclencheur.

Ainsi, dans un premier lieu, nous avons associé une priorité plus haute à la grammaire de reconnaissance des déterminants numériques, décrite à la section 6.5.1, afin de pouvoir identifier les cardinaux écrits en toutes lettres à l'aide du symbole grammatical <DET+NUM>. De plus, nous avons construit une grammaire locale, ci-après, pour identifier les nombres écrits à l'aide des chiffres arabes y compris les nombres composés, négatifs, à virgule ou en écriture scientifique.

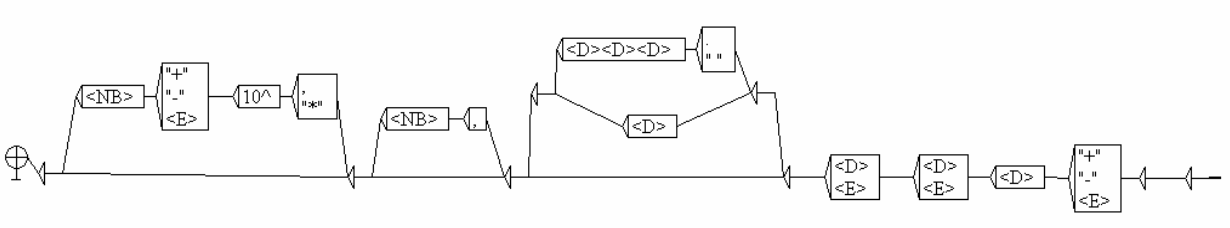


Figure 61 : Grammaire locale de reconnaissance des nombres

Dans un second lieu, nous avons procédé à leurs classifications en quatre sous-catégories. Pour chacune des sous-catégories, nous avons regroupé l'ensemble des unités qui y sont utilisées ainsi qu'un certain nombre de verbes et expressions déclencheurs. En effet, nous avons distingué :

- **Les expressions de mesure** ayant le mètre comme unité de base. A ce niveau, nous avons répertorié les unités de mesure en regroupant tous les multiples ainsi que les sous-multiples tels que كيلومتر (*kiyluwmitr* – kilomètre, km), ملّيمتر (*milliymitr* – millimètre, mm), etc. Ces entrées lexicales servent pour la reconnaissance des distances simples (longueurs, largeurs, profondeurs, hauteurs, etc.) ainsi que les mesures composées telles que les mesures de surfaces obtenues en les faisant suivre du mot clé مُرَبَّع (*murabba'* – carré) et les mesures de volumes au moyen du mot clé مُكَعَّب (*muka''ab* – cube). Nous avons aussi fait usage de verbes ou de groupes verbaux déclencheurs tels que : قـاسَ (*qaāsa* – mesurer), قيَّسَ (*qayyasa* – mesurer), اِمْتَدَّ عَلَى (*imtadda 'alaq* – s'étendre sur), وَقَعَ عَلَى اِرْتِفاع (*waqa' 'alaq irtifaā'* – être à une hauteur de), etc.
- **Les expressions de poids :** A l'instar de l'exemple des expressions de mesure, nous avons regroupé tous les multiples et sous multiples de l'unité de base غرام (*ġraām* - le gramme, g). Par la suite, nous avons listé l'ensemble des verbes qui peuvent devancer de telles expressions tels que : وَزَنَ (*wazana* – peser), etc.
- **Les expressions monétaires :** En ce qui concerne ce type d'expressions, nous avons dénombré 175 unités incluant des devises telles que دينار (*diynaār* – Dinar) ou دولار (*duwlaār* – Dollar) ainsi que leurs subdivisions telles que سنتيم (*santiym* – Centimes) ou مِلّيم (*milliym* – Millimes). Parmi les verbes déclencheurs des expressions monétaires, nous citons : دَفَعَ (*dafa'a* – payer), تَكَلَّفَ (*takallafa* – coûter), اِرْتَفَعَ ثَمَنه إلى (*irtafa'a ṯamanuhu ilaq* – son prix s'élève à), etc.
- **Les expressions de pourcentages :** Les règles de reconnaissance de ce type d'expressions sont les plus simples à mettre en œuvre puisqu'elles sont, exclusivement, formées à l'aide d'un cardinal ou d'un déterminant numérique <DET+NUM> ou un nombre écrit en chiffres arabes (voir la grammaire de la Figure 61), suivi du symbole de pourcentage « % » ou de la forme بالمائة (*bil-miy'at* – pour cent). A ce propos, nous signalons que, dans notre corpus d'étude, les expressions de pour-mille sont rares et ne représentent que 0,5% de l'ensemble des expressions numériques identifiées.

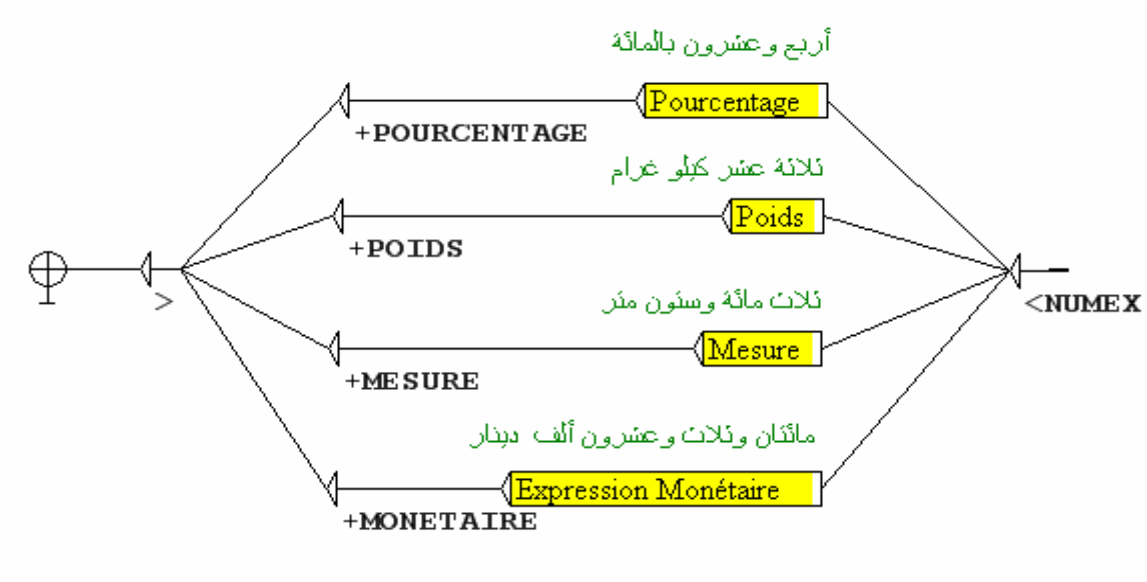


Figure 62 : Grammaire locale de reconnaissance des expressions numériques : Graphe principal (NUMEX)

A l'issue de l'écriture des règles de reconnaissance, nous avons appliqué toutes les grammaires construites sur notre corpus. Grâce aux différentes fonctionnalités de tri et de filtrage des résultats exposés dans une table de concordance, nous avons réussi à alimenter nos listes de mots déclencheurs et à décrire de nouvelles règles contextuelles afin d'améliorer et d'affiner les résultats de notre système. Cette expérience nous a aussi révélé quelques problèmes liés, en premier lieu, à l'identification des entités nommées et, en second lieu, à leur typage tel que décrit dans la section suivante.

6.5.3 Désambiguïsation à l'aide des déterminants numériques

Lors de notre étude sur les déterminants numériques, nous nous sommes aperçus d'un autre phénomène linguistique qui vient s'ajouter à l'ensemble des phénomènes morpho-syntaxiques spécifiques à la langue arabe : il s'agit d'un anti-accord entre le déterminant numérique et le nom qui le suit.

En effet, certains numéraux portent le genre inverse du nom qui suit : lorsqu'un nom est masculin, le numéral porte la marque du genre[48] ; par contre, lorsqu'un nom est féminin, le numéral ne porte pas de marque de genre. Nous proposons une analyse de ce que nous convenons à appeler un anti-accord [Arbaoui, 2005] :

- Pour désigner le singulier et le dual, les déterminants numériques ne sont utilisés que dans des cas très particuliers ;
- Quand la valeur du numéral est comprise entre 3 et 10 : le déterminant porte le genre inverse que le nom qui le suit : il est au féminin si le nom qui le suit est masculin et inversement. Par exemple, le déterminant numérique "خمسة" (*ẖamsat* – cinq, 5) peut accepter deux différentes formes :
  - ✓ Celle qui porte la marque du féminin : "خمسة" (*ẖamsat* – cinq, 5) : elle ne précède que les noms au masculin comme pour خمسة كُتُبٍ (*ẖamsat kutubin* – cinq livres) ;
  - ✓ Celle qui ne porte pas la marque du féminin : "خمس" (*ẖamsa* – cinq, 5) : elle ne précède que les noms au féminin comme pour خمس سَيّـارات (*ẖamsa sayyaāraāt* – cinq voitures).
- Lorsque la valeur du numéral est égale à 11 ou 12, il devient un mot composé (le chiffre des unités et celui des dizaines) et s'écrit comme une expression figée avec une a / ´«فتحة» (*fatḥä*) finale. Dans ce cas, les deux éléments du numéral s'accordent au genre avec le nom qui suit. Par exemple, nous trouvons :
  - ✓ أحد عشرَ كتابا (*aḥada 'ašara kitaāban* – onze livres) où كتابا (*kitaāb* – un livre, accusatif, indéfini) est un nom masculin;
  - ✓ إثنتا عشرةَ سَيّـارة (*iṯntaā 'ašaraṯa sayyaāratan* – douze voitures) où سَيّـارة (*sayyaāratan* – une voiture) est un nom féminin.
- Lorsque la valeur du numéral est comprise entre 13 et 19, il s'écrit toujours comme un mot composé avec une a / ´«فتحة» (*fatḥä*) finale. Dans ce cas, le premier élément du numéral, dont la valeur varie de 3 à 9, prend le genre inverse que celui du nom qui le suit, c'est-à-dire qu'il se met au féminin si le nom est masculin et au masculin si le nom est féminin : nous parlons d'un anti-accord. Par contre, le deuxième élément du numéral, qui est toujours 10, a le même genre que le nom qui suit. Par exemple, nous trouvons :
  - ✓ خمسة عشرَ كتابا (*ẖamsaṯa 'ašara kitaāban* – quinze livres) où كتابا (*kitaāb* – un livre, accusatif, indéfini) est un nom masculin;
  - ✓ خمس عشرةَ سَيّـارة (*ẖamsa 'ašaraṯa sayyaārat* – quinze voitures) où سَيّـارة (*sayyaārat* – une voiture) est un nom féminin.
- Lorsque la valeur du numéral est égale à 20, 30, 40, …, 90, celui-ci reste invariable indépendamment du genre du nom qui le suit. Par exemple, nous trouvons :

[48] Nous notons que, en arabe, les noms au féminin porte la lettre finale "ة" (*ẖ - tâ' marbûṭä*) comme marque du genre.

  - ✓ ستّون كتابا (*sittuwn kitaāban* – soixante livres) où كتابـا (*kitaāban* – un livre, accusatif, indéfini) est un nom masculin ;
  - ✓ ستّون سَيّـارة (*sittuwn sayyaārat* – soixante voitures) où سَيّـارة (*sayyaārat* – une voiture) est un nom féminin.
- Lorsque la valeur du numéral est de 24, 35, 76, etc., celui-ci s'accorde partiellement avec le genre du nom qui le suit. En effet, tandis que le premier élément du numéral, qui représente les unités de 3 à 9 s'accorde avec le genre du nom successeur, le deuxième élément qui représente les 20, 30, …, 90, reste invariable. Par exemple, nous trouvons :
  - ✓ خمس وعشرون كتابا (*hamsun wa-'išruwn kitaāban* – vingt-cinq livres) où كتابا (*kitaāban* – un livre, accusatif, indéfini) est un nom masculin ;
  - ✓ خمسة وعشرون سَيّـارة (*hamsata wa-'išruwn sayyaārat* – vingt-cinq voitures) où سَيّـارة (*sayyaārat* – une voiture) est un nom féminin.
- Les numéraux "مَائة" (*maāýat* – cent, 100), "ألف" (*áalf* – un mille, 1000) et "مَلْيُون" (*malyuwn* – un million, 1 000 000) restent toujours invariables indépendamment du genre du nom qui leur suit. Par exemple, nous trouvons :
  - ✓ مَائة كتاب (*maāýat kitaābin* – cent livres) où كتاب (*kitaāb* – un livre) est un nom masculin;
  - ✓ ألف سَيّـارة (*áalf sayyaārat* – vingt-cinq voitures) où سَيّـارة (*sayyaārat* – une voiture) est un nom féminin.

Cet ensemble d'observations nous révèle la variété du comportement des numéraux vis-à-vis du genre par la présence de cas d'invariabilité, cas d'accord et cas d'anti-accord.

Ainsi, la reconnaissance de ces déterminants et l'attribution de leurs valeurs numériques peuvent servir pour réduire l'ambiguïté au niveau du genre du nom successeur. Ils peuvent, aussi, servir pour réduire l'ambiguïté au niveau de :

- *la voyelle casuelle du nom successeur* : si un nom est précédé par un déterminant numérique, sa flexion casuelle dépend de la valeur de celui-ci :
  - ✓ Si la valeur du déterminant numérique est comprise entre 3 et 10, égale à un nombre entier de centaines (100, 200, …, 900) ou un nombre entier de milliers (1000, 2000, …, 9000) aussi bien qu'un nombre entier de millions ou milliers, le nom successeur prend la marque du génitif ;
  - ✓ Dans tous les autres cas, le nom successeur prend la marque de l'accusatif ;
- *le nombre du nom successeur* : à l'opposé des cardinaux dont la valeur varie de 3 à 10 et qui nécessitent un nom au pluriel, toutes les autres valeurs doivent être suivies d'un nom au singulier.

Pour illustrer ce propos, nous proposons les deux déterminants numériques suivants dont les annotations s'écrivent, rappelons-le, sous la forme :

```
<DET+NUM+Val=9>تِسعة</DET>
<DET+NUM+Val=224>مائَتانِ وَأربع وَعِشْرُونَ</DET>
```

Généralement, ces déterminants sont utilisés pour désigner un nombre d'objets ou un groupe de personnes. Par exemple, nous disons:

تسعة رِجال (*tis'at riğaālin* – neuf hommes)
مائَتانِ وَأربع وَعِشْرُون رجلا (*maāýatayn wa-arba'a wa-'išruwna rağulan* – deux cent vingt-quatre homme)

Dans le premier cas, lorsque le déterminant numérique "تِسعة" (*tis'at* – neuf) a pour valeur "9", le nom successeur se met, automatiquement, au pluriel, génitif, رجال (*riğaāl*in – des hommes). Par contre, dans le second exemple, si le déterminant numérique "مائَتان وَأربع وَعِشْرُون " a pour valeur "224", le nom qui suit se met au singulier, رَجُلًا (*rağulan* – un homme).

Lorsque le nom qui suit le déterminant numérique ne présente pas d'ambiguïté, l'apport de cette démarche reste minime. Toutefois, si la forme nominale est, elle-même, ambiguë et peut représenter aussi bien une forme au singulier, qu'une forme au pluriel, le résultat est beaucoup plus intéressant. En effet, si nous reprenons ces mêmes déterminants avec une forme ambiguë telle que le mot non voyellé "كتب" (*ktb*), nous obtenons :

تسعة كتب (*tis'at kutubin* – neuf livres)
مائة كتب (*maāýata katbin* – cent écrits)

Nous rappelons que, parmi les 17 voyellations acceptées par cette forme non voyellée "كتب" (*ktb*), voir section 2.5.1, une dizaine d'entre-elles désignent un nom:

Voyellation	Translittération	Traduction
كُتُبُ	*kutubu*	des livres (forme au pluriel, cas nominatif)
كُتُبَ	*kutuba*	des livres (forme au pluriel, cas accusatif)
كُتُبِ	*kutubi*	des livres (forme au pluriel, cas génitif)
كُتُبٌ	*kutubun*	des livres (forme au pluriel, cas nominatif, indéfini)
كُتُبٍ	*kutubin*	des livres (forme au pluriel, cas génitif, indéfini)
كَتْبُ	*katbu*	un écrit (forme au singulier, cas nominatif)
كَتْبَ	*katba*	un écrit (forme au singulier, cas accusatif)
كَتْبِ	*katbi*	un écrit (forme au singulier, cas génitif)
كَتْبٌ	*katbun*	un écrit (forme au singulier, cas nominatif, indéfini)
كَتْبٍ	*katbin*	un écrit (forme au singulier, cas génitif, indéfini)

Compte tenu de la valeur du déterminant numérique, le même mot non voyellé donne deux formes différentes :

- Dans le premier cas, la valeur du déterminant est égale à "9", la forme qui suit est alors au pluriel, génitif. Le mot non voyellé peut être alors l'une des formes fléchies plurielles au génitif désignant la forme plurielles « des livres » : "كُتُبِ" (*kutubi*, des livres, cas génitif) ou "كُتُبٍ" (*kutubin*, des livres, cas génitif, indéfini). Ainsi, nous ne retenons que deux voyellations parmi les 17 initialement attribuées par analyse morpho-lexicale. En conséquence, nous réduisons l'ambiguïté pour ne proposer que 11% des annotations initiales ;
- Dans le second cas, la valeur du déterminant est de "100", la forme qui suit est alors mise au singulier, génitif. Le mot non voyellé peut être alors l'une des deux formes fléchies, au singulier et au génitif, désignant « un écrit» : "كَتْبِ" (*katbi*, un écrit, cas génitif) ou "كَتْبٍ" (*katbin*, un écrit, cas génitif, indéfini).

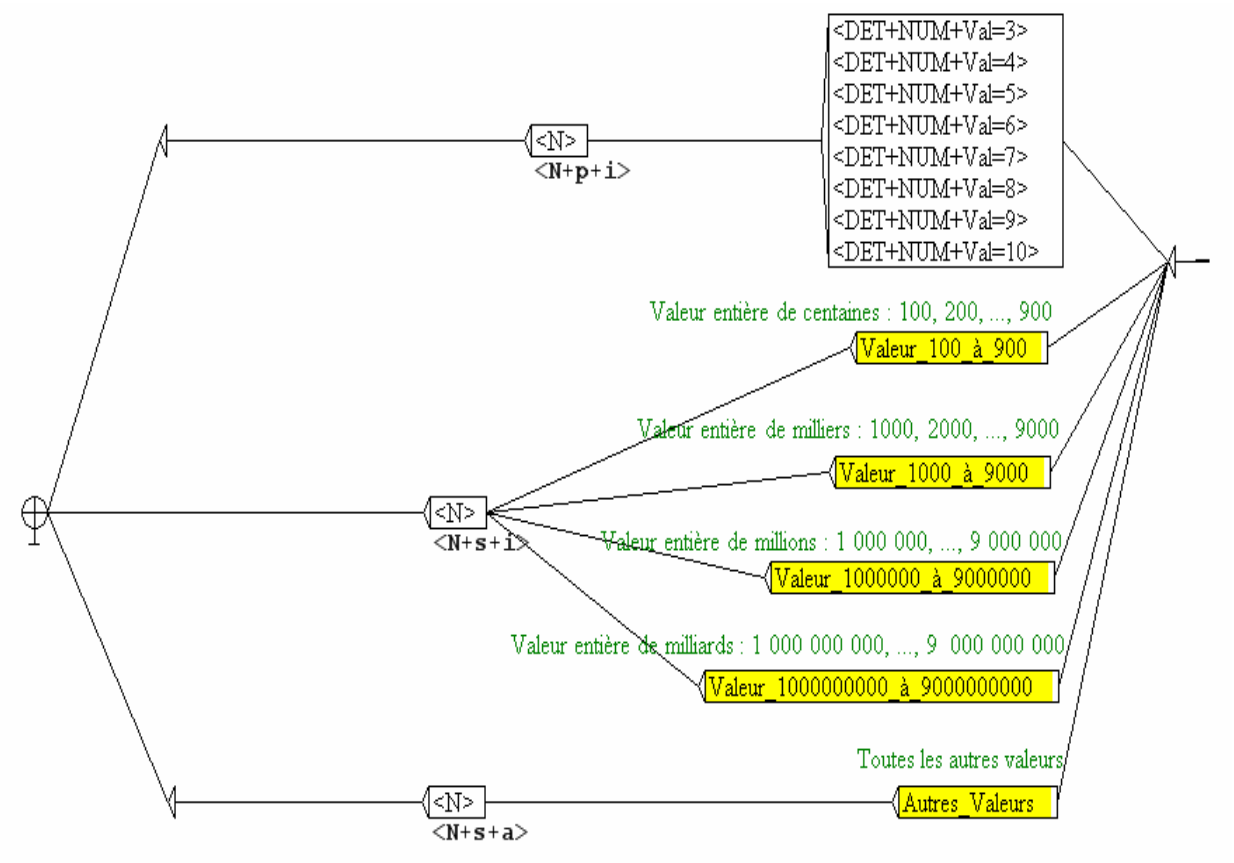


Figure 63 : Grammaire locale de désambiguïsation des formes nominales : Utilisation des déterminants numériques

Dans la grammaire ci-dessus, nous avons trois types de contraintes :

- Si le déterminant a une valeur comprise entre 3 (<DET+NUM+Val=3>) et 10 (<DET+NUM+Val=10>), le nom successeur est restreint aux formes plurielles déclinées au génitif : <N+p+i> ;
- Si le déterminant a une valeur entière de centaines, de milliers, de millions ou de milliards, le nom successeur est restreint aux formes au singulier déclinées au génitif : <N+s+i> ;
- Pour toutes les autres valeurs, le nom successeur est restreint aux formes au singulier déclinées à l'accusatif : <N+s+a>.

6.6 Reconnaissance des expressions temporelles - TIMEX

Les expressions temporelles, TIMEX, sont aussi importantes que les expressions numériques dans les systèmes d'analyse syntaxique ou d'extraction d'information [Poibeau, 2003]. En effet, un utilisateur peut interroger notre système pour obtenir des informations sur un événement. Habituellement, tout événement est lié à une date ou une heure représentées sous forme d'une expression temporelle.

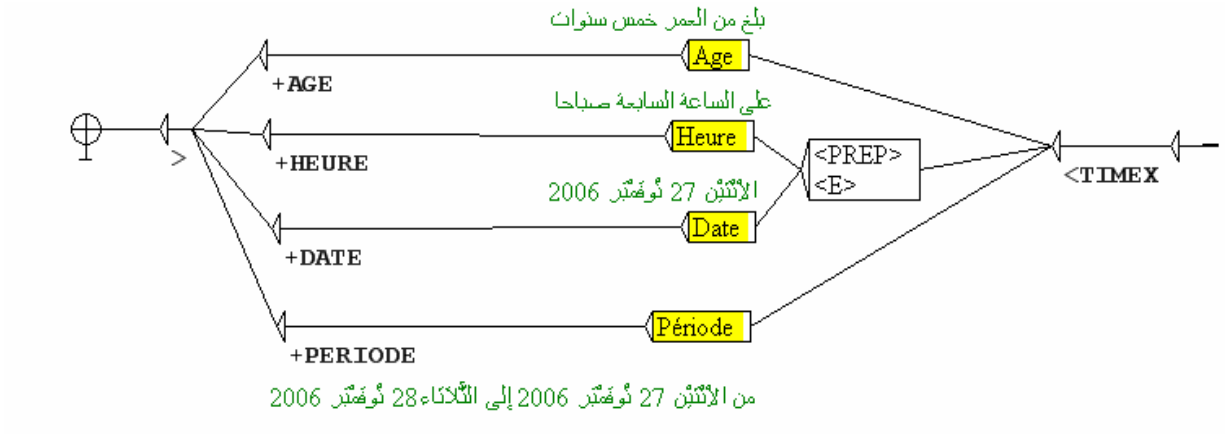


Figure 64 : Grammaire locale de reconnaissance des expressions temporelles : Graphe principal (TIMEX)

Notre système à base de règles permet d'identifier les expressions temporelles (<TIMEX>) et de leur attribuer une catégorisation parmi les 4 sous-catégories distinguées au sein de notre grammaire :

- **Les expressions de date** : nous distinguons, en arabe, 3 types d'écritures des expressions de date :
  - ✓ *Les dates avec un seul système de date* comme le système arabe, hégire ou occidental. Nous pouvons citer : 2000 في 2 ديسمبر (*fiy 2 diysambar 2000* – le 22 décembre 2000), يَوْم الِاثْنَيْن 27 نُوفَمْبَر 2006 (*yawm al-iṯnayn 27 nuwfambar 2006* – le lundi 27 novembre 2006), فِي بِدَايَة شَهْر جُوان 2007 (*fiy bidaāyat šahr ğuwaān 2007* – au début du mois de juin 2007);
  - ✓ *Les dates avec deux systèmes de date*, notamment, le système arabe et le système grégorien : 2006 في 22 كانون الأول/ديسمبر عام (*fiy 2 kaānuwn al-áwwal/diysambar 2006* – le 22 kanoun al awal/décembre 2006),1999 في التاسع من نيسان/أبريل عام (*fiy al-taāsi' min naysaān/abriyl 'aām 1999* – le neuf naysan/avril année 1999), etc. ;
  - ✓ *Les dates qui fournissent la date en arabe* (calendrier grégorien) *et son équivalent dans le système des dates en Hégire* (calendrier musulman) :
    في شهر فبراير عام 1812ميلاديّ الموافق لشهر صفر 1227 هجريّ (*fiy šahr febrayir 'aām 1812 miylaādiyy al-muwafiq li-ṣifr 1227 hiğriyy* – au mois de février année 1812 après Jésus-Christ correspondant au mois de Safar 1227 de l'Hégire) ;

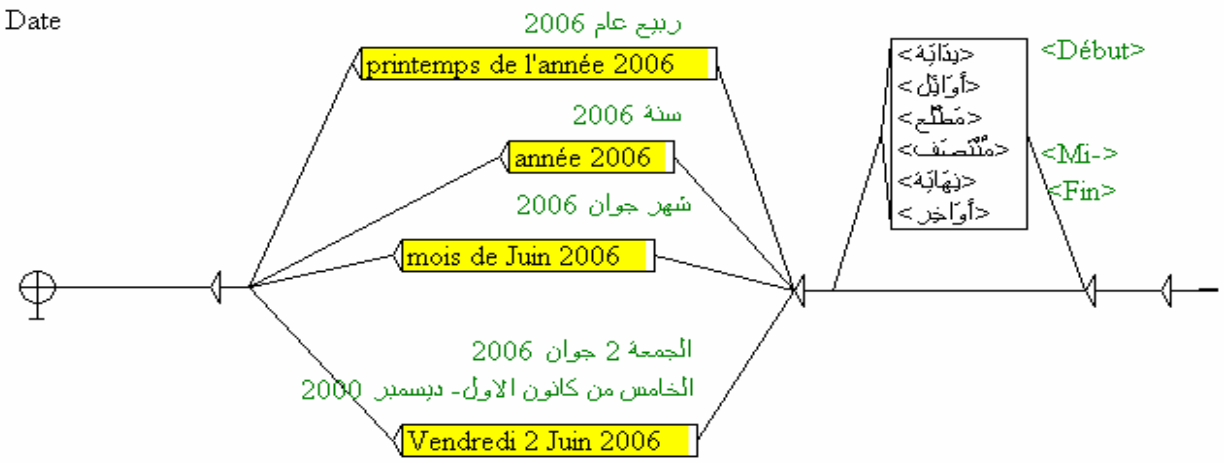


Figure 65 : Grammaire locale de reconnaissance des expressions temporelles : Reconnaissance des dates (TIMEX+DATE)

- **Les expressions d'heures** : على الساعة السابعة صباحا (*'laq al-saā'at al-saābi'at ṣabaāḥan* – à sept heures du matin) ;
- **Les expressions de période ou de durée** :
  - ✓ 1976 ما بين 13 و15 أيلول/سبتمبر من العام (*maā bayna 13 wa-15 áyluwl/sibtambar min al-'aām 1976* – entre le 13 et le 15 ayloul/septembre de l'année 1976) ;
  - ✓ 2006 من الِاثنَيْن 27 نُوفَمْبَر إلى الثُّلاثاء28 نُوفَمْبَر (*min al-iṯnayn 27 nuwfambar ilaq al-ṯulaṯa' 28 nuwfambar 2006* – du lundi 27 novembre au mardi 28 novembre 2006)
  - ✓ 2000 في الأسبوع الأول من تشرين الأول/أكتوبر (*fiy al-ùsbuw' al-áwwal min tašriyn al-áwwal/ùctuwbar 2000* – durant la première semaine du mois tachrine al-awal / octobre 2000) ;

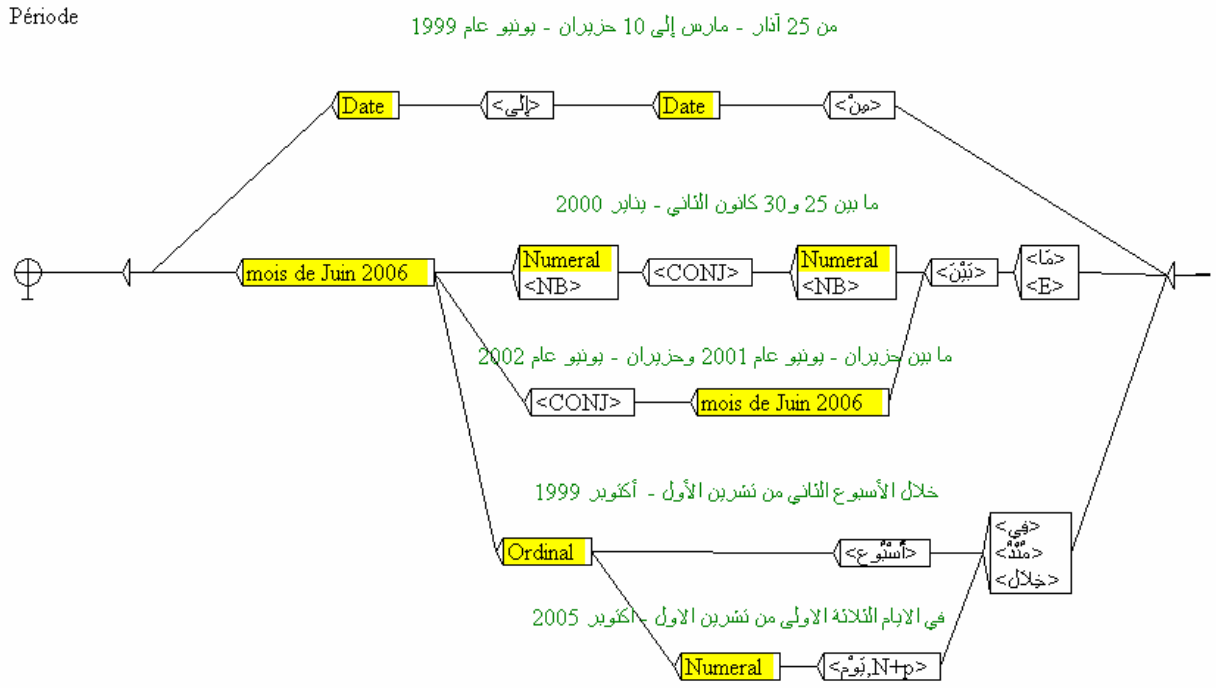


Figure 66 : Grammaire locale de reconnaissance des expressions temporelles : Reconnaissance des périodes et durées (TIMEX+PERIODE)

- **Les expressions d'âge** : بلغ من العمر خمس سنوات (*balaġa min al-'umur ḫams sanawaāt* – il a atteint l'âge de cinq ans). Nous notons que dans le sous-graphe de reconnaissance de telles expressions, nous avons fait appel aux annotations attribuées aux déterminants numériques au moyen de la grammaire locale décrite dans la section 6.5.1.

Pour l'écriture des règles de reconnaissance des expressions temporelles, nous avons dénombré une vingtaine d'entre elles en analysant quelques expressions manuellement rassemblées à partir de nos corpus d'évaluation. Lors de cette phase, nous avons remarqué que, contrairement au traitement des noms propres (personnes, lieux et organisations), l'identification des expressions temporelles dispose, généralement, de suffisamment d'indices structuraux et contextuels. En fait, elles sont, presque toutes, identifiables grâce à des listes statiques de mots déclencheurs (noms de jours, de mois, etc.).

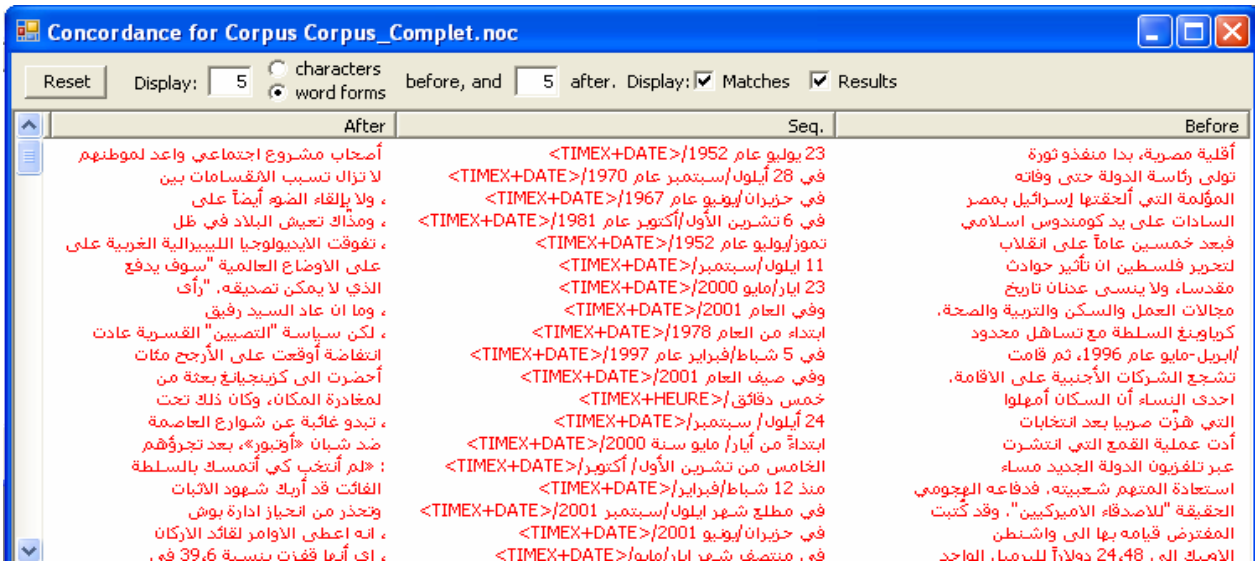


Figure 67 : Table de concordances des expressions temporelles

6.7 Reconnaissance des noms propres – ENAMEX

6.7.1 Identification des noms de personnes

Afin de pouvoir reconnaître automatiquement les noms de personnes, nous avons commencé par l'élaboration d'un dictionnaire de prénoms qui contient environ 12 400 prénoms arabes et prénoms étrangers transcrits reconnaissables par le bais de l'étiquette <N+Prénom>. Les entrées de ce dictionnaire sont de la forme :

أَحْمَد,N+m+Prénom+Hum (*áḥmad* – Ahmed)
بِيَاتْرِيس,N+f+Prénom+Hum (*biyaātriys* – Béatrice)

Outre les entrées simples, ce dictionnaire contient des formes composées telles que :

زِين الدِّين,N+m+Prénom+Hum (*ziyn al-diyn* – Zinedine)
عَبْد الْكَرِيم,N+m+Prénom+Hum (*'abd al-kariym* – Abdelkarim)

Par contre, il ne contient ni les compositions de prénoms telles que "مُحَمَّد أمين" (*muḥammad ámiyn* - Mohamed-Amine) ou "جَان مَاري" (*ğaān maāriy* - Jean Marie), ni les prénoms islamiques introduits par les éléments lexicaux tels que ابْن (*ibn* - le fils de), بن (*bin* - le fils de), أَبُو (*ábuw* - le père de), etc. Pour reconnaître ces prénoms, il nous a suffit de construire la grammaire locale ci-après:

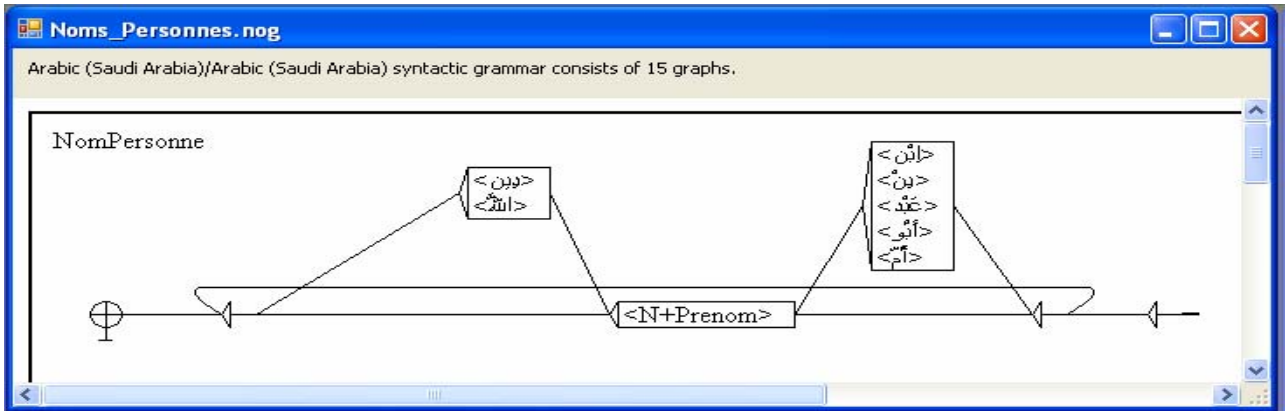


Figure 68 : Grammaire locale de reconnaissance des prénoms composés

Les prénoms ainsi recensés dans des dictionnaires électroniques ou reconnus par grammaire locale servent de preuves internes pour l'identification des noms de personnes. En l'occurrence, la présence du prénom "فرانسوا" (*fraāniswaā* – François), appartenant à notre liste de prénoms, permet de deviner que la séquence "فرانسوا فيون " (*fraāniswaā fiyuwn* – François Fillon) est un nom de personne.

Quant aux preuves externes, nous avons recensé d'autres listes de mots déclencheurs tels que les noms de profession, les titres, etc. Ces listes sont d'une très grande utilité pour la tâche en question ; outre la reconnaissance des noms de personnes, elles permettent la catégorisation de celles-ci. Par exemple, la présence d'un titre politique tel que "الوزير الأوّل" (*al-waziyr al-áwwal* – le Premier Ministre) avant un nom de personne nous permet d'attester la présence d'un syntagme nominal désignant un nom de personne même en cas d'omission du prénom de cette personne ou d'absence du prénom qui s'y rattache dans la liste que nous avons recensé.

Par ailleurs, nous distinguons 8 sous-catégories de noms de personnes. Pour les différencier, nous avons énuméré différentes listes :

- Nationalités: 354 nationalités et gentilés[49] ;
- Professions: 296 métiers et fonctions ;
- Politique: 65 fonctions politiques ;
- Militaire: 52 titres militaires ;
- Religion: 24 titres religieux ;
- Sports: 23 sports.

Ces mots déclencheurs sont soigneusement étiquetés lors de la phase d'analyse morphologique. Par exemple, les noms de profession ont l'étiquette <N+Métier>. Par la suite, ils sont utilisés lors de la description des règles syntactico-sémantiques de reconnaissance.

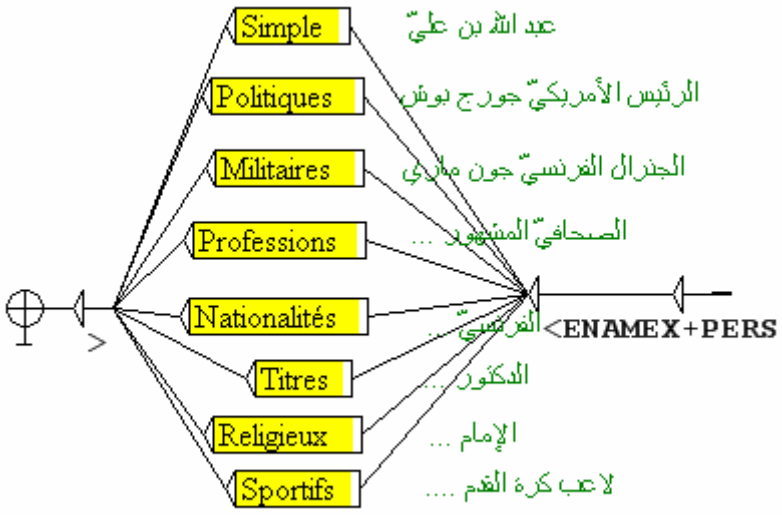


Figure 69 : Grammaire locale de reconnaissance des noms de personnes : Graphe principal (ENAMEX+PERS)

Comparée aux autres entités nommées décrites ci-dessous, notons que la reconnaissance des noms de personnes (<ENAMEX+PERS>) a nécessité le plus grand nombre de règles à écrire. Ceci est principalement dû aux nombreuses possibilités de combinaisons entre preuves internes et externes.

[49] Un gentilé est une dénomination des habitants d'un lieu.

Les règles écrites décrivent aussi bien les contextes potentiels de droite que de gauche. Ceux-ci sont, pour la plupart :

- les civilités : nous citons par exemple : السيد (*al-sayyid* – Mr. / Monsieur), الدكتور (*al-doctuwr* – Dr. / Docteur), etc. ;
- les titres de toutes sortes tels que :
  - ✓ les titres politiques : رئيس (*raýiys* – Président), وزير (*waziyr* – Ministre), etc. ;
  - ✓ les titres militaires : عقيد (*'aqiyd* – Colonel), ملازم (*mulaāzim* – Lieutenant), etc. ;
  - ✓ les titres religieux : شيخ (*šayẖ* – Seigneur), كاردينال (*kaārdiynaāl* – Cardinal), etc. ;

  Ces titres peuvent être reconnus de trois façons différentes : par consultation directe des listes recensées dans le dictionnaire électronique, à l'aide de grammaires morphologiques de tokenisation ou encore à l'aide de grammaires locales de reconnaissance des noms de métier composés tels que رئيس الوزراء (*raýiys al-wuzaraā'* – Chef du gouvernement) ou وزير التربية والتعليم (*waziyr al-tarbiyat wal-ta'liym* – Ministre de l'éducation et de l'enseignement). Par exemple, le sous-graphe « :FonctionsPolitiques » de la Figure 71 décrit tous les titres politiques. Quant à la Figure 72, elle décrit tous les titres "Ministres" pris en compte dans notre étude.
- les noms de professions tels que : مهندس (*muhandis* – ingénieur), بائع (*baāýi'* – vendeur), etc. ;
- le dictionnaire des toponymes pour le repérage des adjectifs de nationalité dans des expressions telles que الرئيس الجزائريّ عبد العزيز بوتفليقة (*al-raýiys al-ğazaāýiriy 'abd al-'aziyz buwtafliqat* – le président algérien Abdelaziz Bouteflika), المستشارة الألمانية أنجيلا مركال (*al-mutašaārat al-álmaāniyat ánğiylaā mirkaāl* - la chancelière allemande Angela Merkel). En l'absence d'indices supplémentaires suffisants, une nationalité isolée ne peut pas être utilisée pour catégoriser un nom propre : elle peut indiquer un nom de personne tel est le cas pour الفرنسي زيدان (*al-firansiyy ziydaān* - le français Zidane)ou un nom de société comme par exemple الفرنسي رينو (*al-firansiyy riynuw* – le français Renault).

Tous ces contextes permettent la catégorisation du nom de personne. En l'occurrence, la grammaire de reconnaissance des noms de personnes politiques est accompagnée de l'ajout de l'annotation « +POLIT », les séquences retenues sont, ainsi, annotées de la forme :

`</ENAMEX>فيون فرانسوا الأوّل الوزير <ENAMEX+PERS+POLIT> (al-wazîr al-áwwal fraāniswaā fiyuwn – le premier ministre François Fillon)`

et de la même façon, nous avons l'annotation :

`</ENAMEX> فرانسوا فيون؛ رئيس الوزراء الفرنسي <ENAMEX+PERS+POLIT> (fraāniswaā fiyuwn, raýiys al-wuzaraā' al-firansiyy – François Fillon, le chef du gouvernement français)`

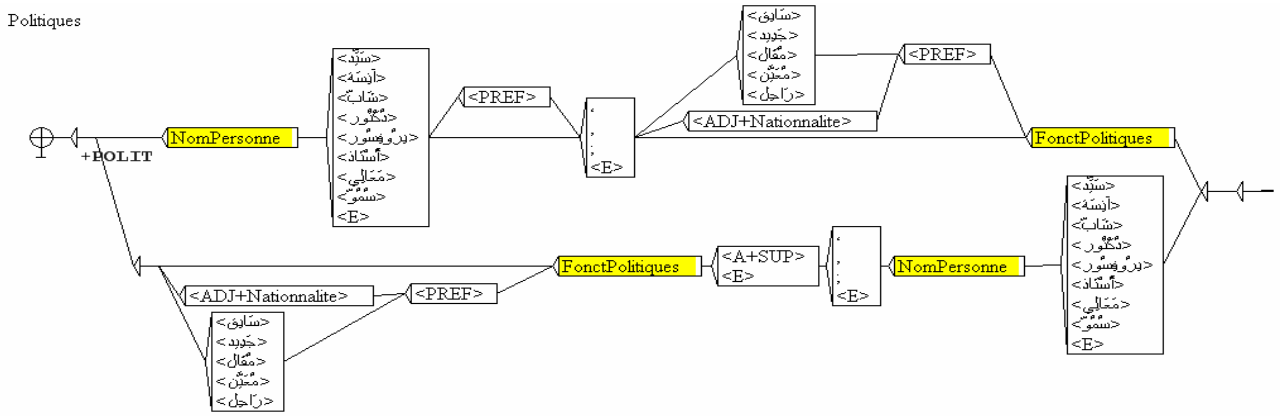


Figure 70 : Grammaire locale de reconnaissance des noms de personnes : Reconnaissance des noms « Politiques » (ENAMEX+PERS+POLIT)

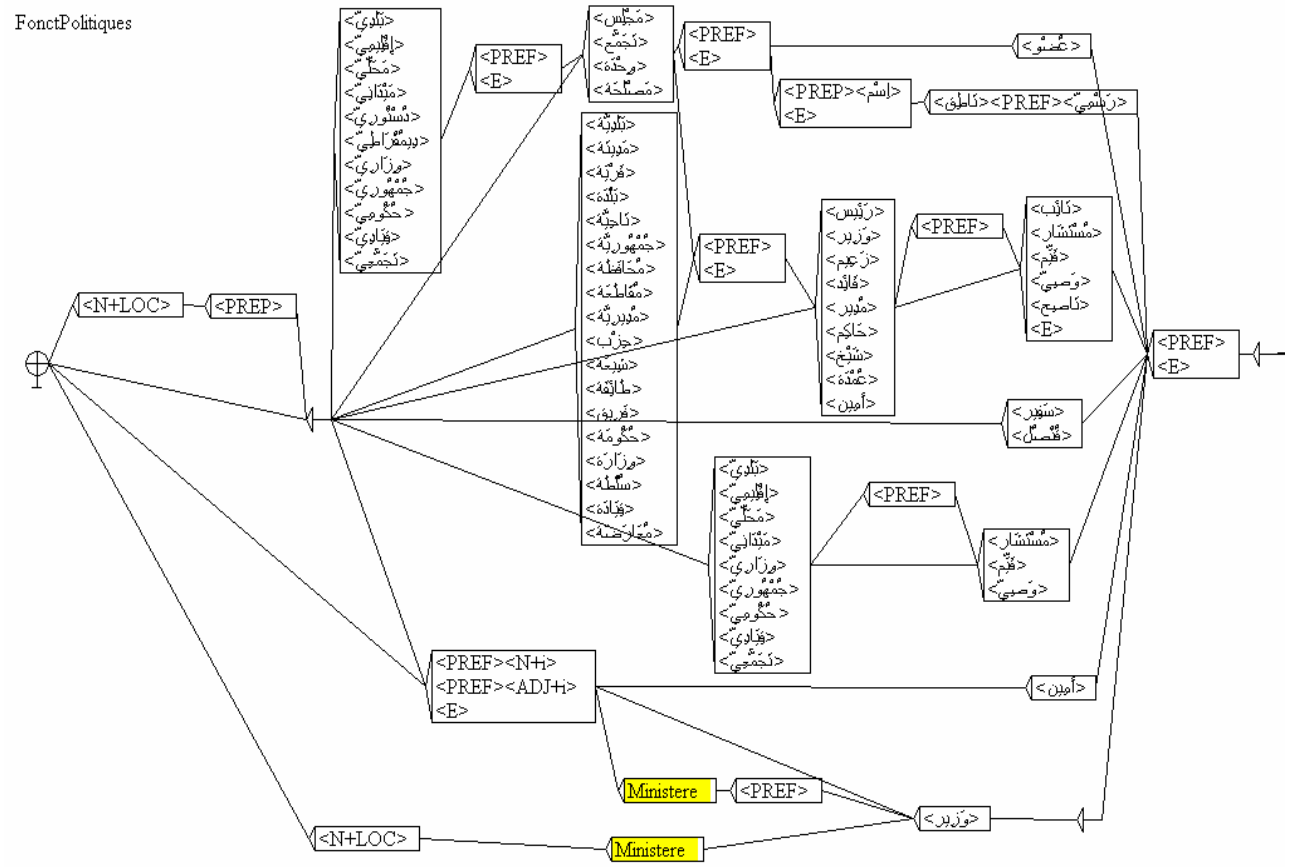


Figure 71 : Grammaire locale de reconnaissance des noms de personnes : Reconnaissance des « Fonctions Politiques »

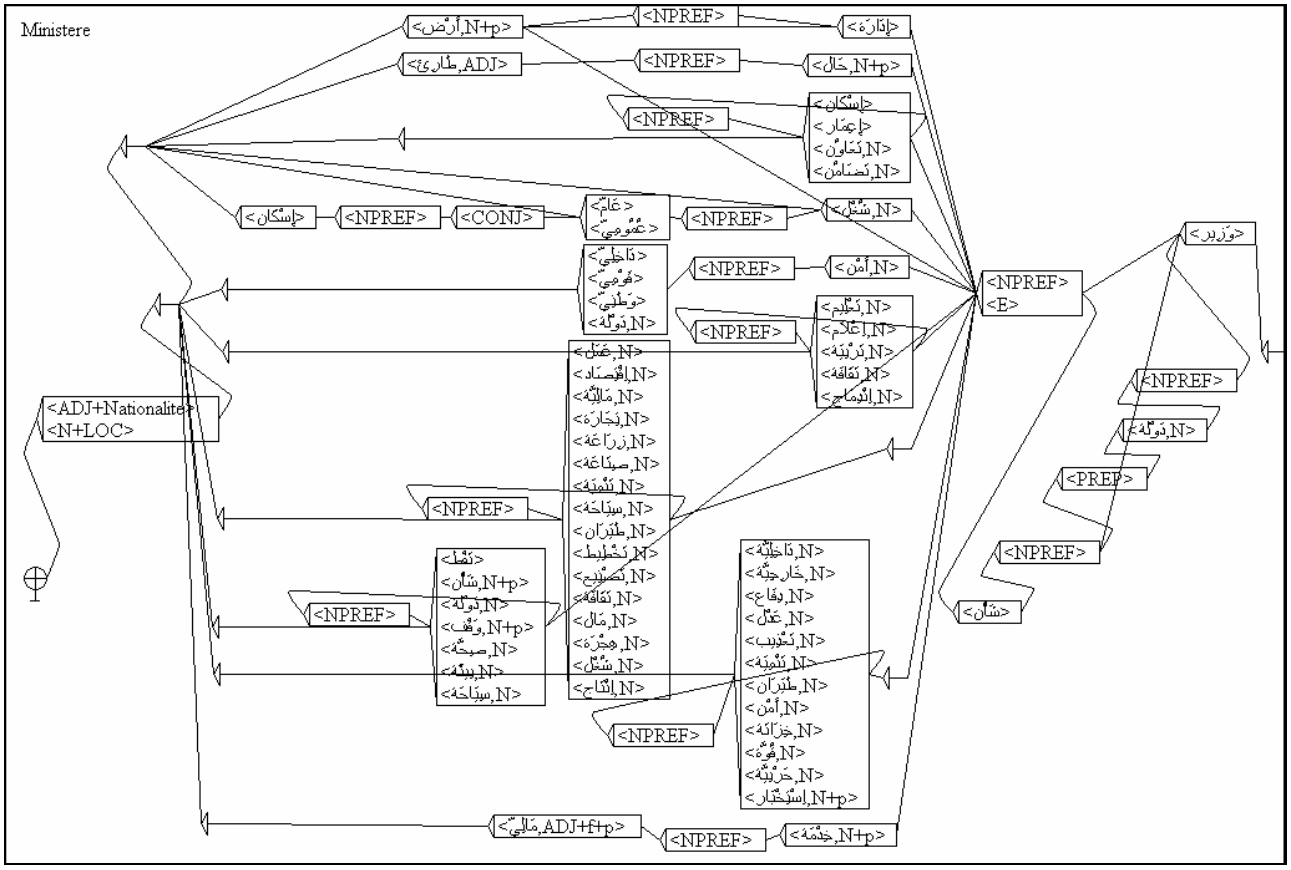


Figure 72 : Grammaire locale de reconnaissance des noms de personnes : Reconnaissance des « Titres Ministériels »

Dans les articles journalistiques du journal « *Le Monde Diplomatique* » qui constituent le corpus que nous utilisons (voir section 5.6.1), les noms de personnes sont accompagnés à hauteur de 89% d'au moins une seule preuve interne ou externe. Ils se répartissent comme suit :

- 70 % des noms de personnes sont accompagnés d'un contexte droit ou gauche, sous forme de preuve interne ou externe, contenant une civilité, un titre, un nom de profession ou un gentilé. S'y ajoute, éventuellement, un prénom (comme preuve interne) identifié grâce au dictionnaire des prénoms ou à la grammaire locale de reconnaissance de prénoms composés. Nous avons distingué trois cas de figures :
  - ✓ *L'entité nommée est accompagnée d'un contexte droit uniquement* : Cette situation représente 60% des cas. Nous pouvons rencontrer par exemple :
    الوزير الأوّل فرانسوا فيون (al-wazîr al-áwwal *fraāniswaā fiyuwn* – le premier ministre François Fillon)
  - ✓ *L'entité nommée est accompagnée d'un contexte gauche uniquement* : Cette situation est beaucoup plus rare que la précédente, elle ne représente que 6% des cas. Nous pouvons rencontrer par exemple :
    فرانسوا فيون؛ رئيس الوزراء الفرنسي (*fraāniswaā fiyuwn*, *raýiys al-wuzaraā' al-firansiyy* – François Fillon, le chef du gouvernement français)
  - ✓ *L'entité nommée est accompagnée d'un contexte droit et un contexte gauche* : Cette situation ne représente que 4% des cas. Nous pouvons rencontrer par exemple :
    العقيد معمر القذافي الزعيم الليبي (*al-'aqiyr mu'ammar al-qaddaāfiy al-za'iym al-liybiy* – Colonel Mouammar Kadhafi, le chef libyen)

- 18 % des noms de personnes n'ont pas de contextes descriptibles mais contiennent une preuve interne apportée par un prénom appartenant à notre dictionnaire ou un prénom composé reconnaissable par une grammaire locale. Nous pouvons rencontrer par exemple :

  فرانسوا فيون (*fraāniswaā fiyuwn* – François Fillon)

  A partir des mesures réalisées sur des articles journalistiques, nous avons été surpris de la rareté de l'usage des noms islamiques introduits par اِبْن (*ibn* - le fils de), بن (*bin* - le fils de), أَبُو (*ábu* - le père de), etc.

- 1 % des noms de personnes sont trouvés par la présence d'un verbe utilisé pour désigner une action mettant en jeu une personne tels que : قَالَ (*qaāla* – dire), شَرَحَ (*šaraḥa* – expliquer), كَتَبَ (*kataba* – écrire), مَاتَ (*maāta* – mourir), تُوُفِّيَ (*tuwuffiya* – mourir), وُلِدَ (*wulida* – venir au monde), etc. Nous citons, par exemple, تُـوُفّـي أبو عمّار في 11 نوفمبر 2004 (*tuwuffiya ábu 'ammaār fiy 11 nuwfambar 2004* – Abou Ammar est décédé, le 11 novembre 2004). Toutefois, nous notons que les verbes comme قَالَ (*qaāla* – dire), شَرَحَ (*šaraḥa* – expliquer) peuvent être employés avec un sujet non humain, un nom d'organisation par exemple : ce type d'indice reste donc difficilement exploitable.
- La dernière proportion des noms de personnes, soit 11%, n'ont aucun contexte. Ces noms de personnes sans contextes sont principalement ceux de personnes déjà citées dans le texte ou ceux de personnes très connues pour lesquels l'auteur du texte estime qu'il n'est pas nécessaire de préciser ni le prénom, ni le titre, ni la profession tel est le cas pour بيكاسو (*biykaāsuw* – Picasso) ou موزار (*muwzaār* - Mozart). Pour remédier à un tel problème, nous envisageons la création d'un dictionnaire de célébrités.

6.7.2 Identification des noms de lieux

Pour les noms de lieux, nous reprenons la classification faite dans [Piton et Maurel, 2004]. Cette classification considère comme nom de lieu ou toponyme : les villes, pays, fleuves, montagnes, etc. De la même façon que pour les noms de personnes, nous avons commencé par l'élaboration d'un dictionnaire de preuves internes incluant plus de 5 000 toponymes ayant plus que 100 000 habitants ou parmi les plus connus dans le monde, y compris :

- Les pays : فِرَنْسَا,N+LOC+Pays
  firansaā,N+LOC+Pays
  France,N+LOC+Pays
- Les villes : وَهْرَانْ,N+LOC+Ville+PAYS=الْجَزَائِر
  wahraān,N+LOC+Ville+PAYS=*al-ğazaāýir*
  Wahrân, N+LOC+Ville+PAYS=Algérie
- Les îles : سَانْتُورَانْ,N+LOC+Ile+PAYS=الْيُونَان
  Saāntuwraān,N+LOC+Ile+PAYS=*al-yuwnaān*
  Santorin, N+LOC+Ile+PAYS=Grèce
- Les états: أُونْتَارِيُو,N+LOC+Etat+PAYS=كَنَدَا
  ùuwntaāriyuw,N+LOC+Etat+PAYS=*kanadaā*
  Ontario,N+LOC+Etat+PAYS=Canada
- Les villes d'un état : هَامِلْتُونْ,N+LOC+Ville+ETAT=أُونْتَارِيُو+PAYS=كَنَدَا
  haāmiltuwn,N+LOC+Ville+ETAT= *ùuwntaāriyuw*+PAYS= *kanadaā*
  Hamilton,N+LOC+Ville+ETAT= Ontario+PAYS= Canada

- Les océans : المُحِيط الأطْلسيّ,N+LOC+Océan
  al-muḥiyṭ al-ȧṭlasiyy,N+LOC+Océan
  Océan Atlantique,N+LOC+Océan
- Les mers : البَحْر الأبْيَض المُتَوَسِّط,N+LOC+Mer
  el batr eláabyaḍ elmutawassiṭ, N+LOC+Mer
  Mer Méditerranée,N+LOC+Mer
- Les fleuves : النِّيل,N+LOC+Fleuve
  al-niyl,N+LOC+Fleuve
  Le Nil,N+LOC+Fleuve
- Les montagnes : هِيمَالايَا,N+LOC+Montagne
  hiymaālaāyaā,N+LOC+Montagne
  Himalaya,N+LOC+Montagne

Outre cette liste de toponymes, nous avons fait usage de la liste de gentilés préparée pour la reconnaissance des noms de personnes. Ensuite, nous avons recensé une liste de 48 mots déclencheurs comme مَدِينَة (*madiynat* – ville), نَهْر (*nahr* – fleuve), جَبَل (*ğabal* – mont), جَزِيرَة (*ğaziyrat* – île), دَوْلَة (*dawlat* - pays), etc. Ces marqueurs lexicaux sont utilisés pour la description des règles de reconnaissance au sein des grammaires locales. Les règles écrites manuellement permettent de distinguer 6 sous-catégories de noms de lieux (voir Figure 73) :

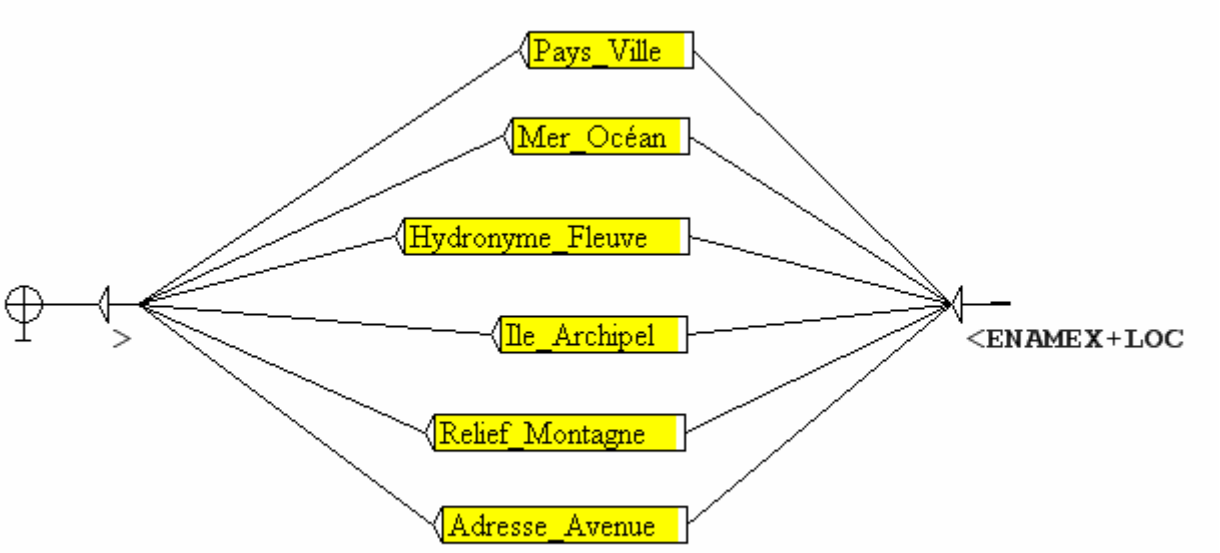


Figure 73 : Grammaire locale de reconnaissance des noms de lieux : Graphe principal (ENAMEX+LOC)

A l'intérieur de ces graphes, nous décrivons différentes règles d'écriture des noms de lieu :

- *Les noms de lieux avec preuve interne uniquement* : tels que : فرَنْسَا (*firansaā* - France), بَاريس (*baāriys* - Paris), تونِس (*tuwnis* – Tunisie), etc.
- *Les noms de lieux avec preuve externe* : tels que : مدينة ليون الفرنسية (*madiynat liyuwn al-firansiyyat* – la ville française de Lyon), جمهورية الكونغو الديمقراطية (*ğumhuwriyyat al-kuwnğuw al-diymuqraātiyyat* – République Démocratique du Congo), etc.
- *Les noms de lieux accompagnés d'un point cardinal* : tels que : جنوب شرق آسيا (*ğanuwb šarq âsiyaā* – le Sud-Est de l'Asie), جنوب الصحراء الجزائرية (*ğanuwb al- ṣaḥraā' al-ğazaāýiriyyat* – le sud du Sahara algérien), etc.

- *Les noms de lieux accompagnés de noms de personnes* : tels que : شارع الزعيم ياسر عرفات (*šaāri' al-za'iym yaāssir 'arafaāt* - Avenue du leader Yasser Arafat), ساحة الجنرال دي غول (*saāḥat al-ğiniraāl diy* ġuwl – Place du Général De Gaulle), etc.
- *Les noms de lieux accompagnés de dates* : tels que : شارع 7 نوفمبر 1987 (*šaāri' 7 nuwfambar 1987* - Avenue du 7 novembre 1987), ساحة 8 مايو 1945 (*saāḥat 8 maāyuw 1945* – Place du 8 mai 1945), etc.

Dans les deux derniers cas, nous réutilisons les expressions de noms de personnes et de dates reconnaissables par l'intermédiaire des annotations <ENAMEX+PERS> décrites dans la section 6.7.1 ou <TIMEX> présentées dans la section 6.6.

Notre étude a révélé que la proportion de noms de lieux ayant une preuve externe ne dépasse pas les 3 %. Le nombre de règles de reconnaissance décrites est assez limité. En effet, les noms de localisations sont identifiés principalement à l'aide de preuves internes identifiées lors de l'étape d'analyse morphologique à l'aide d'une consultation des listes de toponymes recensées.

6.7.3 Identification des noms d'organisations

A l'instar de la procédure de reconnaissance des deux entités nommées précédentes (les noms de personnes et les noms de lieux), l'identification des noms d'organisations a commencé par l'élaboration d'un dictionnaire contenant environ 853 noms d'organisations telles que يُونِيسْكُو (*yuwniyskuw*- Unesco) ou "مُنَظَّمَة الْأُمَم الْمُتَّحِدَة" (*munaẓẓamat al- ùmam al-muttaḥidat* – Organisation des Nations Unies) reconnaissables par le bais de l'étiquette <N+Org>. Nous notons que la majorité des entrées recensées sont composées.
Ensuite, nous avons recensé une liste de 26 mots déclencheurs comme مؤسسة (*muẃassasssat* – société), شركة (*šarikat* - compagnie), جمعية (*ğam'iyyat* – association), etc. Ces marqueurs lexicaux sont utilisés pour la description des règles de reconnaissance au sein des grammaires locales. La grammaire globale de reconnaissance des noms d'organisations est constituée d'une dizaine de sous-graphes, parmi lesquels nous citons :

- *Les noms d'organisations avec une preuve externe simple* tels que :

  شركة ألكاتال (*šarikat álkaātaāl* – la compagnie Alcatel),

  مجموعة فيفندي (*mağmuw'at fiyfindiy* – Groupe Vivendi), etc.

- *Les noms d'établissements institutionnels* (écoles, universités, instituts, facultés, etc.) tels que (voir la Figure 74) :

  المدرسة العليا للتجارة (*al-madrasat al-'ulyaā lil-tiğaārat* – l'Ecole Supérieure de Commerce) ;

  كلية العلوم الإنسانية والاجتماعية بتونس (*kulliyyat al-'aluwm al-insaāniyyat wa-al-iğtimaā'iyyat bi-tuwnis* – La Faculté des Sciences Humaines et Sociales de Tunis) ;

  المعهد التحضيري للدراسات الهندسية (*al-ma'had al-taḥḍiyriy lil-diraāsaāt al-handasiyyat* – l'Institut Préparatoire aux études d'ingénieurs),

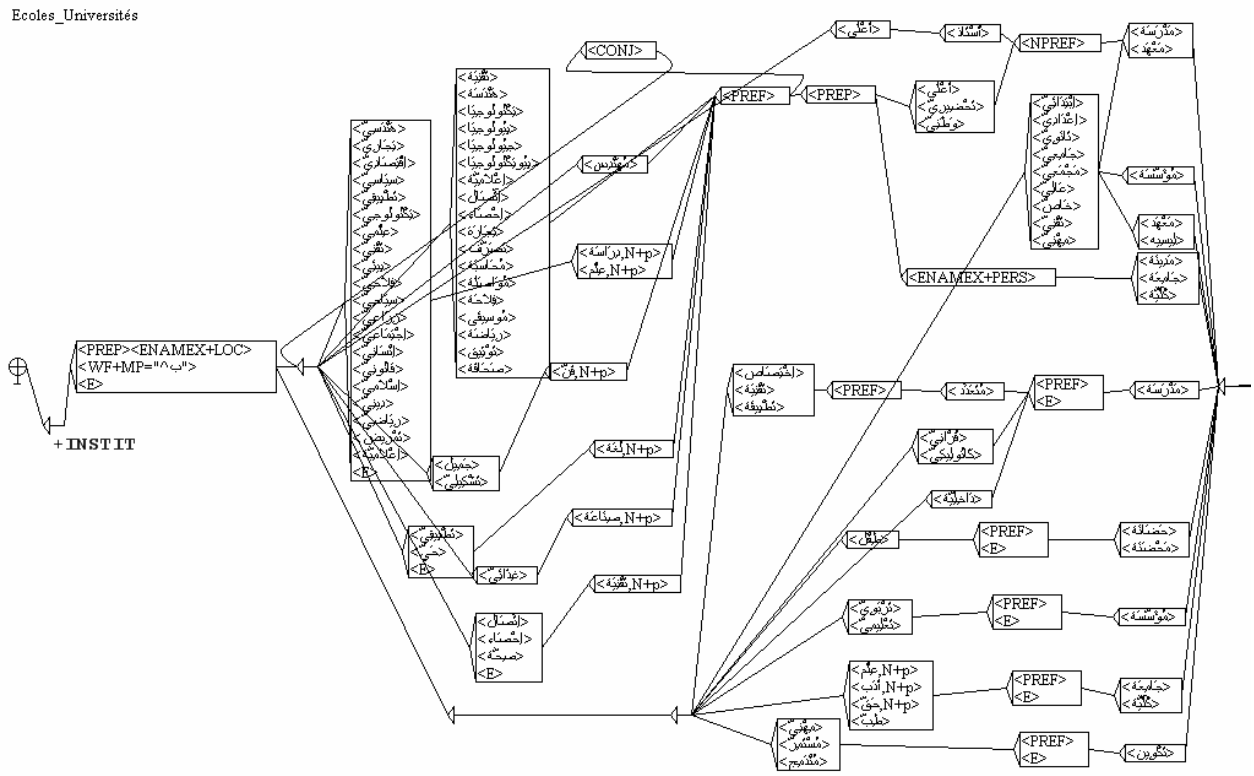


Figure 74 : Grammaire locale de reconnaissance des noms d'organisations : Reconnaissance des « Etablissements Institutionnels » (ENAMEX+ORG+INSTIT)

- *Les noms de ministères et d'organisations internationales* tels que (voir la Figure 75):

  المنظمة العربية للتربية والثقافة والعلوم (ألكسو) (*al-munaẓẓamat al-ʿarabiyyat lil-taribiyat wa-al-ṯaqaāfat wa-al-ʿuluwm – áleksuw* – l'Organisation Arabe pour l'Education, la Culture et les Sciences - ALECSO) ;

  الجمعية الأوروبية للتعاون الاقتصاديّ (*al-ğamʿiyyat al-ùuruwbiyyat lil-taʿaāwun al-iqtiṣaādiyy* – l'Association Européenne pour la Coopération Economique) ;

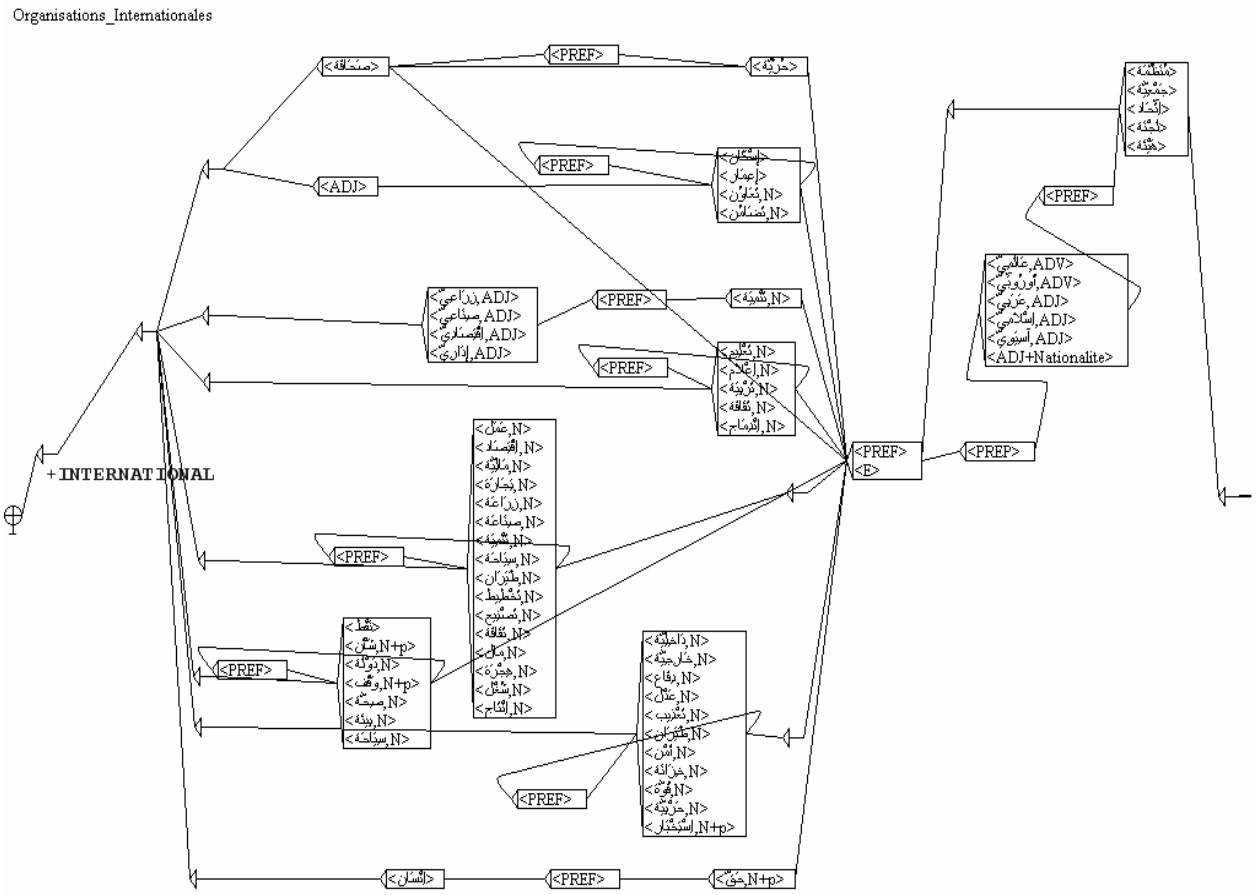


Figure 75 : Grammaire locale de reconnaissance des noms d'organisations : Reconnaissance des associations et organisations internationales (ENAMEX+ORG+INTERNATIONAL)

- *Les noms d'organisations accompagnés d'un nom de personne* tels que :

  مكتبة فرانسوا ميتيران (*maktabat fraānswaā miytiyraān* - Bibliothèque François Mitterand)

  جامعة الإمام محمد بن سعود الإسلامية (*ğaāmi'at al-imaām muḥammad bin sa'uwd al-islaāmiyyat* – l'université Islamique du Imam Mohammed Ben Saoud)

  مدينة الملك عبد العزيز للعلوم والتقنية بالرياض (*madiynat al-malik 'abd al-'aziyz lil-'uluwm wa-al-taqniyat bil-riyaāḍ* - Cité du Roi Abed Al-Aziz pour les Sciences et Techniques au Riyad)

  Nous signalons que, NooJ fournit la possibilité d'assigner des priorités de passage aux ressources linguistiques. Ce système de priorités nous permet d'utiliser les étiquettes précédemment attribuées aux noms de personnes et aux noms de lieux. En effet, grâce à l'attribution d'une plus haute priorité à la grammaire de reconnaissance des noms de personne, nous avons pu inclure dans notre grammaire de reconnaissance des noms d'organisations le symbole grammatical <ENAMEX+PERS> pour désigner n'importe quel nom de personne déjà annoté (voir section 6.7.1).

- *Les noms d'organisations accompagnés d'un nom de lieu* tels que :

  جامعة مدينة لندن (*ğaāmi'at madiynat landan* – London City University)

  جامعة باريس (*ğaāmi'at baāriys* - Université de Paris)

  A l'instar de l'utilisation du symbole grammatical <ENAMEX+PERS>, nous avons utilisé le symbole <ENAMEX+LOC> pour pouvoir réutiliser tous les noms de lieux préalablement identifiés (voir section 6.7.2).

- *Les noms d'organisations accompagnés d'un sigle* tels que :

  مكاتب الآف.بي.آي (*makaātib al- âf.biy.ây* – les bureaux du F.B.I : Federal Bureau of Investigation)
  مركز سي.آن.آر.آس (*markaz al-siy.ân.âr.âss* – le C.N.R.S : Centre National de la Recherche Scientifique)

  Nous notons, que pour ce dernier cas, en l'absence de toute règle d'écriture ou de transcription de sigles et abréviations, nous avons construit une grammaire locale pour la reconnaissance de ceux-ci lorsqu'ils proviennent d'un sigle à 3 lettres ou plus séparées par des points comme le montre la Figure ci-dessous.

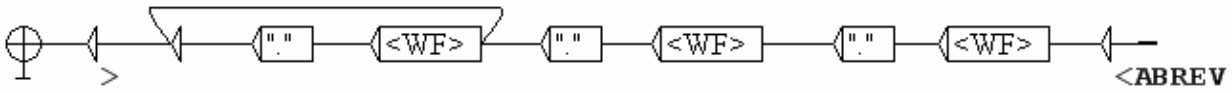


Figure 76 : Grammaire locale de reconnaissance des sigles / abréviations (ABREV)

Les mesures menées sur nos textes journalistiques ont montré qu'environ 82% des noms d'organisations sont accompagnés d'une preuve interne ou externe. Ces preuves peuvent inclure des mots déclencheurs tels que شركة (*šarikat* - compagnie) ou جمعية (*ğam'iyyat* – association), des syntagmes nominaux descriptifs tels que كلية العلوم الإنسانية والاجتماعية بتونس (*kulliyyat al-'aluwm al-insaāniyyat wa-al-iğtimaā'iyyat bi-tuwnis* – La Faculté des Sciences Humaines et Sociales de Tunis) ou المنظمة العربية للتربية والثقافة والعلوم (ألكسو) (*al-munaẓẓamat al-'arabiyyat lil-taribiyat wa-al-ṯaqaāfat wa-al-'uluwm – áleksuw* – l'Organisation Arabe pour l'Education, la Culture et les Sciences - ALECSO), d'autres types de syntagmes tels que les noms propres (y compris les noms de personnes, les noms de lieux) et leurs combinaisons potentielles.

En addition, les noms d'organisations étrangers ont nécessité un traitement particulier. A leur extrémité gauche, ces noms d'organisations ont souvent un emprunt comme preuve interne du type "جورنال" (*ğuwrnaāl* - Journal), "يونايتد" (*yuwnaāytid* - United), etc. Parmi les noms recensés dans notre corpus d'étude, nous pouvons citer :

وول ستريت جورنال (*wuwl striyt juwrnaāl* - Wall Street Journal)
نادي مانشستر يونايتد (*naādiy maānšistir yuwnaāytid* - Manchester United Football Club)

6.8 Evaluation du système de reconnaissance des entités nommées

Traditionnellement, l'évaluation de tout système d'extraction d'information repose sur le calcul d'un ensemble de métriques. Ces calculs permettent d'évaluer la proportion des erreurs affichées par le système par rapport au résultat idéal. Les métriques habituellement utilisées sont :

- **Le rappel (R):** est une évaluation de la couverture du système. Il mesure la quantité de réponses pertinentes d'un système par rapport au nombre de réponses idéales :

$$R = \frac{nombre\ d'entités\ correctes\ détectées}{nombre\ d'entités\ détectées}$$

- **La précision (P):** est une évaluation du bruit du système. Elle mesure la proportion de réponses pertinentes du système parmi l'ensemble des réponses qu'il a fourni :

$$P = \frac{nombre\ d'entités\ correctes\ détectées}{nombre\ d'entités\ manuellement\ identifiées}$$

- **La F-Mesure (F):** est une métrique qui permet de combiner en une seule valeur les mesures de précision et de rappel de manière à pénaliser les trop grandes inégalités entre ces deux mesures. Elle favorise les systèmes dont les deux valeurs sont homogènes :

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

Toutefois, compte tenu des problèmes de délimitation rencontrés, nous avons dû redéfinir les paramètres d'évaluation pour tenir compte des réponses partiellement correctes [Cunningham, 2003]. Ces paramètres d'évaluation deviennent :

$$\textbf{Le rappel :}\ R' = \frac{nombre\ d'entités\ (correctes + partiellement\ correctes)\ détectées}{nombre\ d'entités\ détectées}$$

$$\textbf{La précision :}\ P' = \frac{nombre\ d'entités\ (correctes + partiellement\ correctes)\ détectées}{nombre\ d'entités\ manuellement\ identifiées}$$

$$\textbf{La F-Mesure :}\ F' = \frac{2 \cdot P' \cdot R'}{P' + R'}$$

Une évaluation effectuée sur une partie de notre corpus du journal « Le Monde Diplomatique », décrit dans la section 5.6.1, donne les résultats suivants :

		Précision : P'	Rappel : R'	F-mesure : F'
TIMEX		97%	95%	96%
NUMEX		97%	94%	95,5%
ENAMEX	**Noms de personnes**	92%	79%	85%
	Noms d'organisations	90%	78%	84%
	Noms de lieux	82%	71%	76%

Tableau 21 : Evaluation du système de reconnaissance des entités nommées

A l'issue de cette expérimentation, nous reprenons les résultats de l'évaluation de la couverture lexicale présentés dans le Tableau 20. Nous y ajoutons une troisième étape d'analyse, celle qui concerne la reconnaissance automatique des entités nommées. Dans le tableau ci-dessous, nous affichons les pourcentages de la couverture lexicale à l'issue de chacune des étapes suivantes :

- **1ère étape :** tokenisation et application des dictionnaires électroniques ;
- **2ème étape :** ajout des grammaires de correction des erreurs les plus fréquentes ;
- **3ème étape :** application des grammaires locales de reconnaissance des entités nommées.

	Nombre formes reconnues		**Nombre de formes non reconnues**	
	Nombre	**%**	**Nombre**	**%**
1ère étape	107 914	86,12 %	17 392	13,88 %
2ème étape	115 946	92,53 %	9 360	7,47 %
3ème étape	119 116	95,1 %	6 190	4,9 %

Tableau 22 : Evaluation de la couverture lexicale des ressources linguistiques (2)[50]

Bien que spectaculaires, les résultats affichés ne sont pas parfaits. Ceci est principalement dû aux problèmes liés à la procédure de reconnaissance des entités nommées :

- La prise en compte des variantes orthographiques des noms propres transcrits en l'absence de conventions pour leurs écritures (notamment pour les noms d'organisations et les noms de lieux). En arabe, la translittération et la transcription des noms propres étrangers n'obéissent pas à des règles d'écritures. En l'occurrence, pour la multinationale informatique américaine « Microsoft », malgré le listage de sa forme transcrite la plus courante dans notre dictionnaire des organisations, en moyenne, elle ne serait reconnue qu'à 75% des cas. En effet, une simple recherche sur Internet en utilisant le moteur de recherche Google, nous a permis de constater que cette même organisation peut s'écrire, au moins, de 8 façons différentes[51] :

  - ✓ « مَايكْرُوسُوفِتْ » (*maāyikrūsūfit* - Microsoft): 2 190 000 occurrences (transcription principale à origine anglo-saxonne);
  - ✓ « مَايكْرُسُوفِتْ » (*maāyikrusūfit* - Microsoft): 25 500 occurrences (transcription secondaire);
  - ✓ « مَايكْرُوسُفِتْ » (*maāyikrūsufit* - Microsoft): 64 900 occurrences (transcription secondaire);
  - ✓ « مَايكْرُسُفِتْ » (*maāyikrusufit* - Microsoft): 2 890 occurrences (transcription secondaire);
  - ✓ « مِيكْرُوسُوفِتْ » (*mīkrūsūfit* - Microsoft): 589 000 occurrences (transcription principale à origine francophone);
  - ✓ « مِيكْرُسُوفِتْ » (*mīkrusūfit* - Microsoft): 6 140 occurrences (transcription secondaire);
  - ✓ « مِيكْرُوسُفِتْ » (*mīkrūsufit* - Microsoft): 15 800 occurrences (transcription secondaire);
  - ✓ « مِيكْرُسُفِتْ » (*mīkrusufit* - Microsoft): 129 occurrences (transcription secondaire);

  La présence de l'une ou l'autre de ces transcriptions pourrait dépendre de l'origine de l'auteur (francophone ou anglo-saxonne). Dans notre dictionnaire, nous n'avons introduit que la forme la plus utilisée, « مَايكْرُوسُوفِتْ ». Afin de réduire l'impact d'un tel problème sur les résultats de l'analyse, nous envisageons de lexicaliser au moins les deux transcriptions principales (celle d'origine francophone et celle d'origine anglo-saxonne). Ainsi, nous passerons de 75% à plus

[50] Les cases grisées correspondent aux valeurs déjà affichées dans le Tableau 20.

[51] Les différentes transcriptions ainsi que leurs fréquences ont été recueilli en utilisant le moteur de recherche Google (septembre 2008)

que 96%. Après avoir constaté que les variations de transcription résident, généralement, au niveau des voyelles longues de la forme transcrite, nous envisageons de développer des grammaires morphologiques permettant la prise en compte des variantes orthographiques.

- Le problème de la délimitation des entités identifiées est introduit, principalement, par le haut degré d'ambiguïté qui découle de l'analyse morpho-lexicale de l'arabe. Nous proposons d'effectuer un traitement syntaxique supplémentaire afin de mieux « comprendre » les structures syntaxiques des phrases et procéder à une désambiguïsation morpho-syntaxique avant de procéder à l'identification des entités nommées.

6.9 Conclusion

Nous avons décrit un système pour la reconnaissance de noms propres, expressions temporelles et numériques par le biais de combinaison d'analyseur morphologique et d'un système de reconnaissance à base de règles utilisant des grammaires locales NooJ. Ce système est utilisé pour identifier et classifier les formes qui sont restées inconnues par la simple consultation de dictionnaires et l'analyse morphologique automatique. Ceci nous a permis l'amélioration des performances de notre système en assurant une couverture lexicale dans plus que 95% des cas. Malgré les problèmes décrits plus haut, la méthode préconisée semble être adéquate et affiche des taux de reconnaissance très encourageants.

D'après les résultats obtenus au fur et à mesure du développement de ce système, nous avons remarqué que les grammaires d'extraction des entités nommées construites respectent la loi de Zipf. En effet, une minorité des règles peut suffire pour s'appliquer à une grande partie des motifs et assurer une couverture moyenne. Toutefois, d'autres règles doivent être ajoutées pour améliorer le rappel.

Chapitre 7

Application : Concordancier arabe en ligne (NooJ4Web)

7.1 Introduction

Les concordanciers ont été toujours considérés parmi les outils les plus simples en ce qui concerne la facilité d'utilisation et les plus puissants en termes de quantité des informations fournies dans le domaine de la linguistique des corpus. Traditionnellement, un concordancier est le programme qui fournit une liste de mots ou de séquences, recherchés dans un texte, entourés par les contextes dans lesquels ils apparaissent.

Les processus de traitement des concordanciers présentent une similitude assez remarquable. Après une première étape de recueil des textes de travail, l'utilisateur soumet ses requêtes pour la recherche d'un mot, d'une séquence, d'une collocation ou d'un concept linguistique. Généralement, la séquence pertinente est affichée au centre d'une fenêtre ; elle est entourée par son contexte composé de l'ensemble de caractères qui la précédent et/ou la suivent.
Une fois la recherche est terminée, les résultats peuvent être triés, sauvegardés ou imprimés avec, en option, la possibilité de limitation du nombre de lignes de concordance. Cliquer sur une ligne de concordance conduit à l'élargissement des contextes affichés. Le contenu de ces contextes peut également être sauvegardé ou imprimé.

7.2 Etat de l'art sur les concordanciers

Dans la littérature, nous trouvons de nombreux systèmes de concordance [Rodriguez, 1999]. Leurs principes de travail sont assez équivalents. Nous donnons une brève description des plus connus. A ce jour, nous ne trouvons pas de concordanciers pour l'arabe proposant des services en ligne. Nous allons, ainsi, répartir notre description en deux ; nous exposons, successivement, les concordanciers qui prennent en compte le traitement de l'arabe et les concordanciers qui proposent des services en ligne.

7.2.1 Concordanciers de l'arabe

Parmi les concordanciers traitant des corpus arabes, nous citons :

- **MonoConc:**
  MonoConc est un concordancier commercialisé, il a été développé par Barlow [Barlow, 1998] et publié par Athelstan. Son utilisation est relativement simple, il établit des listes de concordances en se basant sur la spécification d'un certain nombre de paramètres de recherche. Monoconc offre un ensemble limité d'options de recherche (mots simples, lexèmes, ou des séquences de mots), ainsi que des possibilités limitées d'édition. De plus, il ne traite que des textes précédemment chargés. Toutefois, travailler sur des textes du disque local nécessiterait un temps de chargement à chaque fois.

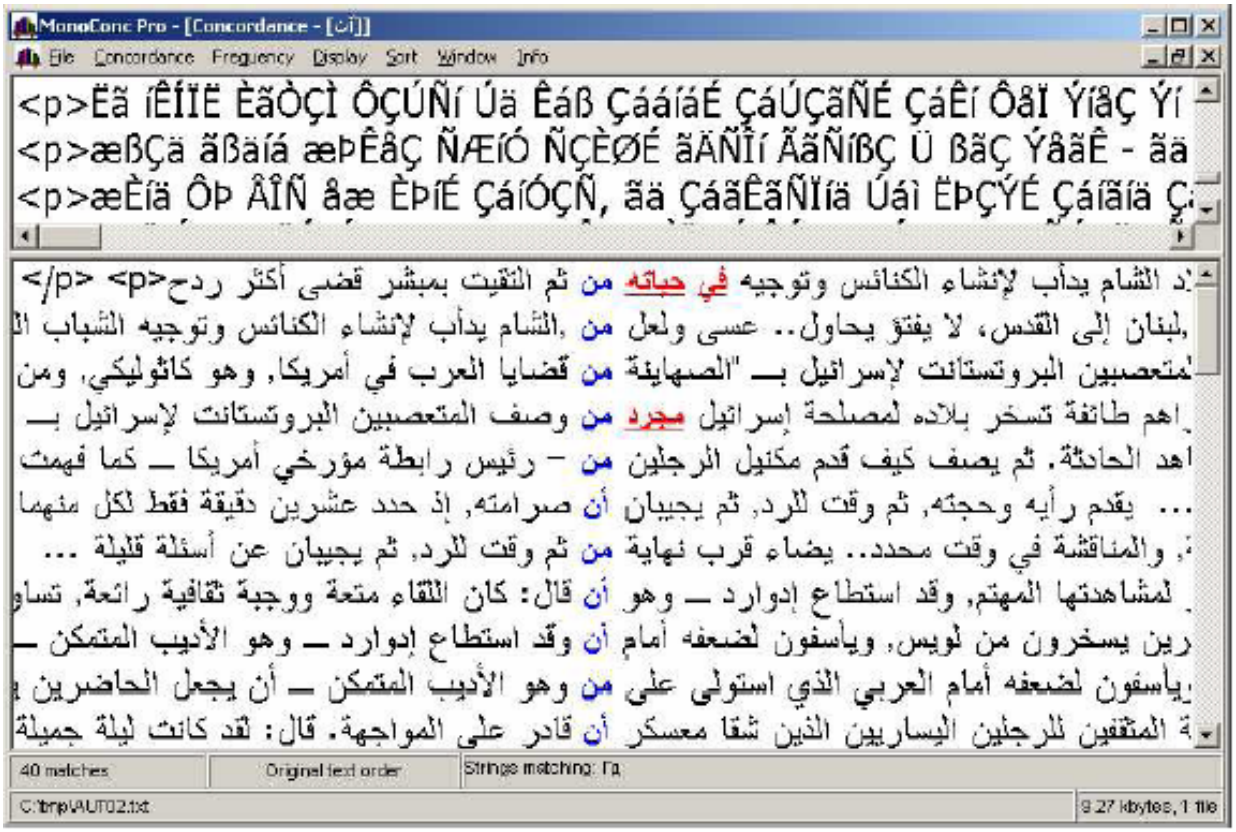


Figure 77 : Table de concordances dans MonoConc

Sur cette capture d'écran, nous montrons les trois insuffisances majeures de MonoConc :

- ✓ Inversion de l'ordre d'affichage des contextes droit et gauche autour de la séquence recherchée ;
- ✓ Affichage erroné des caractères arabes en haut de la fenêtre de concordances. Ceci est dû à l'utilisation du code de page 1256 de Windows au lieu de l'UTF-8 pour coder les textes ;
- ✓ Adjonction de termes indésirables dans la table de concordances. En l'occurrence, sur l'exemple affiché, la recherche de la particule du subjonctif "أن" (*án*) a donné naissance à une reconnaissance de la préposition "من" (*min*). Ce comportement semble être aléatoire car il n'est pas reproductible pour toutes les requêtes.

- **WordSmith :**
  WordSmith a d'abord été publié en 1996 et est toujours développé par le linguiste Mike Scott [Scott, 2008]. WordSmith est actuellement à la version 5 ; il inclut trois outils: liste des mots, liste des mots-clés et un concordancier. Nous notons que WordSmith, dans sa version de démonstration disponible en ligne[52], affiche correctement les caractères Unicode mais il limite le nombre de résultats renvoyés lors de l'exécution de requêtes.
  Toutefois, comme dans MonoConc, il présente le même problème d'inversion de l'ordre d'affichage des contextes droit et gauche autour de la séquence recherchée que dans MonoConc.

[52] La version de démonstration est téléchargeable à l'adresse : http://www.lexically.net/wordsmith/

Figure 78 : Table de concordances dans WordSmith

- **aConcorde :**
  aConCorde n'est ni un produit commercial, ni un projet de recherche. Son développement a été entamé par Andrew Roberts dans le cadre de sa thèse de doctorat et se poursuit toujours [Roberts et al, 2006]. En dépit des fonctionnalités assez basiques qu'il propose, il présente une parfaite prise en charge de l'Unicode et de l'orientation droite-gauche des textes en arabe. Il fournit deux interfaces : une interface anglaise et une interface arabe.

Figure 79 : Table de concordances dans aConcorde

Malgré les avantages de ce concordancier, nous notons qu'il a hérité des insuffisances de l'analyseur morphologique de Buckwalter (voir section 3.3.2) qu'il utilise comme moteur linguistique.

- **AraConc :**
  AraConc est un concordancier de l'arabe développé en extension de l'analyseur AraParse pour fournir les contextes et les fréquences et permettre l'exploration du corpus selon les traits proposés par l'analyse morphologique et selon les informations graphiques qui se trouvent dans le texte [Abbes, 2004]. Le logiciel prend pour entrée un texte ou un ensemble de textes. Il permet :
  - ✓ La construction de listes de fréquences de tokens, de racines ou de tout autre trait de l'analyse morpho-syntaxique. Les listes peuvent être triées par ordre alphabétique ou par ordre de fréquence ;
  - ✓ La construction d'une concordance peut se faire par token, par la racine, par lemme ou par analyse morpho-syntaxique.

Figure 80 : Table de concordances dans AraConc

- **xConcord :**
  xConcord est un programme conçu par M. Boualem et al. pour fournir une grande souplesse dans la recherche effectuée sur des textes multilingues. Son moteur de requête actuel se base sur des expressions rationnelles pour permettre des recherches de surface afin de reconnaître certaines variations morphologiques. Toutefois, les concepteurs travaillent sur l'amélioration du système de concordances dans le but de permettre à l'utilisateur de spécifier une partie du discours [Boualem et al., 1999].

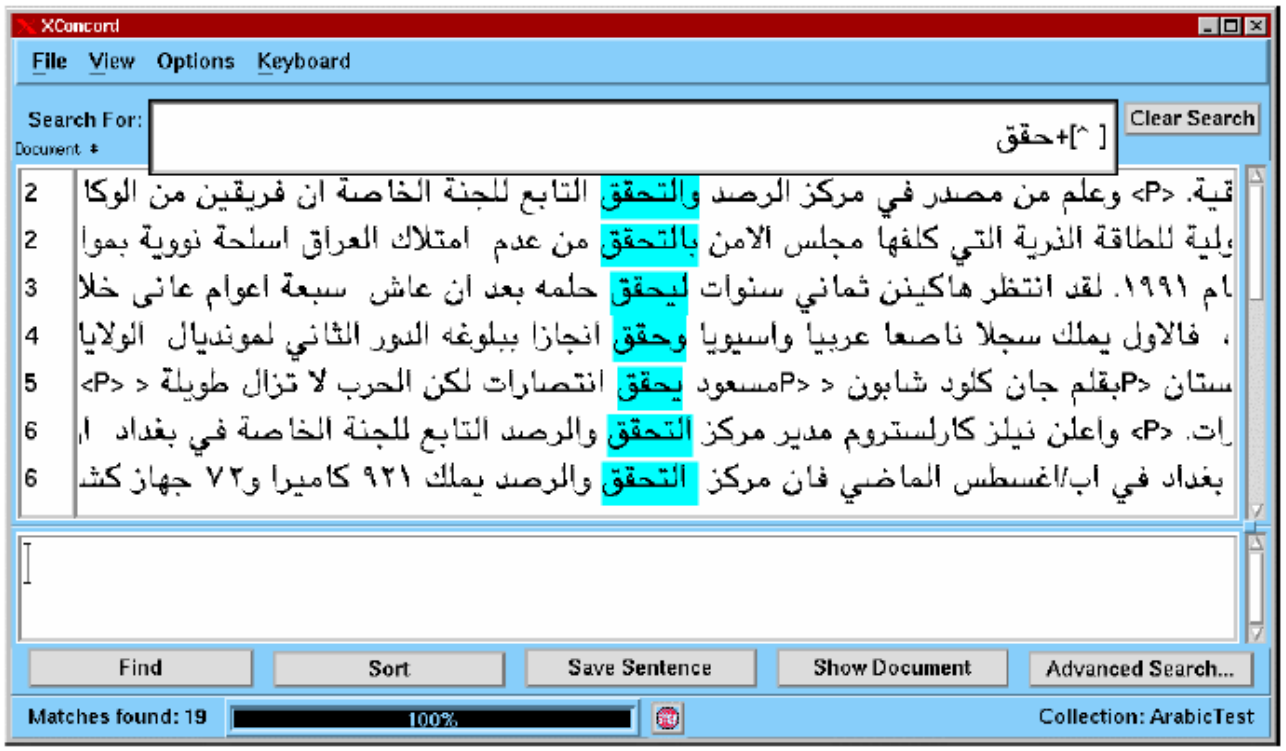


Figure 81 : Table de concordances dans xConcord

L'exemple proposé ci-dessous permet de retrouver tous les mots se terminant par une certaine chaîne de caractères, soit "حقق".

7.2.2 Concordanciers en ligne

Quant aux concordanciers en ligne, nous citons :

- **CobuildDirect – BncWeb :**
  CobuildDirect est un concordancier commercialisé. Développé par l'éditeur Collins, il est considéré comme l'un des concordanciers les plus puissants sur le marché. Il s'agit d'un service en ligne utilisant un large corpus de l'anglais, Collins WordbanksOnline, composé de 56 millions de formes de la langue contemporaine parlée et écrite (40 millions de formes à partir de textes de la Banque de l'anglais – Bank of English, 10 millions de formes à partir de discours transcrits, émissions de radio, livres couvrant différents sujets, etc.). Ces corpus sont étiquetés à l'aide d'outils statistiques pour permettre des recherches complexes combinant des mots de la langue et des étiquettes grammaticales. CobuildDirect fournit diverses informations statistiques telles que le T-Score et la valeur de l'information mutuelle[53]. Il est également possible de sauvegarder toutes les recherches effectuées dans un fichier sur le site de Cobuild. Ensuite, l'utilisateur peut récupérer le fichier enregistré à l'aide d'un client FTP. Toutefois, il reste le concordancier le plus cher.

- **Glossanet :**
  Glossanet est une application qui permet d'analyser l'édition quotidienne en ligne de l'ensemble des journaux sélectionnés par l'utilisateur. Il applique automatiquement la(les) requête(s) de l'utilisateur et procède par l'envoi de courriers électroniques contenant les concordances

[53] Dans la théorie des probabilités et la théorie de l'information, l'information mutuelle de deux variables aléatoires est la valeur mesurant la dépendance statistique de ces variables.

[Fairon, 2001]. Il permet aux utilisateurs de rechercher des mots ou des séquences de mots en se basant sur l'utilisation d'un ensemble de ressources linguistiques.

- **ConcApp:**
  ConcApp est une application développée par Greaves [Greaves, 2003]. Cette application comporte des options d'édition et de recherche très semblables à celles dans MonoConc. ConcApp fournit une analyse de la fréquence d'apparition de formes dans le texte traité. Il supporte l'Unicode ; il permet ainsi le traitement aussi bien du français et de l'anglais que du chinois, japonais, thaïlandais et russe.

- **WebCorp:**
  WebCorp est un concordancier offrant une large gamme d'options de filtrage et de formatage (y compris KWIC - KeyWord In Context). La version actuelle de WebCorp repose sur différents moteurs de recherche tels que Google et AltaVista, auxquels il ajoute la possibilité de raffiner les requêtes. La possibilité de rechercher sur l'ensemble des documents sur le Web engendre de sérieux problèmes concernant le temps de réponse du concordancier.

Bien que naturellement non exhaustif, le panorama des différents concordanciers décrits a permis d'établir les différentes méthodes, approches et fonctionnalités existantes. L'intérêt d'une telle démarche étant de pouvoir positionner nos travaux par rapport à ceux-ci et de pouvoir profiter des avantages que présente chacun des travaux énumérés.

7.3 Architecture générale

Généralement, les concordanciers couvrent un certain nombre de fonctions de base. Les principales différences résident dans le nombre d'options pour l'édition, la recherche, le chargement des textes, ainsi que les apports en termes de vitesse d'exécution.

En l'absence de concordanciers proposant des services en ligne pour le traitement de corpus arabes, nous proposons l'application NooJ4Web (NooJ pour la Toile – NooJ for the Web) dans laquelle nous essayons de regrouper toutes les fonctionnalités de base qui ont été intégrées dans les concordanciers cités ci-dessus. En outre, nous tirons profits des apports de l'utilisation du moteur linguistique de la plate-forme de développement NooJ pour garantir la qualité de la recherche linguistique et la vitesse d'exécution des tâches de traitement.

La Figure 82 montre les deux étapes de traitement de NooJ4Web:

- Etape de pré-traitement ;
- Etape de traitement des requêtes.

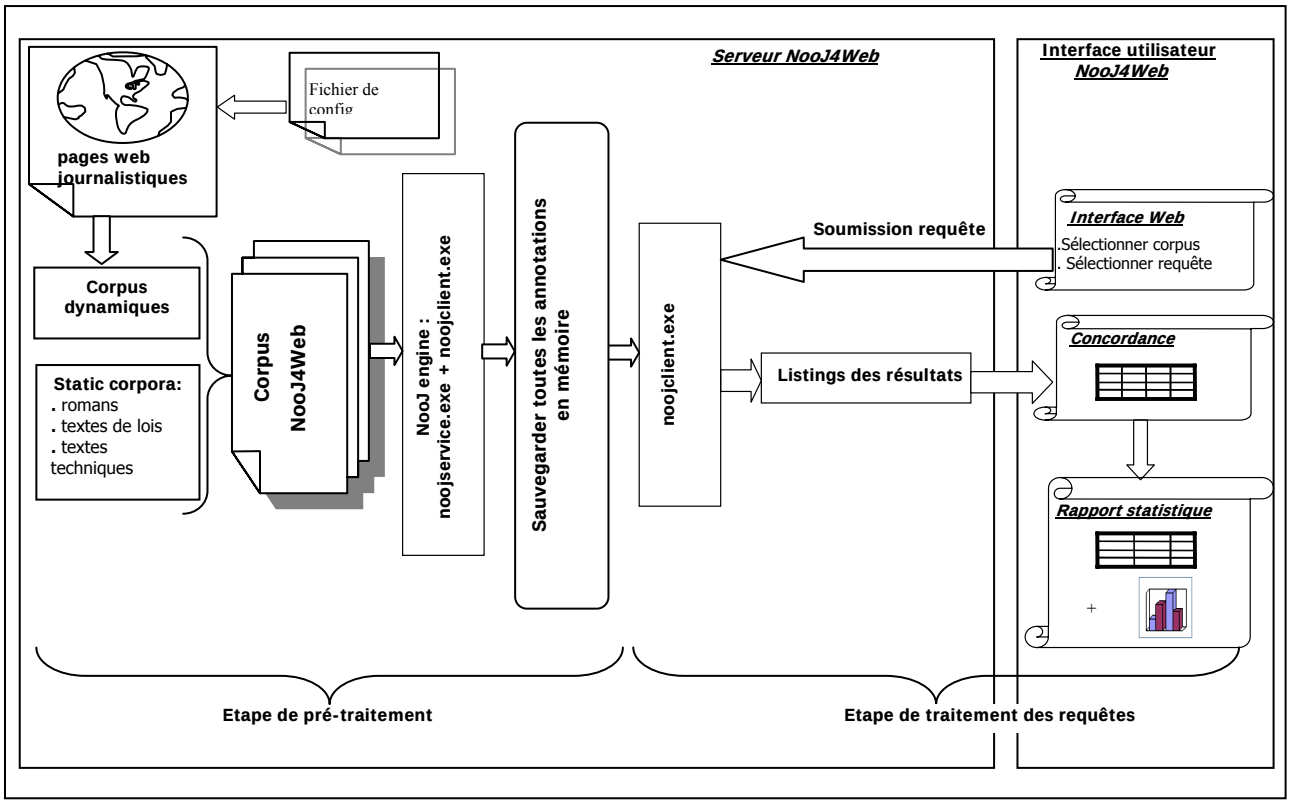


Figure 82 : Architecture générale de NooJ4Web

Dans la section suivante, nous donnons les détails de chacune des phases de traitement et nous décrivons les routines de communication entre les côtés client et serveur de notre application Web.

7.3.1 Etape de pré-traitement

La première phase de traitement dans NooJ4Web se rapporte à la construction et l'annotation de corpus. Elle concerne la collecte de corpus statiques sur Internet et la construction de corpus dynamiques par le biais d'un programme d'aspiration planifiée mené d'un filtre pour l'extraction régulière de textes journalistiques en ligne. Par la suite, ces corpus sont analysés en utilisant la plate-forme linguistique NooJ. NooJ4Web utilise le moteur linguistique de NooJ, construit sur une architecture client/serveur, avec:

- **noojservice.exe :**
  Noojservice.exe est un programme développé sous forme d'un service Windows. Il est lancé automatiquement au démarrage du système d'exploitation Microsoft Windows et s'exécute en arrière-plan tant que le serveur est en marche. Il intègre le moteur linguistique de NooJ. Au cours de cette première phase de pré-traitement, Noojservice.exe est appelé automatiquement pour l'analyse et l'annotation de tous les textes. Ensuite, d'un côté, l'index de toutes les annotations ainsi qu'une liste de requêtes prédéfinies sont stockés dans la mémoire de la machine serveur. De l'autre côté, les résultats de ces requêtes sont enregistrés sur le disque dur. En revanche, au cours de la seconde phase du traitement, Noojservice passe en mode de veille dans l'attente des requêtes clientes. Ces requêtes lui sont communiquées par le biais d'un processus Noojclient auquel seront renvoyées les réponses après consultation des annotations dûment stockées en mémoire.

- **noojclient.exe :**
  Noojclient.exe représente le côté client de notre architecture client /serveur. Pendant la phase de pré-traitement, il est automatiquement lancé pour la mise à jour des corpus dynamiques dans

NooJ4Web. Lors de la phase de traitement des requêtes, il est considéré comme l'interface entre les utilisateurs et Noojservice. Noojclient.exe ordonne les demandes et les communique à Noojservice; Ensuite, après attente de l'exécution de la requête, il reçoit et affiche les réponses aux utilisateurs finaux en utilisant une interface Web.

Une fois tous les index des annotations des corpus stockés dans la mémoire du serveur, notre concordancier est mis en attente jusqu'à la réception des requêtes soumises par les utilisateurs.

7.3.2 Etape de traitement des requêtes

Durant cette deuxième phase du traitement, nous traitons les requêtes soumises par les utilisateurs de NooJ4Web. En effet, après sélection du corpus, les utilisateurs ont la possibilité de choisir une requête parmi un ensemble prédéfini d'exemples pédagogiques, ou de créer leur propre requête selon la syntaxe des expressions rationnelles dans NooJ. Ensuite, la requête de l'utilisateur est transmise au serveur où Noojservice utilisera l'index des annotations stocké en mémoire pour fournir une réponse instantanée. Cette réponse est formatée dans une grille de données sous forme d'une liste de séquences pertinentes. Ces séquences sont entourées par les contextes dans lesquels elles apparaissent. Par la suite, les utilisateurs peuvent construire un rapport statistique des concordances affichées. Ce rapport statistique contient des mesures relatives à la fréquence des séquences pertinentes dans chaque texte. Ces mesures peuvent être utilisées pour construire, dynamiquement, l'histogramme du facteur de dispersion de cette fréquence, dans une fenêtre séparée.

7.3.3 Diagramme des cas d'utilisation

Dans cette section, nous nous basons sur la formulation UML (Unified Modeling Language) pour décrire les possibilités d'interaction entre NooJ4Web et ses utilisateurs à travers un diagramme des cas d'utilisation (use case). Le diagramme décrit dans la Figure 83 représente la séquence des actions potentielles réalisées par notre application :

- Soumettre une requête par le biais de la sélection d'un corpus et de la définition d'une requête (une requête prédéfinie ou une nouvelle requête);
- Afficher une concordance suite à la soumission d'une requête ;
- Demander le rapport statistique de la table de concordances déjà affichée ;
- Demander l'histogramme de la déviation standard (Ecart-type) à partir du rapport statistique.

Chacun de ces services est garni par un ensemble de fonctionnalités afin de trier les résultats affichés, modifier les options d'affichage, exporter les tables et concordances, etc.

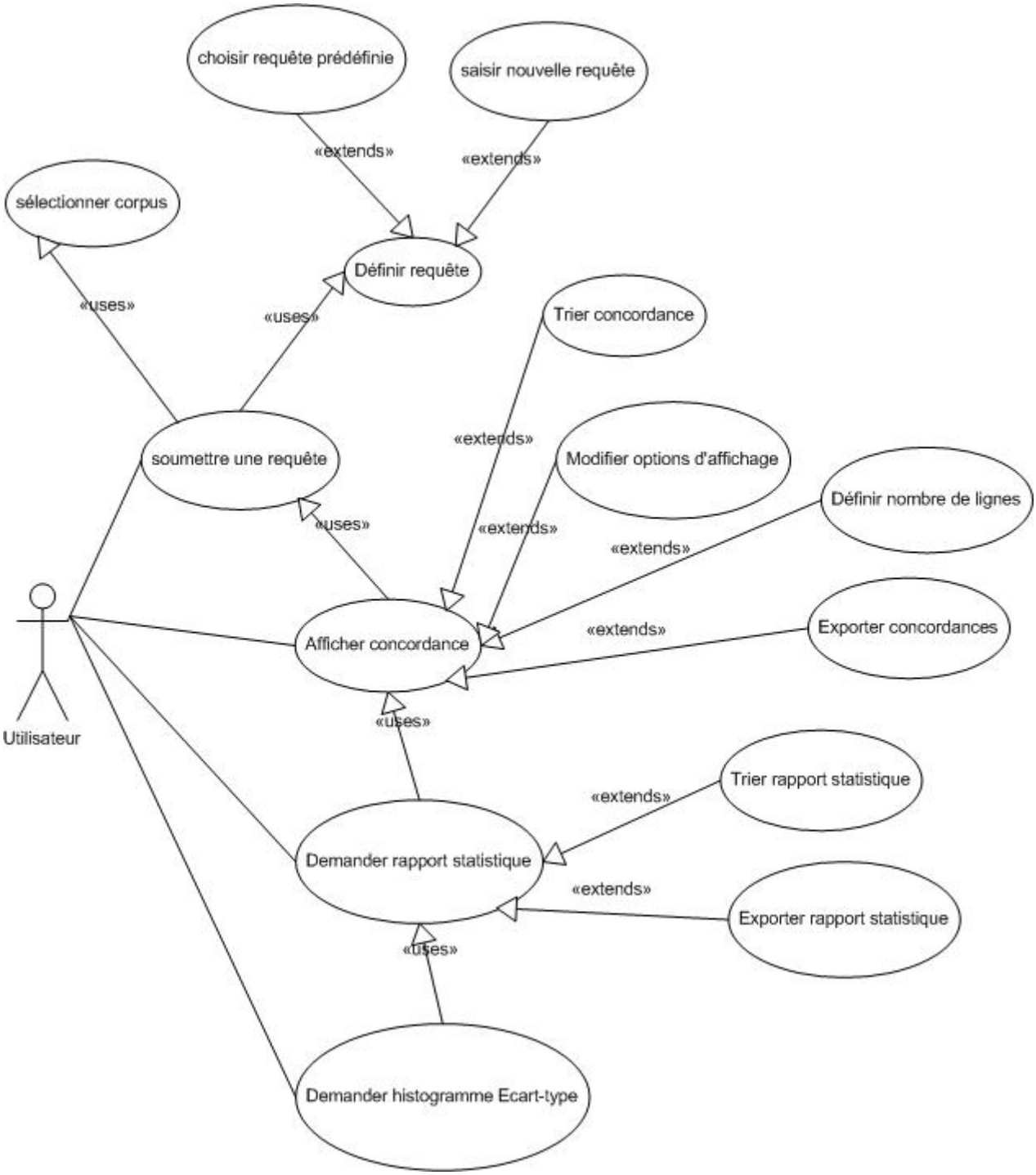


Figure 83 : NooJ4Web : Diagramme des cas d'utilisation

7.3.4 Diagramme de séquence

Dans ce paragraphe, nous regroupons tous les cas d'utilisation décrits plus haut. Nous continuons avec la formulation UML pour donner une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système (NooJ4Web et NooJ engine) et les acteurs (utilisateur ou administrateur).
Dans la Figure 84, nous représentons la séquence d'événements se déroulant durant l'utilisation de notre application.

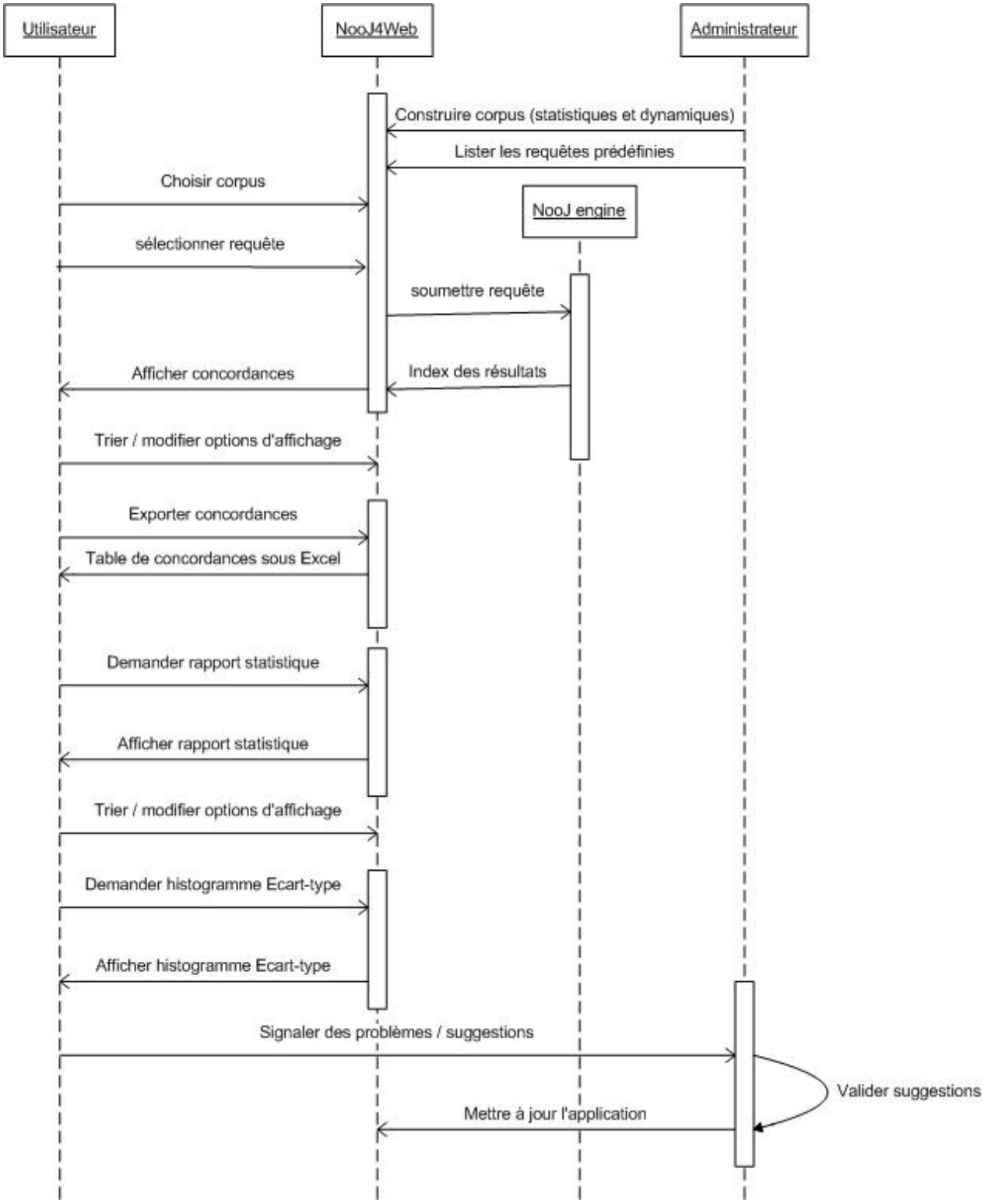


Figure 84 : NooJ4Web : Diagramme de séquence

7.4 Création des corpus

NooJ4Web permet l'utilisation et le traitement de corpus aussi bien statiques que dynamiques.

7.4.1 Corpus statiques

Les corpus statiques font référence à une collection de textes établie selon un certain nombre de critères de sélection en cohérence avec les objectifs didactiques et pédagogiques de notre application. Pour chacune des langues traitées, nous avons recueilli sur Internet un ensemble de textes composé de :

- Un ensemble de romans littéraires écrits par des auteurs renommés dans la littérature de la langue en question. Pour la langue arabe, nous en avons recensé une dizaine. Quant aux langues français et anglaise, nous avons recensé plus qu'une centaine de romans, faisant partie de la littérature du 19ème et 20ème siècle, écrits par les auteurs les plus renommées de leur époque;
- Un ensemble de textes juridiques, y compris des textes de lois, des décrets, des constitutions, des codes juridiques, des actes, etc. ;
- Un ensemble de textes techniques.

Tandis que ces textes sont représentatifs de plusieurs phénomènes linguistiques, ils ne donnent pas accès à des informations actualisées et ne peuvent pas montrer les fluctuations de la langue. Donc, nous avons opté pour l'ajout de corpus dynamiques.

7.4.2 Corpus dynamiques

Contrairement aux corpus statiques, les corpus dynamiques évoluent au cours du temps. Un corpus dynamique est un flux de données textuelles sous forme électronique, tel que les textes journalistiques. Le concept de corpus dynamiques a été introduit par A. Renouf [Renouf, 2002], dans le cadre du projet de recherche AVIATOR dont l'objectif était d'étudier les formes d'évolutions potentielles d'un corpus (nouveaux mots, nouveaux sens, nouvelles combinaisons possibles, évolution des fréquences, etc.). Dans NooJ4Web, les corpus dynamiques sont mis à jour régulièrement à partir des éditions quotidiennes de certains journaux en ligne. Un fichier de configuration est utilisé pour filtrer les URLs et limiter le téléchargement aux articles d'actualité. Étant donné la grande quantité de données stockées sur les sites Web des journaux en ligne, ces derniers évitent les changements structurels profonds. Cette hypothèse est soutenue par la constatation d'aucune modification de configuration tout au long de notre période de test. Le cas échéant ne nécessiterait qu'une simple modification de la ligne de configuration relative au site du journal en question.

Tous les articles téléchargés sont filtrés pour supprimer toutes les balises HTML, les publicités, les images ainsi que d'autres éléments ajoutés automatiquement, et en extraire le texte brut. Ensuite, ces textes sont analysés en utilisant le moteur linguistique de NooJ. Outre les ressources linguistiques habituellement disponibles dans NooJ, tels que des dictionnaires électroniques à large couverture et des grammaires locales, nous utilisons certains dictionnaires filtres afin de résoudre les cas d'ambiguïté les plus fréquents.

7.5 Formulation des requêtes

NooJ4Web permet la recherche de mots, de lemmes aussi bien que leurs formes fléchies ou dérivées, ainsi que des collocations complexes, tels que les noms composés et les expressions figées. Parmi les nouveautés apportées par notre concordancier, nous citons la possibilité de combiner des requêtes linguistiques avec celles de type PERL. Il permet aussi la recherche d'expressions discontinues telle que "prendre [...] en compte" tout en ayant la possibilité d'assigner le nombre et l'ordre des différents constituants dans de telles expressions.

Parmi les requêtes potentielles applicables aux corpus pré-annotés dans NooJ4Web, les utilisateurs ont la possibilité de soumettre des requêtes morphologiques ainsi que des requêtes syntaxiques ou syntagmatiques.

7.5.1 Requêtes morphologiques

Ce premier type de requêtes permet de faire des recherches morphologiques, portant sur la forme des mots, en recherchant des chaînes de caractères entrant dans la composition des mots, c'est-à-dire faire une recherche des mots qui commencent, finissent ou contiennent une chaîne de caractères fournie par l'utilisateur en utilisant quelques fonctions de base de type PERL. En outre, ces requêtes morphologiques permettent d'effectuer des recherches plus complexes en utilisant les annotations préalablement insérées dans le corpus. Ces annotations représentent l'ensemble des informations linguistiques fournies par les dictionnaires électroniques ou les grammaires morphologiques tels que le lemme, la catégorie grammaticale, les informations flexionnelles, les informations distributionnelles (ex. +Hum), etc.

Parmi les requêtes morphologiques, nous pouvons rechercher :

- des formes caractérisées par la donnée de leur racine. Par exemple, <بَدَأ> correspond à toutes les formes fléchies du verbe "بَدَأ" (badaáa – commencer);
- des formes associées à une catégorie morpho-syntaxique. Par exemple, <PREP> désigne toutes les prépositions;
- toutes combinaisons de codes disponibles dans les dictionnaires électroniques de NooJ. Par exemple :
    - <N+Métier+m-s> désigne toutes les formes nominales d'un métier et déclinées au masculin et autre qu'au singulier ;
    - <V+tr+Pr-3> désigne tout verbe transitif conjugué au présent et non à la troisième personne ;

Dans toutes les requêtes, des opérateurs de négation sont utilisables sur deux niveaux différents : une négation globale par le biais du caractère "!" ou une négation relative à une propriété particulière en utilisant le caractère "-".

Il est également possible d'utiliser des expressions rationnelles de type SED, GREP ou PERL pour limiter la recherche à l'ensemble des formes qui correspondent à un certain modèle. Ce type de requêtes est introduit par l'ajout d'une contrainte de type « + MP ="..." ». Par exemple, l'expression rationnelle <N+MP="^اللا"+MP="ة$"> permet de rechercher toutes les formes nominales, qui commencent par "اللا" et qui se terminent par "ة" => "اللامركزية" ou "اللاسلكية",... Notons que, pour tenir compte de la deuxième contrainte, les instances telles que "اللاوعي" seront écartées.

7.5.2 Requêtes syntaxiques ou syntagmatiques

Ce deuxième type de requêtes utilise, entre autres, les annotations attribuées à l'aide de grammaires locales telles que celles reconnaissant des entités nommées (noms de personnes, expressions de temps, expressions numériques), des expressions ou termes médicaux classés selon une vingtaine de domaines de la biomédecine, des constructions verbales telles que les constructions aspectuelles ou modales, ainsi que quelques constructions nominales, etc.

Parmi les requêtes syntaxiques, nous pouvons citer :

- <ENAMEX+PERS+POLIT> : recherche de tous les noms de personnes politiques → " طوني بلير؛ الوزير الأوّل الانجليزي"; …
- <TIMEX+HEURE> : recherche de toutes les expressions temporelles indiquant l'heure → "الثانية بعد الزوال", "على الساعة السابعة صباحا", …

- <V+F> + <سَوْفَ><V+P>: recherche de toutes occurrences du futur formées soit par un verbe fléchie au futur (<V+F>), soit par la particule du futur <سَوْفَ> (<*sawfa*>) suivie par un verbe conjugué au présent de l'indicatif (<V+P>)→ "سوف تكتشفون", "سأذهب", …
- (<فِي><شَرَعَ>+<اِفْتَتَحَ>+<اِسْتَهَلَّ>+<بَدَأَ>)<N> : recherche des constructions aspectuelles inchoatives formés par : un verbe à aspect inchoatif, éventuellement accompagné d'une préposition, tel que (<فِي><شَرَعَ>+<اِفْتَتَحَ>+<اِسْتَهَلَّ>+<بَدَأَ>), suivi par un nom (<N>) → "شرع في عمله", …

7.6 Fonctionnalités de NooJ4Web

Initialement, un concordancier désignait la personne qui déterminait manuellement des concordances par la lecture d'un texte écrit en extrayant toutes les occurrences d'un mot ou concept particulier ainsi que les contextes dans lesquels elles apparaissent. Actuellement, un concordancier fait référence à des programmes informatiques qui remplissent cette tâche. Sans doute, tous ces travaux pourraient être effectués manuellement, mais la tâche reste assez accablante à réaliser. L'application Web NooJ4Web, en plus de sa tâche principale de concordancier, peut être utilisée pour différents objectifs, tels que :

7.6.1 Recherche linguistique

NooJ4Web, comme tout concordancier, peut être utilisé comme un moteur de recherche servant à reconnaître l'occurrence d'un mot ou d'une séquence de lettres dans un ensemble donné de textes. Parmi d'autres options de recherche, il comprend des possibilités d'utilisation des opérateurs de sélection tels que ET, OU et SAUF permettant ainsi d'effectuer des recherches plus complexes.

7.6.2 Analyse quantitative

NooJ4Web fournit une analyse quantitative du corpus traité par le biais d'un rapport statistique. En fait, le programme construit une grille de données offrant :

- Les noms des fichiers sources dans le corpus sélectionné ;
- La proportion de chacun des textes dans le corpus choisi ;
- La fréquence absolue des occurrences ;
- La fréquence attendue / moyenne pondérée des occurrences ;
- L'écart-type.

Les informations mentionnées peuvent être triées selon l'ordre alphabétique ou l'ordre des valeurs de l'une des colonnes. L'analyse quantitative fournit également le nombre total d'occurrences, le nombre des différentes instances ainsi que le nombre de textes répondant à la requête. Les statistiques affichées facilitent la tâche de l'analyste.

7.6.3 Analyse qualitative

Les concordances affichées peuvent servir pour une analyse qualitative ainsi qu'une interprétation de la fonction sous-jacente des textes. Ces concordances peuvent être utilisées pour détecter les cas d'intertextualité, c'est-à-dire détecter les liens du texte avec d'aucuns écrits religieux, certains poètes, quelques romans gothiques ou romans de la renaissance. Il est aussi possible d'établir le degré d'hétérogénéité d'un texte ou de l'ensemble de tous les textes dans un corpus.

NooJ4Web analyse les textes dans le but d'extraire des informations autour d'un certain type d'événements ou d'entités. En effet, NooJ4Web peut être utilisé comme :

- Un outil d'extraction d'entités nommées (NER) : NooJ4Web utilise des grammaires locales de reconnaissance d'entités nommées permettant l'identification et le typage de :
  - ✓ Noms propres – ENAMEX : noms de personnes (+PERS), noms d'organisations (+ORG) et les localisations (+LOC) ;
  - ✓ Expressions temporelles – TIMEX : expressions de date, heure, période, etc. ;

- ✓ Expressions numériques – NUMEX : expressions monétaires, pourcentages, longueurs, profondeurs, etc.

- Un système de veille médicale : NooJ4Web peut être utilisé pour effectuer des recherches automatiques planifiées ou quotidiennes d'expressions et termes médicaux dans un corpus dynamique. En cas de dépassement d'un seuil prédéfini du nombre moyen attendu de ce type d'expressions, notre système de veille déclenche une alerte pour signaler une potentielle apparition d'un événement lié à la biomédecine tel que l'organisation de la journée mondiale de lutte contre le SIDA ou l'apparition d'une nouvelle épidémie (comme la grippe aviaire), etc.

7.6.4 Aide à l'enseignement de l'arabe

NooJ4Web peut être utilisé par les enseignants afin de trouver des exemples authentiques et de montrer quelques caractéristiques du vocabulaire, de certains aspects de la grammaire ou encore de la structure d'un texte. Les enseignants peuvent générer des exercices reposant sur des exemples tirés de la variété des corpus disponibles. Il peut être aussi utilisé en vue de préparer des polycopiés ou des feuilles de travail [Silberztein et Tutin, 2004]. En outre, les étudiants peuvent travailler sur des règles de la grammaire ou des propriétés lexicales par la recherche de certaines formes dans leur contexte. Ils peuvent ainsi être invités à faire des analyses linguistiques approfondies et découvrir de nouvelles significations et utilisations de certaines expressions ou collocations.

7.7 Illustration à travers un exemple

NooJ4Web est une application Web développée en ASP.NET (cf. http://nooj4web.univ-fcomte.fr/). Notre système de concordances est monolingue; Cependant, nous proposons 3 interfaces différentes pour le traitement de l'arabe, le français et l'anglais (cf. Figure 85).

Figure 85 : **Les 3 en-têtes des interfaces utilisateurs dans NooJ4Web**

Dans ce qui suit, nous exposons la séquence d'exécution d'une requête en utilisant l'interface arabe de NooJ4Web montrée dans la figure ci-après.

L'utilisateur a la possibilité de choisir l'un des corpus décrits ci-dessus et de sélectionner une requête parmi une vingtaine d'exemples fournis pour des fins pédagogiques. Une brève description du dernier élément sélectionné, un corpus ou une requête, est affichée dans un espace réservé, côté droit. Dans la capture d'écran ci-dessous, nous affichons, à droite de la liste des requêtes, une description de l'expression rationnelle sélectionnée : "<V+F> + <سَوْفَ><V+P>". Cette requête permet la reconnaissance du concept du futur dans le corpus sélectionné (l'ensemble des textes journalistiques dans cet exemple). Dans la zone réservée aux descriptions des ressources sélectionnées, nous exposons la liste des journaux qui ont servi pour la construction de notre corpus dynamique.A ce propos, nous signalons qu'en cas de description de la requête selectionnée, nous illustrons les diverses possibilités à travers des exemples.

Figure 86 : Interface arabe : للواب « NooJ » – NooJ4Web

Légende :

1 : Liens pour l'interface française et l'interface anglaise ;

2 : Liste des différents corpus (journalistiques, littéraires et juridiques ;

3 : Liste des requêtes prédéfinies : une vingtaine d'expressions rationnelles donnée à titre d'exemple ;

4 : Zone de texte consacrée à la saisie d'une requête autre que celles de la liste des exemples ;

5 : Cadre réservé à la description de la requête ou du corpus dernièrement sélectionné ;

6 : Bouton pour le lancement de l'analyse ;

7 : Informations générales : numéro du visiteur actuel, utilisation du moteur linguistique NooJ et financement par le projet VODEL.

Après soumission de la requête, nous affichons, dans une nouvelle page Web (cf. Figure 87), la concordance présentant la possibilité de visualisation d'un seul exemple par occurrences ou la totalité des occurrences dans la grille de données. Cette grille de données affiche chaque occurrence entourée par ses deux contextes ainsi que le nom du fichier source. Il est aussi possible de trier sur la valeur de n'importe quelle colonne. De plus, chaque occurrence est représentée par un lien hypertexte permettant d'élargir le contexte cinq fois par un simple click.

Figure 87 : Table de concordances dans NooJ4Web

Légende :

① : Rappel du corpus sélectionné et de la requête soumise ;

② : Liste des paramètres d'affichage de la concordance ;

③ : Paramétrage de l'affichage des occurrences ;

④ : Paramétrage du nombre de lignes par page : 10, 20 ou 50 ;

⑤ : Paramétrage du nombre total de lignes de concordances (100, 200 ou 500) et affichage des « Premières lignes », « Lignes choisies aléatoirement » ou « Lignes réparties sur les différents textes » ;

⑥ : Lien pour l'affichage du rapport statistique ;

⑦ : Table de concordances avec possibilité de tri sur chacune des colonnes. Chaque ligne affiche le texte source, l'occurrence entourée par ses contextes droit et gauche ;

⑧ : Liens pour la pagination de la table de concordances ;

⑨ : Bouton pour exporter les résultats des concordances.

NooJ4Web fournit la possibilité de construire un rapport statistique, comme indiqué dans la Figure 88. Le rapport statistique comporte le calcul de l'écart type des occurrences dans chaque texte. Les données affichées dans ce tableau incluent le nom du fichier source, la fréquence absolue dans le texte, la fréquence moyenne, ainsi que l'écart type calculé relativement à la proportion du corpus représentée par chaque texte.

La figure ci-dessous montre un rapport statistique comportant les valeurs suivantes :

- **La fréquence absolue** des occurrences dans le texte : $Freq[i]$;
- **Le ratio du texte** : $Ratio[i] = \frac{Taille[i]}{\sum_{j=1}^{n} Taille[j]}$;
- **La fréquence attendue** / **moyenne pondérée** :

  $\overline{Freq} = \frac{\sum_{i=1}^{n} Freq[i] * Taille[i]}{\sum_{i=1}^{n} Taille[i]}$;
- **Ecart type :**

  $= \frac{Freq[i] - \overline{Freq}}{\sqrt{\overline{Freq} * (1 - Ratio[i])}}$.

Figure 88 : Rapport statistique des concordances

Légende :

1 : Rappel du corpus sélectionné et de la requête soumise ;

2 : Bouton et lien pour l'affichage de l'histogramme de l'Ecart-Type ;

3 : Table de concordances avec possibilité de tri sur chacune des colonnes. Chaque ligne affiche le texte source, la fréquence absolue des occurrences, le ratio du texte dans le corpus, la moyenne pondérée et l'écart-type ;

4 : Affichage de quelques informations statistiques (nombre total d'occurrences, nombre total de textes, etc.).

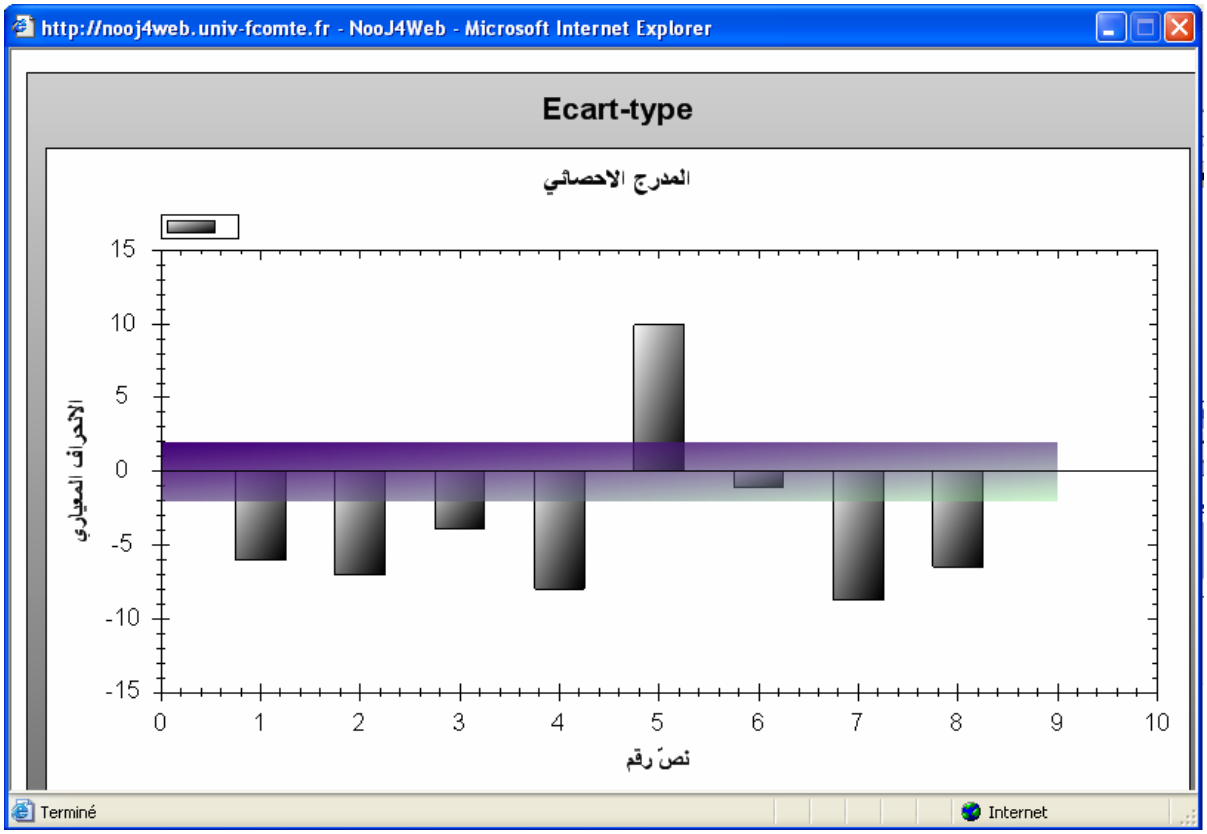


Figure 89 : Histogramme de l'écart type

A la demande de l'utilisateur, nous construisons, dynamiquement, à partir des valeurs affichées dans le rapport statistique précédent, l'histogramme de l'écart-type des occurrences (voir Figure 89). Cet histogramme montre la déviation standard de chaque texte dans le corpus sélectionné en termes du nombre d'apparition du motif recherché. En effet, le calcul de l'écart type permet de déterminer la répartition ou la dispersion du nombre d'occurrences dans chaque texte autour de la valeur moyenne. Alors qu'une déviation importante se traduit par un nombre d'occurrences beaucoup plus important que la valeur moyenne, une variation mineure indique que le nombre d'occurrences dans le texte en question tourne autour de la fréquence attendue. Nous remarquons que cette mesure ne donne une signification importante pour les statisticiens qu'au cas où elle s'élève à plus de "2" ou qu'elle est réduite à moins de "-2". Dans l'exemple montré à la Figure 89, seuls les textes dont la barre dépasse la zone mise en évidence, en violet, permettrait une interprétation statistique antérieure.

7.8 Conclusion

Dans ce chapitre, nous avons décrit un service de concordances en ligne, NooJ4Web, basé sur l'utilisation du moteur linguistique NooJ pour traiter des corpus statiques et dynamiques. Une architecture client / serveur a été développée pour fournir des résultats instantanés aux requêtes soumises sous forme d'expressions rationnelles. NooJ4Web peut être utilisé comme un système d'extraction de mots, séquences ou phrases à partir d'une masse assez importante de données linguistiques. Bien que conçu, principalement, pour des linguistes, NooJ4Web peut être utilisé par des étudiants, des professeurs de langue, des lexicographes, des journalistes, ainsi que ceux effectuant des recherches générales. À présent, nous travaillons sur l'amélioration des services proposés dans NooJ4Web ; D'une part, nous essayons d'étendre le nombre de corpus ainsi que celui des langues traitées. D'autre part, nous étudions la possibilité d'offrir les mêmes services aux utilisateurs souhaitant travailler sur leurs propres textes.

Conclusion et perspectives

L'objectif principal visé par ce travail était d'étudier et de formaliser un lexique à large couverture ainsi que de concevoir et réaliser un analyseur morpho-syntaxique automatique de la langue arabe. Cet objectif est accompli grâce aux divers aspects novateurs que nous avons intégrés dans les différents volets de ce travail :

- *Construction et compactage des dictionnaires électroniques de l'arabe* : Nous avons étudié la macrostructure et la microstructure des dictionnaires électroniques de la langue arabe. Cette étude a porté sur la distinction de l'ensemble des informations qui doivent figurer dans le lexique tout en prenant en compte les différentes spécificités de la langue telles que la voyellation, la flexion et la dérivation. Le lexique construit, baptisé « El-DicAr », a permis de générer automatiquement plus de 3 millions de formes fléchies accompagnées de leurs informations flexionnelles, morphologiques, syntactico-sémantiques à partir d'une liste d'environ quarante cinq milles lemmes. L'ensemble de ces formes fléchies est stocké dans des machines à états finis répandues comme étant les plus adéquates pour représenter de tels volumes de données. Cependant, les méthodes traditionnelles de minimisation et compression de ces transducteurs étaient insuffisantes pour la prise en compte de la structure infixale des formes fléchies ainsi que les informations qui s'y rattachent. Nous avons proposé un nouvel algorithme pour la compression dynamique de ces données. En moyenne, cet algorithme affiche un taux de compression moyen de 75,1 % par rapport à la méthode courante, permettant ainsi de réduire la taille du fichier contenant la liste des formes fléchies de 40 %.
- *Analyse morpho-lexicale robuste de la langue arabe* : Nous avons adopté une approche déclarative. En effet, outre la cohérence théorique de cette approche, elle facilite la tâche de maintenance et d'extension du système. A ce niveau d'analyse, nous convenons de signaler que la majorité des mots arabes ont une structure complexe par adjonction de clitiques aux formes fléchies. Souvent, cette agglutination s'accompagne d'une altération de la forme initiale du mot nécessitant, ainsi l'application d'un ensemble de contraintes lexicales représentées à l'intérieur des grammaires morphologiques et liées aux informations introduites dans le dictionnaire « El-DicAr ». Ces contraintes permettent de ne retenir que les découpages dont l'agencement des morphèmes est valide. Par ailleurs, nous avons proposé un nouvel algorithme de parcours des machines à états finis décrivant le lexique afin de pouvoir traiter aussi bien des textes voyellés et non voyellés que ceux qui le sont partiellement, sans aucun calcul supplémentaire. Cet algorithme a permis de faire évoluer le moteur lexical de la plateforme NooJ pour traiter les langues sémitiques. En outre, la robustesse de notre analyseur a été renforcée par la prise en compte de certaines erreurs typographiques fréquentes de manière à limiter les répercussions de leurs non reconnaissances sur les tâches ultérieures d'analyse. Finalement, nous avons exposé les différentes possibilités d'exploration des annotations ainsi introduites pour faire des recherches linguistiques d'ordre lexical ou morphologique.
- *Reconnaissance automatique d'entités nommées* : Nous avons commencé par une étude sur la typologie des entités nommées et les problèmes majeurs auxquels leur reconnaissance automatique peut être confrontée. En l'absence d'indices naïfs pour l'identification des entités nommées et plus particulièrement les noms propres (personnes, lieux et organisations) et compte tenu de leur abondance notamment dans les textes journalistiques (85% des formes non reconnues par analyse morpho-lexicale), nous avons construit un système de reconnaissance d'entités nommées basé sur l'écriture de règles d'identification dans des transducteurs à états finis et des réseaux de transitions augmentés (ATNs). Ces machines à états finis permettent aussi bien l'identification que l'étiquetage et l'attribution des traits distributionnels adéquats (+PERS ou +Militaire + …..). A l'issue de toutes ces étapes de formalisation, une analyse automatique menée sur notre corpus du journal « Le Monde Diplomatique » a aboutit à une couverture lexicale de plus que 95 %.

- *Développement d'un service de concordances en ligne* : L'ensemble des outils, ci-dessus décrits, fournit des analyses complètes, détaillées et fines des différentes formes lexicales, compositions morphologiques ou structures syntaxiques prises en compte. Ces analyses sont mises à disposition de différents utilisateurs (linguistes, informaticiens, enseignants, documentalistes, etc.) en vue d'effectuer l'analyse de corpus statiques et dynamiques par le biais de l'application web de concordances en ligne « NooJ4Web ». Cette application est construite sur une architecture client / serveur basée sur le moteur linguistique de NooJ : une architecture qui permet d'optimiser le temps de traitement et de fournir des résultats instantanés à des requêtes écrites sous forme d'expressions rationnelles. Les résultats sont affichés sous forme de concordances dans une grille de données faisant apparaître les séquences pertinentes dans leur contexte. Quelques résultats statistiques sont aussi fournis aux utilisateurs. Notre concordancier est monolingue; Toutefois, nous traitons des corpus écrits en arabe, en français et en anglais (cf. http://nooj4web.univ-fcomte.fr). Ce service permet la validation des ressources développées à travers leurs mises à la disposition de la communauté scientifique. Une analyse approfondie des réactions des différents utilisateurs est envisagée avant de les adopter en vue d'améliorer les performances de notre système.

Les perspectives de nos travaux sont multiples :

- Notre système de reconnaissance d'entités nommées vient d'être repris par des collègues du laboratoire MIRACL à l'université de Sfax en Tunisie en vue de sa réutilisation. Dans un premier lieu, l'objectif de ce travail est de réaliser une désambiguïsation sémantique des entités nommées retenues ; dans un second lieu, il vise l'analyse syntaxique des phrases simples et composées.
- L'architecture de NooJ4Web est facilement extensible pour admettre un système de questions/réponses automatique. Ce module permettra de répondre aux interrogations des utilisateurs écrites en langue naturelle pour en fournir une réponse automatique à partir des corpus statiques déjà stockés (notamment les textes techniques et juridiques) ainsi que les corpus dynamiques construits à partir d'articles d'actualité.

ANNEXES

Annexe A :
Tableau de translittération de l'alphabet arabe

Lettre	Nom	Translittération	Commentaires
ء	*hamzä*	’	
ا	*álif*	ā	Voyelle longue (*)[54]
ب	*Bâ’*	b	Lettre lunaire
ت	*Tâ’*	t	Lettre solaire
ث	*Tâ’*	t̲	Lettre solaire
ج	*Jîm*	ǧ	Lettre lunaire
ح	*ḥâ’*	ḥ	Lettre lunaire
خ	*Xâ’*	h̲	Lettre lunaire
د	*dâl*	d	Lettre solaire (*)
ذ	*d̲âl*	d̲	Lettre solaire (*)
ر	*Râ’*	r	Lettre solaire (*)
ز	*zây*	z	Lettre solaire (*)
س	*Sîn*	s	Lettre solaire
ش	*Sîn*	š	Lettre solaire
ص	*ṣâd*	ṣ	Lettre solaire
ض	*ḍâd*	ḍ	Lettre solaire
ط	*ṭâ’*	ṭ	Lettre solaire
ظ	*ẓâ’*	ẓ	Lettre solaire
ع	*ʿayn*	ʿ	Lettre lunaire
غ	*ġayn*	ġ	Lettre lunaire

[54] Le symbole (*) désigne les six lettres de l'alphabet qui ne s'attachent jamais à la lettre suivante.

ف	*Fâ'*	f	Lettre lunaire
ق	*qâf*	q	Lettre lunaire
ك	*kâf*	k	Lettre lunaire
ل	*Lâm*	l	Lettre solaire
م	*mîm*	m	Lettre lunaire
ن	*nûn*	n	Lettre solaire
ه	*Hâ'*	h	Lettre lunaire
و	*wâw*	w / ū	Lettre lunaire (*) / Voyelle longue
ي	*Yâ'*	y / ī	Lettre lunaire / Voyelle longue
ـَ	*fatḥä*	a	Voyelle brève
ـُ	*ḍammä*	u	Voyelle brève
ـِ	*kasrä*	i	Voyelle brève
ـً	*tanwîn*	ã / an	Voyelle brève / Tanwîn
ـٌ	*tanwîn*	ũ / un	Voyelle brève/ Tanwîn
ـٍ	*tanwîn*	ĩ / in	Voyelle brève/ Tanwîn
ـَة	*tâ' marbûṭä*	ä (at en annexion)	
ى	*Alif maqṣûrä*	ą	
آ	*Alif mamdûdä*	å̀	
أ	*hamzä*	á	
أُ	hamzä	ù	
إ	*hamzä*	í	
ؤ	*hamzä*	ẃ	
ئ	*hamzä*	ý	
ـْ	*sukun*		non transcrit
ـّ	*šaddä*	lettre redoublée	Signe de gémination

Annexe B :
Module de l'arabe dans NooJ

Tout au long de ce travail, nous avons utilisé la plateforme linguistique de développement NooJ pour construire, tester, maintenir et gérer des dictionnaires électroniques et des grammaires à large couverture afin de formaliser différents niveaux d'analyse de la langue arabe écrite : lexique des mots simples et composés, morphologie flexionnelle, morphologie dérivationnelle, structure agglutinative, orthographe, syntaxe locale et désambiguïsation. Nous avons, par la suite, montré que ces descriptions formalisées peuvent être appliquées aux textes et corpus afin de construire des concordances complexes et annoter interactivement le corpus traité. L'ensemble des ressources linguistiques ainsi construites constitue le module de l'arabe pour NooJ. Ce module est téléchargeable gratuitement à l'adresse http://www.nooj4nlp.net ; il est constitué de toute une série de ressources linguistiques :

- Les dictionnaires de l'arabe : le dictionnaire des noms, verbes, adjectifs, particules, prénoms, noms de lieux et noms d'organisations. Nous fournissons, aussi, un dictionnaire exemple à source ouverte fourni avec l'ensemble des descriptions flexionnelles et dérivationnelles qui y sont incluses. Tout le contenu de ce dictionnaire échantillon est donné en AnnexeS C et D ;
- Une grammaire morphologique de tokenisation incluant 38 sous-graphes ;
- Une grammaire morphologique de correction des erreurs orthographiques les plus fréquentes ;
- Un ensemble de grammaires locales, syntactico-sémantiques :
  - ✓ une grammaire locale de reconnaissance des déterminants numériques. Cette grammaire permet l'annotation des expressions reconnues sous la forme : <DET+NUM+Val=xxx> sachant que la propriété « +Val=xxx » correspond à l'attribution automatique de la valeur numérique des déterminants retenus ;
  - ✓ une grammaire locale de reconnaissance des noms de personnes. Cette grammaire permet l'annotation des expressions reconnues sous la forme : <ENAMEX+PERS> avec la possibilité d'avoir une sous-catégorisation telle que « +POLIT » pour les noms de personnes politiques, « +MILIT » pour les personnes militaires, etc.
  - ✓ une grammaire locale de reconnaissance des noms de lieux. Cette grammaire permet l'annotation des expressions reconnues sous la forme : <ENAMEX+LOC> avec la possibilité d'avoir une sous-catégorisation telles que « +Pays », « +Ville », « +Etat », « +Montagne », etc. ;
  - ✓ une grammaire locale de reconnaissance des noms d'organisations. Cette grammaire permet l'annotation des expressions reconnues sous la forme : <ENAMEX+ORG> ;
  - ✓ une grammaire locale de reconnaissance des expressions temporelles incluant heures (<TIMEX+Heure>), dates (<TIMEX+Date>), périodes (<TIMEX+Période>), etc. ;
  - ✓ une grammaire locale de reconnaissance des expressions numériques y compris les expressions monétaires (<ENAMEX+Monétaire>), les mesures de distance (<NUMEX+Mesure>), de surfaces (<NUMEX+Surface>), les pourcentages (<NUMEX+Pourcentage>), etc.
- Une grammaire locale de désambiguïsation ;
- Le texte de la déclaration universelle des droits de l'homme en arabe ;
- Un petit corpus de phrases permettant de tester les différents phénomènes morpho-syntaxiques étudiés telles que la dérivation, l'agglutination, la voyellation automatique, la correction orthographique, la reconnaissance des entités nommées, la désambiguïsation, etc.

Annexe C :
Quelques extraits de « El-DicAr »

C1 Dictionnaire: « Exemple.dic »

```
# NooJ V2
# Dictionary
#
# Input Language is: ar
#
# Alphabetical order is not required.
#
# Use inflectional & derivational paradigms' and properties' definition files (.nof or .def), e.g.:
# Special Command: #use properties.def
# Special Command: #use paradigms.nof
#
# Special Features: +NW (non-word) +FXC (frozen expression component) +UNAMB
(unambiguous lexical entry)
#                +FLX= (inflectional paradigm) +DRV= (derivational paradigm)
#
# Special Characters: '\' '"' '+' ',' '#' ' '
#
#use _Exemple.flx
#use _Properties.def

# Quelques mots invariables
مِنْ,PREP
حِينَ,ADV

# Quelques noms

كِتَاب,N+Conc+FLX=kitAb+DRV=kutub:FlexionPL
جَزِيرَة,N+Conc+FLX=jazIra+DRV= kutub:FlexionPL
رَجُل,N+Conc+Hum+FLX=kitAb+DRV=rijAl:FlexionPL
لجْنَة,N+Abst+FLX= jazIra+DRV=rijAl:FlexionPL

# Quelques verbes

# Les deux verbes "كَتَبَ" (*kataba* – écrire) et "ذَكَرَ" (*dakara* – rappeler, mentionner) se conjuguent selon le même modèle (V_kataba)

# Les trois verbes "كَتَبَ" (*kataba* – écrire), "ذَكَرَ" (*dakara* – rappeler, mentionner) et "جَرَحَ " (*jaraha* – blesser) se dérivent selon le même modèle (D_faâala)

كَتَبَ,V+Tr+FLX=V_kataba+DRV=D_ faâala:FlxDRV
ذَكَرَ,V+Tr+FLX= V_kataba +DRV=D_ faâala:FlxDRV
```

جَرَحَ,V+Tr+FLX=V_jaraha+DRV=D_faâala:FlxDRV

Quelques mots composés

كُرَة القَدَم,N+Sport+FLX=FlexionNC1

C2 Fichier des descriptions flexionnelles et dérivationnelles : « Exemple.nof »

```
# NooJ V2
# Inflectional/Derivational Description
#
# Language is: ar
#
# Special Characters: '=' '<' '>' '\' '"' '+' '/' '#' ' '
#
# Generic Commands:
# <B>: keyboard Backspace
# <C>: change Case
# <D>: Duplicate current char
# <E>: Empty string
# <L>: keyboard Left arrow
# <N>: go to end of Next word form
# <P>: go to end of Previous word form
# <R>: keyboard Right arrow
# <S>: delete/Suppress current char
# Arguments for commands <B>, <L>, <N>, <P>, <R>, <S>:
# xx number: repeat xx times
# W: whole word
# Examples
# <R3>: go right 3 times
# <LW>: go to beg. of word
#
# Language-Specific Commands:
# <T>: replaces Teh marbuta with Teh
# <M>: processes consonant 'n' for past form, 3rd person, feminine plural
# <Z>: processes consonant 't' for past form, 1st person, singular
```

Acc_faâala=

```
# Voici  les formes à l'accompli, voix active :
<B>(<Z>ُA+I+1+s +
<M>اَA+I+1+P +
<Z>َA+I+2+m+s +
<Z>ِA+I+2+f+s +
<Z>ُمَاA+I+d +
<Z>ُمْA+I+2+m+p +
<Z>ُنَّA+I+2+f+p +
اُوA+I+3+m+p +
<M>َA+I+3+f+p) +
<E>/A+I+3+m+s +
تْA+I+3+f+s +
اِA+I+5+m +
تَاA+I+5+f ;
```

Passive_faâala=

Voici les formes au passé, voix passive :
(<L5>ُ<S><R><S>ِ<R><S>) (تُ/K+I+1+s +
نَا/K+I+1+P +
تَ/K+I+2+m+s +
تِ/K+I+2+f+s +
تُمَا/K+I+d +
تُمْ/K+I+2+m+p +
تُنَّ/K+I+2+f+p +
وا/K+I+3+m+p +
نَ/K+I+3+f+p) +

(<L5>ُ<S><R><S>ِ<R2>) (<E>/K+I+3+m+s +
تْ/K+I+3+f+s +
ا/K+I+5 +
تَا/K+I+5+f) +

Voici les formes à l'inaccompli (présent de l'indicatif), voix passive :

<LW> (أ<R><S>ْ<R><S>َ<R><S>ُ/K+1+s +
نُ<R><S>ْ<R><S>َ<R><S>ُ/K+1+p) +

(<LW>تُ<R><S>ْ<R><S>َ<R>)
(<S>ُ/K+2+m+s +
<S>ِينَ/K+2+f+s +
<R>ان/K+d +
<S>ُونَ/K+2+m+p +
<S>نَ/K+2+f+p +
<S>ُ/K+3+f+s +
<R>ان/K+5+f) +

(<LW>يُ<R><S>ْ<R><S>َ<R>)
(<S>ُ/K+3+m+s +
<R>ان/K+5+m +
<S>ُونَ/K+3+m+p +
<S>نَ/K+3+f+p) +

Voici les formes au présent au subjonctif, voix passive :

<LW> (أ<R><S>ْ<R><S>َ<R2>/K+S+1+s +
نُ<R><S>ْ<R><S>َ<R2>/K+S+1+p) +

(<LW>تُ<R><S>ْ<R><S>َ<R>)
(<R>/K+S+2+m+s +
ِي/K+S+2+f+s +
<R>ا/K+S+d +
<S>ُوا/K+S+2+m+p +
نَ/K+S+2+f+p +

<R>/K+S+3+f+s +
<R>ا/K+S+5+f) +

(<LW>يُ<R><S>ْ<R><S>َ<R>)
(<R>/K+S+3+m+s +
<R>ا/K+S+5+m +
<S>ُوا/K+S+3+m+p +
<S>نَ/K+S+3+f+p) +

Voici les formes au présent apocopé, voix passive :

<LW> (أ<R><S>ْ<R><S>َ<R><S>/K+C+1+s +
نُ<R><S>ْ<R><S>َ<R><S>/K+C+1+p) +

(<LW>تُ<R><S>ْ<R><S>َ<R>)
(<S>/K+C+2+m+s +
<S>ي/K+C+2+f+s +
<R>ا/K+C+d +
<S>ُوا/K+C+2+m+p +
<S>نَ/K+C+2+f+p +
<S>/K+C+3+f+s +
<R>ا/K+C+5+f) +

(<LW>يُ<R><S>ْ<R><S>َ<R>)
(<S>/K+C+3+m+s +
<R>ا/K+C+5+m +
<S>ُوا/K+C+3+m+p +
<S>نَ/K+C+3+f+p) ;

V_kataba=

Voici les formes à l'accompli, voix active et passive :
:Acc_faâala +

Voici les formes au présent de l'indicatif, voix active:
<E>/P **: Inacc_kataba** +

Voici les formes au futur :
سَ/F **: Inacc_kataba** +

Voici les formes à la voix passive (passé, présent de l'indicatif, subjonctif et apocopé):
:Passive_faâala +

Voici les formes au subjonctif, voix active :

<LW> (أ<R><S>ْ<R><S>ُ<R2>/A+S+1+s +
نَ<R><S>ْ<R><S>ُ<R2>/A+S+1+p) +

(<LW>تَ<R><S>ْ<R><S>ُ<R>)
(<R>/A+S+2+m+s +
<S>ي/A+S+2+f+s +
<R>ا/A+S+d +
<R>ا/وُA+S+2+m+p +
<R>نْ/A+S+2+f+p +
<R>/A+S+3+f+s +
<R>ا/A+S+5+f) +

(<LW>يَ<R><S>ْ<R><S>ُ<R>)
(<R>/A+S+3+m+s +
<R>ا/A+S+5+m +
<S>وا/A+S+3+m+p +
<S>نْ/A+S+3+f+p) +

Voici les formes au présent apocopé, voix active :

<LW> (أ<R><S>ْ<R><S>ُ<R><S>ْ/A+C+1+s +
نَ<R><S>ْ<R><S>ُ<R><S>ْ/A+C+1+p) +

(<LW>تَ<R><S>ْ<R><S>ُ<R>)
(<S>ْ/A+C+2+m+s +
<S>ي/A+C+2+f+s +
<R>ا/A+C+d +
<S>وا/A+C+2+m+p +
<S>نْ/A+C+2+f+p +
<S>ْ/A+C+3+f+s +
<R>ا/A+C+5+f) +

(<LW>يَ<R><S>ْ<R><S>ُ<R>)
(<S>ْ/A+C+3+m+s +
<R>ا/A+C+5+m +
<S>وا/A+C+3+m+p +
<S>نْ/A+C+3+f+p) +

Voici les formes à l'impératif :
(<LW>ا<R><S>ْ<R><S>ُ<R><S>)
(ْ/Y+2+m+s +
ي/Y+2+f+s +
وا/Y+2+m+p +
<M>َ/Y+2+f+p +
ا/Y+d);

V_jaraha=

Voici les formes à l'accompli, voix active et passive :
:Acc_faâala +

Voici les formes au présent de l'indicatif, voix active:
<E>/P **: Inacc_jaraha** +

Voici les formes au futur :
سَـ/F **: Inacc_jaraha** +

Voici les formes à la voix passive (passé, présent de l'indicatif, subjonctif et apocopé):
:Passive_faâala +

Voici les formes au subjonctif, voix active :

<LW> (أ<R><S>ْ<R> <S>َ<R2>/A+S+1+s +
نَ<R><S>ْ<R> <S>َ<R2>/A+S+1+p) +

(<LW>تَ<R><S>ْ<R> <S>َ<R>)
(<R>/A+S+2+m+s +
<S>ي/A+S+2+f+s +
<R>ا/A+S+d +
<R>ا/وُA+S+2+m+p +
<R>نَ/ْA+S+2+f+p +
<R>/A+S+3+f+s +
<R>ا/A+S+5+f) +

(<LW>يَ<R><S>ْ<R> <S>َ<R>)
(<R>/A+S+3+m+s +
<R>ا/A+S+5+m +
<S>ا/وُA+S+3+m+p +
<S>نَ/ْA+S+3+f+p) +

Voici les formes au présent apocopé, voix active :

<LW> (أ<R><S>ْ<R> <S>َ<R><S>ْ/A+C+1+s +
نَ<R><S>ْ<R> <S>َ<R><S>ْ/A+C+1+p) +

(<LW>تَ<R><S>ْ<R>< <S>َ<R>)
(<S>ْ/A+C+2+m+s +
<S>ي/A+C+2+f+s +
<R>ا/A+C+d +
<S>ا/وُA+C+2+m+p +
<S>نَ/ْA+C+2+f+p +
<S>ْ/A+C+3+f+s +
<R>ا/A+C+5+f) +

(<LW>يَ<R><S>ْ<R> <S>َ<R>)
(<S>ْ/A+C+3+m+s +
<R>ا/A+C+5+m +
<S>ا/وُA+C+3+m+p +
<S>نَ/ْA+C+3+f+p) +

Voici les formes à l'impératif :
<LW> (اِ<R><S>ْ<R><S>َ<R><S>
(ْ/Y+2+m+s +
ي/Y+2+f+s +
ا/وُ/Y+2+m+p +
<M>ِ/Y+2+f+p +
ا/Y+d);

Inacc_kataba=

Voici les formes à l'inaccompli, voix active :

<LW>
(أ<R><S>ْ<R><S>ُ<R><S>/A+1+s +
نَ<R><S>ْ<R><S>ُ<R><S>/A+1+p) +

(<LW>تَ<R><S>ْ<R><S>ُ<R><S>)
(/A+2+m+s +
ينَ/ A+2+f+s +
انِ/ A+2+d +
ونَ/ A+2+m+p +
<M>ِ/ A+2+f+p +
/ A+3+f+s +
انِ/5+f) +

(<LW>يَ<R><S>ْ<R><S>ُ<R><S>)
(/3+m+s +
انِ/5+m +
ونَ/3+m+p +
<M>ِ/3+f+p) ;

Inacc_jaraha=

Voici les formes à l'inaccompli, voix active :

<LW>
(أ<R><S>ْ<R> <S>َ<R><S>/A+1+s +
نَ<R><S>ْ<R> <S>َ<R><S>/A+1+p) +

(<LW>تَ<R><S>ْ<R> <S>َ<R><S>)
(/A+2+m+s +
ينَ/ A+2+f+s +
انِ/ A+2+d +
ونَ/ A+2+m+p +
<M>ِ/ A+2+f+p +
/ A+3+f+s +
انِ/5+f) +

(<LW>يَ<R><S>ْ<R> <S>َ<R><S>)
(/3+m+s +
انِ/5+m +

ونَ/3+m+p +
<M>ٔ/3+f+p);

D_faâala=
#Ism Il Fa3il – Participe actif
<L4>ا<R><S>ِ<R><S>/PA +

#Ism Il Maf3oul – Participe passif
<LW>مَ<R><S>ْ<R><S>وُ<R><S>/PP;

Flexion_DRV=
#Les formes au masculin singulier
ٔ/m+s+a +
ُ/m+s+u +
ِ/m+s+i +
اً/m+s+a+Tanwin +
ٌ/m+s+u+Tanwin +
ٍ/m+s+i+Tanwin +
#Les formes au feminin singulier
(ةَ)
(ٔ/f+s+a +
ُ/f+s+u +
ِ/f+s+i +
اً/f+s+a+Tanwin +
ٌ/f+s+u+Tanwin +
ٍ/f+s+i+Tanwin) +
#Les formes duales et plurielles

ـَيْنِ/m+d+a +
انِ/m+d+u +
ـَيْنِ/m+d+i +
ـَيْ/m+d+a +
اً/m+d+u +
ـَيْ/m+d+i +

ينَ/m+p+a +
ونَ/m+p+u +
ينَ/m+p+i +
ي/m+p+a +
ُوا/m+p+u +
ي/m+p+i +
ُو/m+p+u +

ـَتَيْنِ/m+d+a +
ـَتَانِ/m+d+u +
ـَتَيْنِ/m+d+i +

اتِ/f+p+a +
اتُ/f+p+u +

ـَاتِ/f+p+i +
ـَاتٍ/f+p+a+Tanwin +
ـَاتٌ/f+p+u+Tanwin +
ـَاتٍ/f+p+i+Tanwin;

kutub=
<LW><R><S>ـُ<R><S2>ـُ<R><SW>/N+p;

rijAl=
<LW><R><S>ـِ<R><S>ا<R><SW>/N+p;

Duel_Nom=
(<T>)
(ـَيْنِ/d+a +
ـَانِ/d+u +
ـَيْنِ/d+i +
ـَيْ/d+a +
ا/d+u +
ـَيْ/d+i);

kitAb=
ـَ/a +
ـُ/u +
ـِ/i +
ا/an +
ـٌ/un +
ـٍ/in +
:Duel_Nom;

jazIra=
ـَ/a +
ـُ/u +
ـِ/i +
ـً/an +
ـٌ/un +
ـٍ/in +
:Duel_Nom;

FlexionNC=
ـِ<P>/a +
ـِ<P>ـُ/u +
ـِ<P>/i;

C3 Fichier de définitions des propriétés : « Properties.def »

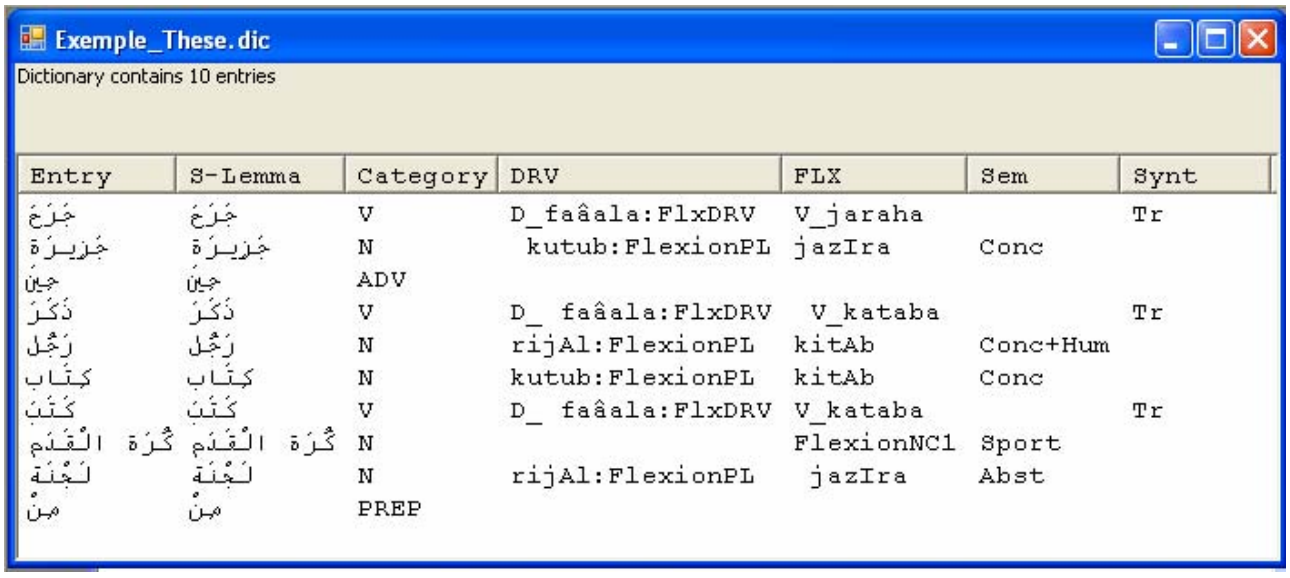
```
# NooJ V2
# Dictionary properties' definition
#
# Language is: ar
#
# Special Characters: '=' '+' '#' ' '
#
# List categories and properties associated with features
#   Example: N_Number = m + f;
# Special KEYWORD: INFLECTION lists all inflectional features (used by variables $xF)
#   Example: INFLECTION = m + f + Present + Futur;

INFLECTION = m + f + s + p +
             1 + 2 + 3 + d + 5 +
             P + I + S + C +
             A + K +
             Y + F;

N_Genre = m + f;
N_Cas = a + u + i + an + un + in;
N_Nombre = s + d + p;
N_Sem = Abst + Conc + Hum + Sport;

V_Pers = 1 + 2 + 3 + d + 5;
V_Voix = A + K;
V_Temps = P + I + S + C + Y + F;
V_Synt = Tr + TrInd + Intr;
V_Genre = m + f;
V_Nombre = s + p;
```

Annexe D :
Visualisation des dictionnaires sous forme de tableau

La figure, ci-dessous, permet de visualiser le dictionnaire donnée en exemple (voir Annexe C1) sous forme d'un tableau. Cette visualisation utilise le fichier des propriétés « Properties.def » afin de pouvoir extraire les propriétés simples à partir de chaque entrée lexicale. Par exemple, si nous considérons les définitions suivantes incluses dans le fichier des propriétés :

N_Nombre = s + d + p;
N_Sem = Abst + Conc + Hum + Sport;

Ces définitions indiquent que "Sem" est une propriété valable pour la catégorie grammaticale "N" (nom), il peut prendre l'une des valeurs suivantes: Abst, Conc, Hum, et Sport. "Nombre" est une aussi une propriété valable pour les noms, mais elle peut prendre trois valeurs « s » (singulier), « d » (duel) ou « p » (pluriel).



Entry	S-Lemma	Category	DRV	FLX	Sem	Synt
جَرَحَ	جَرَحَ	V	D_faâala:FlxDRV	V_jaraha		Tr
جَزِيرَة	جَزِيرَة	N	kutub:FlexionPL	jazIra	Conc	
جِينَ	جِينَ	ADV				
ذَكَرَ	ذَكَرَ	V	D_ faâala:FlxDRV	V_kataba		Tr
رَجُل	رَجُل	N	rijAl:FlexionPL	kitAb	Conc+Hum	
كِتَاب	كِتَاب	N	kutub:FlexionPL	kitAb	Conc	
كَتَبَ	كَتَبَ	V	D_ faâala:FlxDRV	V_kataba		Tr
كُرَة الْقَدَم	كُرَة الْقَدَم	N		FlexionNC1	Sport	
لَجْنَة	لَجْنَة	N	rijAl:FlexionPL	jazIra	Abst	
مِنْ	مِنْ	PREP				

Bibliographie

[Abbes, 2004] R. Abbes, La conception et la réalisation d'un concordancier électronique pour l'arabe, *Thèse de doctorat en Science de l'information et de la communication, INSA Lyon*, (2004).

[Abou Il Azm, 2003] A. Abou Il Azm, Mo3jim tasrif il af3al : 10 000 verbes, *Rabat, Editions Dar Ittawhidi*, (2003).

[Achour, 1998] H. Achour, Contribution à l'étude du problème de la voyellation automatique de l'arabe, *Thèse de doctorat, Université Paris 7*, (1998).

[Ait Taleb, 2005] S. Ait Taleb, Dictionnaires électroniques arabes : le modèle des dictionnaires de Sakhr, *Revue de l'Association Marocaine des Etudes Lexicographiques, Numéro 3-4*, (2005).

[Alharabi, 2006] M. Alharabi, Désambiguïser par le contexte : le cas de la graphie alef-noun en arabe», *Actes du Colloque international Annotation automatique de relations sémantiques et recherche d'informations : vers de nouveaux accès aux savoirs, Université Paris4*, (2006).

[Arbaoui, 2005] N. H. Arbaoui, "Les cardinaux en arabe classique : distribution du genre", *Actes du 9ème Atelier des Doctorants en Linguistique (ADL), LLF-Université Paris 7*, (2005).

[Attia, 2006] M. Attia, An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks, *Actes de la conférence internationale 'The Challenge of Arabic for NLP/MT', The British Computer Society, London,* (2006).

[Azmi, 1988] M. Azmi, Arabic Morphology: A Study in the System of Conjugation, *Hyderabad: Hasan Publishers,* (1988).

[Baccar et al., 2008] F. Baccar, A. Khemakhem, B. Gargouri, K. Haddar, A. Ben Hamadou, Modélisation normalisée LMF des dictionnaires électroniques éditoriaux de l'arabe, *Actes de la conférence internationale TALN'08, Avignon,* (2008).

[Balvet, 2000] A. Balvet, Evaluation de stratégies linguistiques pour le filtrage d'information, *Actes des Troisièmes Journées Intex, Revue Informatique et Statistique dans les Sciences Humaines, Liège, Université de Liège,* (2000).

[Barlow, 1998] M. Barlow, "MonoConc", *Houston TX: Athelstan*, (1998).

[Beesley et Karttunen, 2003] K. Beesley et L. Karttunen, Finite state morphology, *Stanford, California: Ann Copestake (CSLI Studies in Computer Linguistics), 509 p,* (2003).

[Beesley, 2001] K. Beesley, Finite-state Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001, *Actes du Workshop the Arabic Language Processing, ACL Workshop, Toulouse,* (2001).

[Beesley, 1998] K. Beesley, Arabic Morphological Analysis on the Internet, *6th international conference and exhibition on multi-lingual computing, Cambridge,* (1998).

[Ben Hamadou, 1993] A. Ben Hamadou, Vérification et correction automatiques par analyse affixale des textes écrits en langage naturel : cas de l'arabe non voyellé, *Thèse en Sciences informatiques, Faculté des Sciences de Tunis,* (1993)

[Ben Othmane, 1998] C. Ben Othmane – Zribi, De la synthèse lexicographique à la détection et la correction des graphies fautives arabes, *Thèse de doctorat, Université de Paris-Sud, Orsay,* (1998).

[Berrendonner, 1990] A. Berrendonner, Grammaire pour un analyseur : Aspect morphologique, *Nouvelle édition, Grenoble, Université Mendes France, cahier du CRISS N° 15,* (1990).

[Bescherelle, 1999] Bescherelle – Arabe – Verbes, *Editions Hatier, Collection Bescherelle,* (1999).

[Blachère et Gaudefroy, 1975] R. Blachère et M. Gaudefroy-Demombynes, Grammaire de l'arabe classique (morphologie et syntaxe), *G.P. Maisonneuve & Larose, Editeurs à Paris, 508 p.* (1975).

[Bohas, Guillaume et Kouloughli, 1990] G. Bohas, J. P. Guillaume et D. E. Kouloughli, The Arabic Linguistic Tradition, *London, Routledge. Réimpression Georgetown University Press* en 2006, (1990).

[Boualem et al., 1999] M. Boualem, M. Leisher et B. Ogden, Concordancer for Arabic, *Actes de la conférence internationale ATLAS'99*, (1999).

[Brzozowski, 1962] J. A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events. *In Mathematical Theory of Automata, MRI Symposia Series,* (1962).

[Buckwalter, 2002] T. Buckwalter, Arabic Morphological Analyzer Version 1.0, *Linguistic Data Consortium, Catalogue numéro LDC2002L49,* (2002).

[Buckwalter, 2004] T. Buckwalter, Issues in Arabic Orthography and Morphology Analysis. *Actes de la 20ème conférence internationale COLING'04, Workshop on Computational Approches to Arabic Script-bases Languages, Genève, Suisse,* (2004).

[Carrasco et Forcada, 2002] R.C. Carrasco, M.L. Forcada, Incremental construction and maintenance of minimal finite-state automata, *Comput. Linguistics 28,* (2002), pp. 207–216.

[Chairet, 1996] M. Chairet, Fonctionnement du système verbal en arabe et en français, *Gap, Ophrys,* (1996).

[Chalabi, 2004] A. Chalabi, Sakhr Arabic Lexicon, *Actes de la conférence internationale NEMLAR, Arabic Language Resources and Tools, Le Caire, Egypte,* (2004).

[Chrobot, 2000] A. Chrobot, Description des déterminants numéraux anglais par automates et transducteurs finis, *Actes des Troisièmes Journées Intex, Revue Informatique et Statistique dans les Sciences Humaines. Liège, Université de Liège,* (2000).

[Coates-Stephens, 1993] S. Coates-Stephens, "The Analysis and Acquisition of Proper Names for the Understanding of Free Text, *dans Computers and the Humanities, Kluwer Academic Publishers, Vol. 26, Hingham*, (1993).

[Cohen, 1961/1970] D. Cohen, Essai d'une analyse automatique de l'arabe. *Dans: David Cohen. Etudes de linguistique sémitique et arabe. Paris:Mouton, p. 49-78,* (1970).

[Collins et Singer, 1999] M. Collins, et Y. Singer, Unsupervised models for named entity classification", *Actes de the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora,* (1999).

[Constant, 2003] M. Constant, Grammaires locales pour l'analyse automatique de textes : Méthodes de construction et outils de gestion, *Thèse de doctorat, Université de Marne la Vallée,* (2003).

[Courtois et Silberztein, 1990] B. Courtois et M. Silberztein, Dictionnaires électroniques du français. *Langues française 87,* (1990), pp. 11–22.

[Crochemore et Rancart, 1997] M. Crochemore et C. Rancart , Automata for matching patterns. *Handbook Formal Languages, Springer,* (1997).

[Cunningham, 2003] H. Cunningham et K. Bontcheva, "Named Entity Recognition", *Actes de la conférence internationale RANLP 2003, Borovets, Bulgaria,* (2003)

[Daciuk et al., 2000] J. Daciuk, S. Mihov, B.W. Watson, R.E. Watson, Incremental construction of minimal acyclic finite state automata, *Comput. Linguistics 26,* (2000), pp. 3–16.

[Daciuk, 1998] J. Daciuk, Incremental construction of finite-state automata and transducers, and their use in the natural language processing, *Thèse de doctorat en informatique, Technical University of GdaJnsk, Poland,* (1998).

[Daciuk, 2000] J. Daciuk, Experiments with Automata Compression, *in: D. Wood, S. Yu (Eds.), Proceedings of CIAA'2000 conference, London, Canada,* (2000), pp. 113–119.

[Darwish, 2002] K. Darwish, "Building a Shallow Morphological Analyzer in One Day!", *Actes du workshop on Computational Approaches to Semitic Languages au meeting annuel de l'Association for Computational Linguistics (ACL-02), Philadelphie, USA,* (2002).

[Debili et Achour, 1998] F. Debili et H. Achour, Voyellation Automatique De L'arabe, *Actes du Workshop On Computational Approaches To Semitic Languages, Université de Montréal, Canada,* (1998).

[Debili, 2001] Debili F, Traitement automatique de l'arabe voyellé ou non, *Correspondances n°46, IRMC, Tunis,* (2001).

[Dichy et Fargaly, 2003] J. Dichy et A. Farghaly, Roots & Patterns vs. Stems plus Grammar-Lexis Specifications: on what basis should a multilingual lexical database centred on Arabic be built?, *Actes de la 9ème MT Conference, Workshop on Machine Translation for Semitic Languages: issues and approaches, New Orleans, Louisiana, USA,* (2003).

[Dichy, 1990] J. Dichy, L'Ecriture dans la représentation de la langue : la lettre et le mot en arabe, *Thèse pour le doctorat d'état, Lyon: Université Lumière-Lyon 2,* (1990).

[Dichy, 1997] J. Dichy, Pour une lexicomatique de l'arabe: l'unité lexicale simple de l'inventaire du mot. *META - journal de traduction, Vol. 42, n° 2, pp. 291-306,* (1997).

[Dichy, 2001] J. Dichy, On lemmatization in Arabic : A formal definition of the Arabic Entries of multilingual lexical databases, *Actes du Workshop the Arabic Language Processing, ACL Workshop, Toulouse,* (2001).

[Fairon, 2001] C. Fairon, GlossaNet en-ligne : service de concordances automatique, *CENTAL, Université Catholique de Louvain,* 2001.

[Friburger et Maurel, 2004] N. Friburger et D. Maurel, Finite-state transducer cascades to extract named entities in texts, *Theoretical Computer Science, vol. 313,* (2004).

[Friburger, 2002] N. Friburger, Reconnaissance automatique des noms propres: Application à la classification automatique de textes journalistiques, *Thèse de doctorat en Informatique, Université de Tours,* (2002).

[Gal, 2002] Y. Gal, An HMM Approach to Vowel Restoration in Arabic and Hebrew, *Actes du workshop on Computational Approaches to Semitic Languages au meeting annuel de l'Association for Computational Linguistics (ACL-02), Philadelphie, USA,* (2002).

[Garrigues, 1999] M. Garrigues, Nouvelles concordances pour l'enseignement des langues, *Linguisticae Investigationes, Tome XXII, pp. 59-70,* (1999).

[Graña et al., 2001] J. Graña, F.M. Barcala, M.A. Alonso, Compilation methods of minimal acyclic finite-state automata for large dictionaries, *in: B.W. Watson, D. Wood (Eds.), Proceedings of CIAA'2001, Pretoria, South Africa,* (2001), pp. 116–129.

[Greaves, 2003] C. Greaves, ConcApp, *Public Version,* (2003).

[Gross, 1997] M. Gross, The Construction of Local Grammars. *Finite-State Language Processing, The MIT Press,* (1997).

[Habash, 2004] N. Habash, Large Scale Lexeme Based Arabic Morphological Generation, *Actes de la conferences JEP-TALN'04, Session Traitement Automatique de l'Arabe, Fès, Maroc,* (2004).

[Hajič et al., 2005] J. Hajič, O. Smrž, T. Buckwalter et H. Jin, Feature-Based Tagger of Approximations of Functional Arabic Morphology, *Actes de la quatrième conférence sur les Treebanks et les théories linguistiques, Université de Barcelona,* (2005).

[Hopcroft, 1971] J. E. Hopcroft, An (n log(n)) algorithm for minimizing the states in a finite automaton. *The Theory of Machines and Computations,Academic Press,* (1971), pp 189-196.

[Huffman, 1954] D. A. Huffman, The synthesis of sequential switching circuits. *Journal of Franklin Institute, 257,* (1954), pp 161-190.

[Ibrahim, 2002] K. Ibrahim, Al-Murshid fi Qawa'id Al-Nahw wa Al-Sarf [Le guide des règles de syntaxe et morphologie], *Amman, Jordan: Al-Ahliyyah for Publishing and Distribution,* (2002).

[Johnson, 1972] C. D. Johnson, "Formal Aspects of Phonological Description", *Mouton, The Hague,* (1972).

[Kadri et Benyamina, 1992] Y. Kadri et A. Benyamina, Un système d'analyse syntaxico-sémantique du langage arabe non voyellé, *Mémoire d'ingénieur, Université d'Oran,* (1992).

[Karttunen, 1995] L. Karttunen, The replace operator, *Actes du Annual Meeting of the Association for Computational Linguistics,* (1995).

[Khoja et al., 2001] S. Khoja, R. Garside, G. Knowles, A tagset for the morpho-syntactic tagging of arabic, *Actes de la conférence internationale Corpus Linguistics 2001, Lancaster,* (2001).

[Koskenniemi, 1983] K. Koskenniemi, Two-level Morphology: A General Computational Model for Word-Form Recognition and Production, *Publications N° 6, Université de Helsinki, Department of General Linguistics,* (1983).

[Kouloughli, 1991] D.E. Kouloughli, Lexique fondamental de l'arabe standard moderne, *Paris: l'Harmattan, 287 p.,* (1991).

[Kouloughli, 1994] D.E. Kouloughli, Grammaire de l'arabe d'aujourd'hui, *Collection "Pocket-Langues pour tous", 350 p,* (1994).

[Kowaltowski et al., 1993] T. Kowaltowski, C. L. Lucchesi and J. Stolfi, Minimization of binary automata. *Actes du premier South American String Processing Workshop, Belo Horizonte, Brasil,* (1993)

[Larcher, 2003] P. Larcher, Le système verbal de l'arabe classique, *Aix-en-Provence : Publications de l'Université de Provence, Collection Didactilangue, 194 p.,* (2003).

[Maamouri, 2006] M. Maamouri, Diacritization: A Challenge to Arabic Treebank Annotation and Parsing, *Actes de la conférence internationale 'The Challenge of Arabic for NLP/MT', The British Computer Society, London,* (2006).

[MacDonald, 1996] M. MacDonald, Internal and external evidence in the identification and semantic categorisation of Proper Names, *Corpus Processing for Lexical Acquisition, Massachussetts Institute of Technology,* (1996)

[Maloney, 1998] J. Maloney et M. Niv, TAGARAB: A fast, accurate Arabic name recognizer using high-precision morphological analysis, *Actes de la conférence internationale COLING-ACL Workshop on Computational Approaches to Semitic Languages,* (1998).

[Maurel, 1990] D. Maurel, Adverbes de date: Étude préliminaire à leur traitement automatique. *Lingvisticæ Investigationes 14:1,* (1990).

[Maurel, 2008] D. Maurel, Prolexbase. A multilingual relational lexical database of proper names, *Actes de la 6ème conférence internationale Language Resources and Evaluation Conference (LREC 2008), Marrakech, Maroc,* (2008).

[McCarthy, 1985] J. McCarthy, Formal Problems in Semitic Phonology and Morphology, *Outstanding Dissertations in Linguistics Series, Garland Publishing, New York, 430 p,* (1985).

[Mihov et Maurel, 2000] S. Mihov , D. Maurel, Direct Construction of Minimal Acyclic Subsequential Transducers, *in: D. Wood, S. Yu (Eds.), Proceedings of CIAA'2000 conference, London, Canada,* (2000), pp. 217-229.

[Mihov, 1999] S. Mihov, Direct building of minimal automaton for given list, *Thèse de doctorat, Bulgarian Academy of Science,* (1999).

[Mikheev et al., 1999] A. Mikheev, M. Moens, C. Grover, Named Entity Recognition without Gazetteers, *Actes de the Ninth Conference of the European Chapter of the Association for Computational Linguistics, Bergen, Norway,* (1999).

[Moore, 1956] E. F. Moore, Gedanken experiments on sequential machines. *In Automata Studies, Annals of Mathematical Studies, Princeton University Press, 34,* (1956).

[Nelken et Shieber, 2005] R. Nelken and S. M. Shieber, Arabic diacritization using weighed finite-state transducers", *Actes du Workshop on Computational Approaches to Semitic Languages. Association for Computational Linguistics ACL, Michigan, USA,* (2005).

[Neyreneuf et Hakkâk, 1996] M. Neyreneuf et Ghalib Hakkâk, Grammaire active de l'arabe littéral, *Editions LGF,* (1996).

[Ouersighni, 2002] R. Ouersighni, La conception et la réalisation d'un système d'analyse morpho-syntaxique robuste pour l'arabe : Utilisation pour la détection et le diagnostic des fautes d'accord, *Thèse de doctorat en Sciences de l'information et de la communication, Université Lumière-Lyon2,* (2002).

[Piton et al., 1996] O. Piton, M. C. Taieb, D. Maurel, L'importance du traitement informatique des noms propres ; exemple des noms de pays et de leurs dérivés, *Actes du 15ème Colloque Européen sur la Grammaire et le Lexique Comparés des Langues Romanes, Munich, Allemagne,* (1996).

[Piton et Maurel, 2004] O. Piton et D. Maurel, Les Noms Propres Géographiques et le Dictionnaire Prolintex, *Cahiers de la MSH Ledoux, 'Série Archive, Bases, Corpus', n°1,* (2004).

[Poibeau, 1999] T. Poibeau, Le repérage des entités nommées, un enjeu pour les systèmes de veille, *Terminologies Nouvelles : actes du colloque Terminologie et Intelligence Artificielle, TIA'99, Nantes,* (1999).

[Poibeau, 2001]T. Poibeau, Extraction d'information dans les bases de données textuelles en génomique au moyen de transducteurs à nombre fini d'états, *Actes de la 8 ème conférence nationale sur le Traitement Automatique des Langues Naturelles (TALN'2001), Tours,* (2001).

[Poibeau, 2003] T. Poibeau, Extraction d'information, du texte brut au web sémantique, *Editions Hermès,* (2003).

[Poibeau, 2005]T. Poibeau, Sur le statut référentiel des entités nommées. *Actes de la conférence Traitement Automatique des Langues Naturelles (TALN 2005), Dourdan, France,* (2005).

[Ratcliffe, 1998] R. R. Ratcliffe, The Broken Plural Problem in Arabic and Comparative Semitic : Allomorphy and Analogy in Non-concatenative Morphology. *Amsterdam ; Philadelphia: J. Benjamins,* (1998).

[Renouf, 2002] A. Renouf, et A. Kehoe, WebCorp: Applying the Web to Linguistics and Linguistics to the Web. *World Wide Web 2002 Conference, Honolulu, Hawaii,* (2002).

[Revuz, 1991] D. Revuz, Dictionnaires et lexiques: méthodes et algorithmes, *Thèse de doctorat, Institut Blaise Pascal, Paris, France,* (1991).

[Revuz, 1992] D. Revuz, Minimisation of acyclic deterministic automata in linear time, *Theorical computer science,* (1992), pp. 181–189.

[Roberts et al, 2006] A. Roberts, L. Al-Sulaiti et E. Atwell, aConCorde: Towards an open-source, extendable concordancer for Arabic», *Dans: McEnery, Tony et al. (eds.) Corpora Journal, Edinburgh University Press,* (2006).

[Rodriguez, 1999] M. Rodriguez, "Using a concordancer in literary studies", *Université Jaume I, Castellón, The European English Messenger Vol VII/2,* (1999).

[Sabah, 1989] G. Sbah, L'intelligence artificielle et le langage, Volume 2 (processus de compréhension), *Hemès, Paris,* (1989).

[Safadi et al. , 2006] H. Safadi, O. Dakkak et N. Ghneim, Computational Methods to Vocalize Arabic Texts, *Actes du 2ème "Workshop on Internationalizing SSML", Crète, Grèce,* (2006).

[Scott, 2008] M. Scott, Introduction to WordSmith Tools : online manual, (2008).

[Senellart, 1998] J. Senellart, Locating Noun Phrases with Finite State Transducers, *Actes de la conference internationale COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics, Université de Montréal, Quebec, Canada,* (1998).

[Silberztein et Tutin, 2004] M. Silberztein et A. Tutin, NooJ : Un outil TAL de corpus pour l'enseignement des langues et de la linguistique, *Journée ATALA TAL et apprentissage des langues,* (2004).

[Silberztein, 1991]M. Silberztein, A new approach to tagging: the use of a large-coverage electronic dictionary, *Applied Computer Translation,* (1991).

[Silberztein, 1993] M. Silberztein, Dictionnaires électroniques et analyse de textes : le système INTEX. *Masson : Paris* (1993).

[Silberztein, 1999a] M. Silberztein, Transducteurs pour le traitement automatique des textes. *Rapport technique 57. LADL, Paris,* (1999).

[Silberztein, 1999b] M. Silberztein, INTEX: a Finite State Transducer Toolbox. *Theorical Computer Science. Vol. 231 :1, p 33-46,* (1999).

[Silberztein, 2000] M. Silberztein, Traitement des expressions figées avec INTEX. *Fairon Cédrick (ed). Linguisticae Investigationes,* (2000).

[Silberztein, 2003] M. Silberztein, Finite-State Description of the French Determiner system. *Journal of French Language Studies. Cambridge University Press,* (2003).

[Silberztein, 2004] Silberztein, M. (2004). NooJ : an oriented object approach. *In Royauté, J. & Silberztein, M. (dir.) INTEX pour la Linguistique et le Traitement Automatique des Langues. Actes des 4èmes et 5èmes journées INTEX, Bordeaux, mai 2001 et Marseille, mai 2002. Besançon : Presses universitaires de Franche-Comté*,(2004).

[Silberztein, 2005a] M. Silberztein, NooJ : The Lexical Module In NooJ pour le Traitement Automatique des Langues, *S. Koeva, D. Maurel, M. Silberztein Eds, Cahiers de la MSH Ledoux. Presses Universitaires de Franche-Comté,* (2005).

[Silberztein, 2005b] M. Silberztein, NooJ's dictionaries. *Actes de la conférence internationale LTC 2005, Poznan, Pologne,* (2005).

[Tounsi, 2007] L. Tounsi, Sous-automates à nombre fini d'états : Application à la compression de dictionnaires électroniques, *Thèse de doctorat en informatique, Université François Rabelais, Tours, France,* (2007).

[Tran, 2006] M. Tran, Prolexbase : Un dictionnaire relationnel multilingue de noms propres : conception, implémentation et gestion en ligne, *Thèse de doctorat en Informatique, Université de Tours,* (2002).

[Tuerlinckx, 2004] L. Tuerlinckx, La lemmatisation de l'arabe non classique, *7èmes Journées internationales d'Analyse statistique des Données Textuelles, JADT'04, Louvain-la-Neuve, Belgique,* (2004).

[Watson, 1998] B.W. Watson, A fast new semi-incremental algorithm for the construction of minimal acyclic DFAs, *in: D. Wood, D. Maurel (Eds.), Proceedings of CIAA'98 conference, Rouen, France,* (1998), pp. 91–98.

[Watson, 2001] B.W. Watson, A taxonomy of algorithms for constructing minimal acyclic deterministic finite automata, *South African Comput. J. 27* (2001) 12–17.

[Watson, 2003] B.W. Watson, A new algorithm for construction of minimal acyclic DFAs, *Science of Computer Programming* (2003), pp. 81-97.

[Wehr, 1979] H. Wehr, A Dictionary of Modern Written Arabic. *Arabic-English, Wiesbaden, Harrassowitz, 1110 p.,* (1979).

[Woods, 1970] W. A. Woods, Transition network grammars for natural language analysis, *Communications of the ACM, v.13 n.10,* (1970).

[Yona et Wintner, 2005] S. Yona et S. Wintner, "A finite-state morphological grammar of Hebrew", *Actes du ACL Workshop on Computational Approaches to Semitic Languages, Association for Computational Linguistics,* (2005).

[Zaafrani, 2001] R. Zaafrani, Développement d'un environnement interactif d'apprentissage avec l'ordinateur de l'arabe langue étrangère, *Thèse de doctorat en Science de l'information et de la communication, Université Lumière - Lyon 2,* (2001).

Publications et Communications

S. Mesfar, M. Silberztein, Transducer minimization and information compression for NooJ dictionaries, *Actes de la 11ème conférence internationale Finite-State Methods and Natural Language Processing - FSMNLP 2008, Joint Research Centre of the EC, Ispra, Italy,* Septembre 2008.

S. Mesfar, Corpus Linguistics with NooJ, *Tutoriel présenté aux 11èmes journées NooJ 2008, Académie Hongroise des Sciences, Budapest, Hongrie,* Juin 2008.

S. Mesfar, Morphological grammars for standard Arabic tokenization, *Actes des 11èmes journées NooJ 2008, Académie Hongroise des Sciences, Budapest, Hongrie,* Juin 2008. *Ed. Cambridge Schooling Press* (à paraître).

S. Mesfar, M. Silberztein, Ingénierie linguistique avec NooJ, *Séminaire doctoral, Université Paris12 – Val de Marne*, Mai 2008.

S. Mesfar, Electronic dictionaries for the automatic analysis of texts : form Intex to NooJ, *Communication au 2ème Workshop des Experts des dictionnaires électroniques de l'Arabe, Cité du Roi Abdel-Aziz des Sciences et Technologies, Riyad, Arabie Saoudite*, Avril 2008.

S. Mesfar, M. Silberztein, NooJ4Web: un système de concordances pour la veille médicale, *Grand Colloque STIC 2007, centre des conférences, Cité des Sciences de La Villette, Paris,* Novembre 2007.

S. Mesfar, M. Silberztein, T. Varadi et K. Gabor, Corpus querying with NooJ, *Workshop NooJ : A sophisticated finite-state linguistic analysis tool for corpora, Conférence internationale Corpus Linguistics CL2007, à l'Université de Birmingham, Angleterre,* Juillet 2007.

S. Mesfar, NooJ4Web: an on-line concordance service, *Actes des 10èmes journées NooJ 2007 à l'Université Autonome de Barcelone, Espagne, Juin 2007. Ed. Cambridge Schooling Press* (à paraître).

S. Mesfar, Named Entity Recognition for Arabic using syntactic grammars, *Actes de la 12ème conférence internationale NLDB 2007, CNAM, Paris, France, Juin 2007. Ed. LNCS Series, Springer Verlag*.

S. Mesfar, Des transducteurs finis pour la reconnaissance des noms propres en arabe, *Communication au 7èmes journées scientifiques des jeunes chercheurs en génie électrique et informatique GEI'07, Monastir, Tunisie*, Mars 2006.

S. Mesfar, NooJ pour la reconnaissance des entités nommées en arabe standard, *Communication et tutorial présentés au workshop "NooJ : un moteur de recherche linguistique", Université de Sfax, Tunisie,* Novembre 2006.

S. Mesfar, NooJ pour la traduction, *Tutoriel présenté au workshop "NooJ : outils pour la traduction automatique", Université Paris 1, France*, Septembre 2006.

S. Mesfar, Standard Arabic formalization and linguistic platform for its analysis, *Actes de la Conférence "The challenge of Arabic NLP/MT conference", Londres – Angleterre*, Octobre2006. *Eds BCS - British Computeur Society*

S. Mesfar, Reconnaissance des entités nommées en arabe standard, *Communication aux 9èmes journées INTEX/NooJ à l'Université de Belgrade, Serbie,* Juin 2006.

S. Mesfar, Analyse lexicale et morphologique de l'arabe standard utilisant la plateforme linguistique NooJ, *Actes de la 13ème Conférence sur le Traitement Automatique des Langues Naturelles RECITAL/TALN 2006, Presses universitaires de Louvain, Louvain-la-Neuve, Belgique,* Avril 2006

S. Mesfar, Analyse lexicale et morphologique de l'arabe standard utilisant la plateforme linguistique NooJ, *Présentation à la journée Jeunes chercheurs en linguistique appliquée, Association Française de Linguistique Appliquée AFLA2005, Université Paris3 – Sorbonne Nouvelle,* Octobre 2005

S. Mesfar, Analyse lexicale et morphologique de l'arabe standard utilisant la plateforme linguistique NooJ, *Actes du 9ème Atelier des Doctorants en Linguistique ADL2005 à l'Université Paris7, Jussieu,* Octobre 2005

S. Mesfar, Lexique arabe et analyse morphologique, *Actes des 8èmes journées INTEX/NooJ à l'Université de Franche Comté, Besançon,* Mai 2005.

S. Mesfar, VOWEL : outil interactif pour la voyellation de textes arabes, *Mémoire de projet de fin d'études d'ingénieurs, Manouba, Tunisie,* (2003).

Résumé

Résumé :

La langue arabe, bien que très importante par son nombre de locuteurs, elle présente des phénomènes morpho-syntaxiques très particuliers. Cette particularité est liée principalement à sa morphologie flexionnelle et agglutinante, à l'absence des voyelles dans les textes écrits courants, et à la multiplicité de ses formes, et cela induit une forte ambiguïté lexicale et syntaxique. Il s'ensuit des difficultés de traitement automatique qui sont considérables. Le choix d'un environnement linguistique fournissant des outils puissants et la possibilité d'améliorer les performances selon nos besoins spécifiques nous ont conduit à utiliser la plateforme linguistique NooJ.

Nous commençons par une étude suivie d'une formalisation à large couverture du vocabulaire de l'arabe. Le lexique construit, nommé «El-DicAr», permet de rattacher l'ensemble des informations flexionnelles, morphologiques, syntactico-sémantiques à la liste des lemmes. Les routines de flexion et dérivation automatique à partir de cette liste produisent plus de 3 millions de formes fléchies. Nous proposons un nouveau compilateur de machines à états finis en vue de pouvoir stocker la liste générée de façon optimale par le biais d'un algorithme de minimisation séquentielle et d'une routine de compression dynamique des informations stockées. Ce dictionnaire joue le rôle de moteur linguistique pour l'analyseur morpho-syntaxique automatique que nous avons implanté. Cet analyseur inclut un ensemble d'outils: un analyseur morphologique pour le découpage des formes agglutinées en morphèmes à l'aide de grammaires morphologiques à large couverture, un nouvel algorithme de parcours des transducteurs à états finis afin de traiter les textes écrits en arabe indépendamment de leurs états de voyellation, un correcteur des erreurs typographiques les plus fréquentes, un outil de reconnaissance des entités nommées fondé sur une combinaison des résultats de l'analyse morphologique et de règles décrites dans des grammaires locales présentées sous forme de réseaux augmentés de transitions (ATNs), ainsi qu'un annotateur automatique et des outils pour la recherche linguistique et l'exploration contextuelle.

Dans le but de mettre notre travail à la disposition de la communauté scientifique, nous avons développé un service de concordances en ligne «NooJ4Web: NooJ pour la Toile» permettant de fournir des résultats instantanés à différents types de requêtes et d'afficher des rapports statistiques ainsi que les histogrammes correspondants. Les services ci-dessus cités sont offerts afin de recueillir les réactions des divers usagers en vue d'une amélioration des performances. Ce système est utilisable aussi bien pour traiter l'arabe, que le français et l'anglais.

Abstract:

The Arabic language, although very important by the number of its speakers, it presents special morpho-syntactic phenomena. This particularity is mainly related to the inflectional and agglutinative morphology, the lack of vowels in currents written texts, and the multiplicity of its forms; this induces a high level of lexical and syntactic ambiguity. It follows considerable difficulties for the automatic processing. The selection of a linguistic environment providing powerful tools and the ability to improve performance according to our needs has led us to use the platform language NooJ.

We begin with a study followed by a large-coverage formalization of the Arabic lexicon. The built dictionary, baptised "El-DicAr" allows to link all the inflexional, morphological, syntactico-semantic information to the list of lemmas. Automatic inflexional and derivational routines applied to this list produce more than 3 million inflected forms. We propose a new finite state machine compiler that leads to an optimal storage through a combination of a sequential minimization algorithm and a dynamic compression routine for stored information. This dictionary acts as the linguistic engine for the automatic morpho-syntactic analyzer that we have developed. This analyzer includes a set of tools: a morphological analyzer that identifies the component morphemes of agglutinative forms using large coverage morphological grammars, a new algorithm for looking

through finite-state transducers in order to deal with texts written in Arabic with regardless of their vocalisation statements, a corrector of the most frequent typographical errors, a named entities recognition tool based on a combination of the morphological analysis results and rules described into local grammar presented as Augmented Transition Networks (ATNS), an automatic annotator and some tools for linguistic research and contextual exploration.

In order to make our work available to the scientific community, we have developed an online concordance service “NooJ4Web: NooJ for the Web”. It provides instant results to different types of queries and displays statistical reports as well as the corresponding histograms. The listed services are offered in order to collect feedbacks and improve performance. This system is used to process Arabic, as well as French and English.

Titre :

Analyse morpho-syntaxique automatique et reconnaissance des entités nommées en arabe standard

Résumé :

La langue arabe, bien que très importante par son nombre de locuteurs, elle présente des phénomènes morpho-syntaxiques très particuliers. Cette particularité est liée principalement à sa morphologie flexionnelle et agglutinante, à l'absence des voyelles dans les textes écrits courants, et à la multiplicité de ses formes, et cela induit une forte ambiguïté lexicale et syntaxique. Il s'ensuit des difficultés de traitement automatique qui sont considérables. Le choix d'un environnement linguistique fournissant des outils puissants et la possibilité d'améliorer les performances selon nos besoins spécifiques nous ont conduit à utiliser la plateforme linguistique NooJ.

Nous commençons par une étude suivie d'une formalisation à large couverture du vocabulaire de l'arabe. Le lexique construit, nommé «El-DicAr», permet de rattacher l'ensemble des informations flexionnelles, morphologiques, syntactico-sémantiques à la liste des lemmes. Les routines de flexion et dérivation automatique à partir de cette liste produisent plus de 3 millions de formes fléchies. Nous proposons un nouveau compilateur de machines à états finis en vue de pouvoir stocker la liste générée de façon optimale par le biais d'un algorithme de minimisation séquentielle et d'une routine de compression dynamique des informations stockées. Ce dictionnaire joue le rôle de moteur linguistique pour l'analyseur morpho-syntaxique automatique que nous avons implanté. Cet analyseur inclut un ensemble d'outils: un analyseur morphologique pour le découpage des formes agglutinées en morphèmes à l'aide de grammaires morphologiques à large couverture, un nouvel algorithme de parcours des transducteurs à états finis afin de traiter les textes écrits en arabe indépendamment de leurs états de voyellation, un correcteur des erreurs typographiques les plus fréquentes, un outil de reconnaissance des entités nommées fondé sur une combinaison des résultats de l'analyse morphologique et de règles décrites dans des grammaires locales présentées sous forme de réseaux augmentés de transitions (ATNs), ainsi qu'un annotateur automatique et des outils pour la recherche linguistique et l'exploration contextuelle.

Dans le but de mettre notre travail à la disposition de la communauté scientifique, nous avons développé un service de concordances en ligne «NooJ4Web: NooJ pour la Toile» permettant de fournir des résultats instantanés à différents types de requêtes et d'afficher des rapports statistiques ainsi que les histogrammes correspondants. Les services ci-dessus cités sont offerts afin de recueillir les réactions des divers usagers en vue d'une amélioration des performances. Ce système est utilisable aussi bien pour traiter l'arabe, que le français et l'anglais.

Title:

Automatic morpho-syntactic analyzer and named entities recognition for standard Arabic

Abstract:

The Arabic language, although very important by the number of its speakers, it presents special morpho-syntactic phenomena. This particularity is mainly related to the inflectional and agglutinative morphology, the lack of vowels in currents written texts, and the multiplicity of its forms; this induces a high level of lexical and syntactic ambiguity. It follows considerable difficulties for the automatic processing. The selection of a linguistic environment providing powerful tools and the ability to improve performance according to our needs has led us to use the platform language NooJ.

We begin with a study followed by a large-coverage formalization of the Arabic lexicon. The built dictionary, baptised "El-DicAr" allows to link all the inflexional, morphological, syntactico-semantic information to the list of lemmas. Automatic inflexional and derivational routines applied to this list produce more than 3 million inflected forms. We propose a new finite state machine compiler that leads to an optimal storage through a combination of a sequential minimization algorithm and a dynamic compression routine for stored information. This dictionary acts as the linguistic engine for the automatic morpho-syntactic analyzer that we have developed. This analyzer includes a set of tools: a morphological analyzer that identifies the component morphemes of agglutinative forms using large coverage morphological grammars, a new algorithm for looking through finite-state transducers in order to deal with texts written in Arabic with regardless of their vocalisation statements, a corrector of the most frequent typographical errors, a named entities recognition tool based on a combination of the morphological analysis results and rules described into local grammar presented as Augmented Transition Networks (ATNS), an automatic annotator and some tools for linguistic research and contextual exploration.

In order to make our work available to the scientific community, we have developed an online concordance service "NooJ4Web: NooJ for the Web". It provides instant results to different types of queries and displays statistical reports as well as the corresponding histograms. The listed services are offered in order to collect feedbacks and improve performance. This system is used to process Arabic, as well as French and English.