Applications du dictionnaire électronique des séquences nominales figées et de leurs formes fléchies

Dans ce chapitre, nous présentons une méthode de génération des formes fléchies des séquences nominales figées, puis nous appliquons notre dictionnaire et nous examinons ses avantages dans plusieurs domaines du traitement automatique des langues naturelles.

Dans la section 1, à partir de codes flexionnels comme N11 ou N33 du dictionnaire électronique et de la fonctionnalité de flexion du menu DELA d'INTEX (« Inflection »), nous engendrons toutes les formes fléchies des séquences nominales figées. Nous fusionnons deux bases de données : les séquences nominales figées et les séquences de postpositions nominales. Dans la section 2, nous justifions la validité de notre méthode en appliquant notre dictionnaire des formes fléchies des noms figés à espacement facultatif (NFF) à des textes coréens. Dans la section 3, nous examinons l'application de notre dictionnaire électronique dans plusieurs domaines du traitement automatique des langues naturelles comme la recherche d'informations, l'extraction de mots-clés, l'analyse syntaxique et la traduction automatique.

1. Construction automatique du dictionnaire des formes fléchies des séquences nominales figées avec INTEX

1-1. Génération automatique des formes fléchies des séquences nominales figées

Avant d'aborder l'étape de la génération automatique des formes fléchies des séquences nominales figées du coréen, nous présentons brièvement la méthode de production des formes fléchies des mots simples en français : il s'agit de la méthode mise au point au LADL par B. Courtois (1990) et M. Silberztein (1993). DELAS est une liste de tous les mots simples sous leurs formes lemmatisées : par exemple, l'infinitif pour les verbes, le masculin singulier pour les adjectifs, etc. Chacun des mots simples est accompagné d'un code flexionnel qui permet d'engendrer automatiquement toutes les formes fléchies à partir de sa forme lemmatisée. Par exemple, pour l'entrée du DELAS français (M. Silberztein, 1996 et A. Chrobot, 2001) :

nouveau,A72

L'exemple ci-dessus représente le lemme *nouveau* qui est un adjectif. Voici le

transducteur de flexion A72 représenté par le graphe au format INTEX. Il est associé aux adjectifs du type *nouveau* :

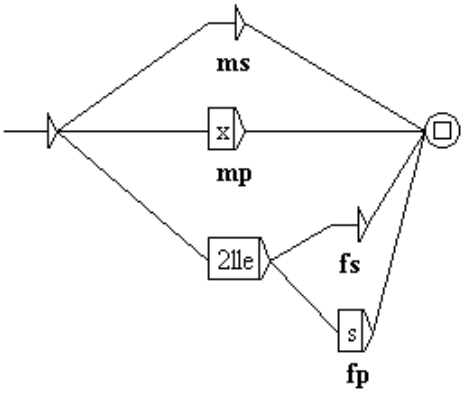


Figure 1. Transducteur de flexion A72

Puisque le masculin singulier est équivalent au lemme, le suffixe vide est ajouté au lemme. Mais pour produire le pluriel au masculin, il faut ajouter le suffixe « x ». Pour produire le féminin singulier, il faut effacer les deux dernières lettres du lemme et ajouter la terminaison « lle ».

Introduisons les opérateurs de pile dans INTEX. L'effacement du dernier caractère est représenté par l'opérateur « L » (*Left*) et on peut abréger une séquence d'opérateur d'effacement en indiquant simplement le nombre d'effacements : « 2 » est équivalent à « LL ». Si on efface les deux dernières lettres du lemme et qu'on ajoute « lles », on produit la forme du pluriel au féminin. Le transducteur de flexion A72 dans le dossier « Inflection » génère les quatre entées suivantes du DELAF :

nouveau,nouveau.A72:ms
nouvelle,nouveau.A72:fs
nouveaux,nouveau.A72:mp
nouvelles,nouveau.A72:fp

Les quatre formes fléchies sont associées au même lemme *nouveau* après une virgule. Les propriétés linguistiques suivent le point. Les deux points « : » introduisent les traits morphologiques des formes fléchies obtenues. INTEX contient deux autres opérateurs « R » (*Right*) et « C » (*Copy*) pour décrire plus facilement le changement systématique d'une voyelle dans un ensemble de mots de base qui ne partagent pas la même désinence : « R » permet de sauter une lettre vers la droite dans le lemme, et « C »

permet de la dupliquer. Par exemple, la flexion de la première ou de la troisième personne du singulier présent (*:P1s:P3s*) des verbes comme *acheter*, *geler* ou *mener* est représentée grâce à la commande suivante :

LLLLRèCe (ou de façon abrégée : ***4RèCe***)

En se plaçant à la fin du mot, il faut reculer de 4 lettres à gauche, aller d'une lettre à droite *R*, insérer la lettre *è*, recopier *C* la lettre courante *n*, et insérer la lettre *e* :

Opération	**Lemme**	**Pile**	**Résultat**
	mener^		mener
4	m^ener	ener	m
R	me^ner	ner	m
è	me^ner	ner	mè
C	men^er	er	mèn
e	men^er	er	mène

Ici, « L » correspond à l'opérateur de pile « empiler » (PUSH), « R » à l'opérateur de pile « dépiler » (POP) et « C » à l'opérateur de pile « dépiler et imprimer ». La flexion d'un dictionnaire DELAS est immédiate : chaque opérateur prend un temps constant ; la construction de chaque forme prend un temps proportionnel à longueur du suffixe du transducteur ; la flexion de chaque entrée prend un temps proportionnel au nombre de chemins du transducteur (M. Silberztein, 2000)[1].

Maintenant, revenons au problème de génération des formes fléchies des

[1] Pour le programme sur le processus de la commande, voir Silberztein (1996, p.85) :

```
char *inflect (char *lemma, char *commands) {
static int ic, ir, il ; static char result [512] ;
        strcpy (result, lemma);
        for (ic=0, ir=il=strlen (res); commands[ic];ic++) {
                switch (commands[ic]) {
                        case 'C': result [ir++]=lemma [il++];break;
                        case 'L': il--; ir--;break;
                        case 'R': il++;break;
                        default: result[ir++]=commands[ic];
                }
        }
        result[ir]='\0';
        return result;
}
```

séquences nominales figées du coréen dans INTEX. Dans le chapitre 3, nous avons examiné le codage de chaque entrée selon les contraintes combinatoires avec les postpositions nominales. Dans le chapitre 4, nous avons étudié la construction du dictionnaire des séquences de postpositions nominales sous forme de graphes. Pour construire un dictionnaire électronique exploitable, il existe une étape indispensable : la fusion de deux bases de données. En effet, des formes fléchies des séquences nominales figées apparaissent dans les textes coréens. A partir de codes flexionnels comme N11 ou N33 d'un dictionnaire électronique et de la fonctionnalité de flexion du menu DELA d'INTEX (« Inflection »), nous engendrons toutes les formes fléchies des séquences nominales figées. Par exemple :

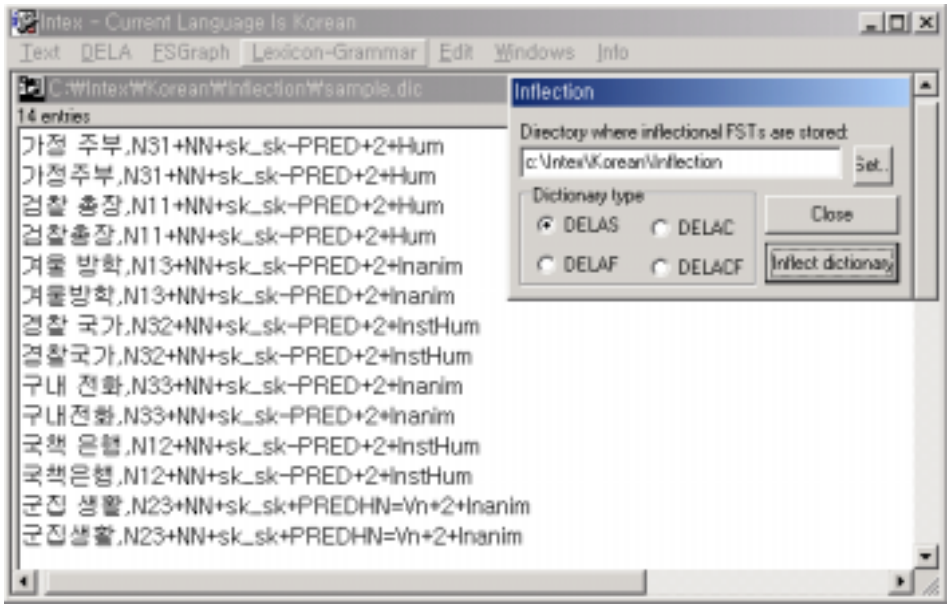


Figure 2. Dictionnaire du type DELAS pour la flexion

Cette fonctionnalité permet la flexion automatique en appelant des transducteurs de postpositions nominales. Le nom du transducteur est exactement le code associé à chaque entrée. Lorsqu'il s'agit de la flexion des séquences nominales du coréen, le module de flexion est suffisant pour engendrer tous les types de séquences nominales figées : les postpositions nominales ou les séquences de postpositions nominales se combinent toujours avec la dernière syllabe des séquences nominales figées, sans qu'aucun changement de forme n'intervienne au niveau de cette dernière syllabe.

Voici par exemple une entrée de nom figé à espacement facultatif et ses formes fléchies. Chaque exemple ci-dessous est comparable aux entrées du DELAC (1a) et du

DELACF ((1b) et (1c)) dans le système DELA (M. Silberztein, 1990).

(1) a. *gukchaek ^ eunhaeng*, N12+NN+sk_sk -PRED+2+InstHum
(politique-national - banque = banque nationale; Banque de France, Banque de Corée, etc.)

Dans (1a), le code « ^ » représente l'espacement facultatif et le code flexionnel N12 appelle le graphe de séquences de postpositions nominales « N12.grf ». À partir de « *gukchaek eunhaeng*, N12 », les formes fléchies ((1b) et (1c)) de (1a) sont automatiquement engendrées à l'aide du module de flexion d'INTEX.

b. *gukchaekeunhaeng**deulmaneun***, *gukchaek eunhaeng*.
N12+NN+sk_skPRED+2+InstHum:*_deul***Pl**_*man***Aux**_*neun***Aux**[2]
c. *gukchaek eunhaeng**deulmaneu**n*, *gukchaek eunhaeng*.
N12+NN+sk_sk–PRED+2+InstHum:*_deul***Pl**_*man***Aux**_*neun***Aux**

Dans (1b) et (1c), les formes *gukchaekeunhaengdeulmaneun* et *gukchaek eunhaengdeulmaneun* sont associées au même lemme *gukchaek eunhaeng* après une virgule. Les propriétés linguistiques suivent le point. Les deux points « : » introduisent les informations sur les postpositions nominales. Chaque postposition nominale est délimitée par « _ ». Les formes *deul, man, neun* en italique sont les formes canoniques des postpositions nominales. Chaque forme en gras **Pl** (morphème pluriel) et **Aux** (Auxiliaire) représente une classe linguistique de postpositions nominales[3].

1-2. Résultats

Avec les graphes des postpositions nominales, nous engendrons toutes les formes fléchies des séquences nominales figées. La figure 3 montre un échantillon très réduit des listes produites avec INTEX. La génération des formes fléchies des séquences nominales figées avec INTEX respecte en particulier l'usage coréen en ce qui concerne :

[2] Le codage après les deux points n'est pas tout à fait celui utilisé dans INTEX, mais les informations sont les mêmes. Pour faciliter l'exposition, nous marquons les formes canoniques en caractères coréens et les classes des postpositions nominales sous la forme des étiquettes *Postp*, *Aux*, etc. En fait, avec le logiciel INTEX chaque forme canonique accompagnée de l'étiquette d'une classe de postpositions nominales est représentée par un caractère ASCII. Pour ce codage, voir la note 19 dans le chapitre 4.

[3] Pour la classification linguistique, voir la section 3-1 du chapitre 4.

- les variantes phonologiques et leurs formes canoniques
- les variantes libres et leurs formes canoniques
- la contraction graphique des formes non syllabiques (cf. figure 12 dans le chapitre 4)

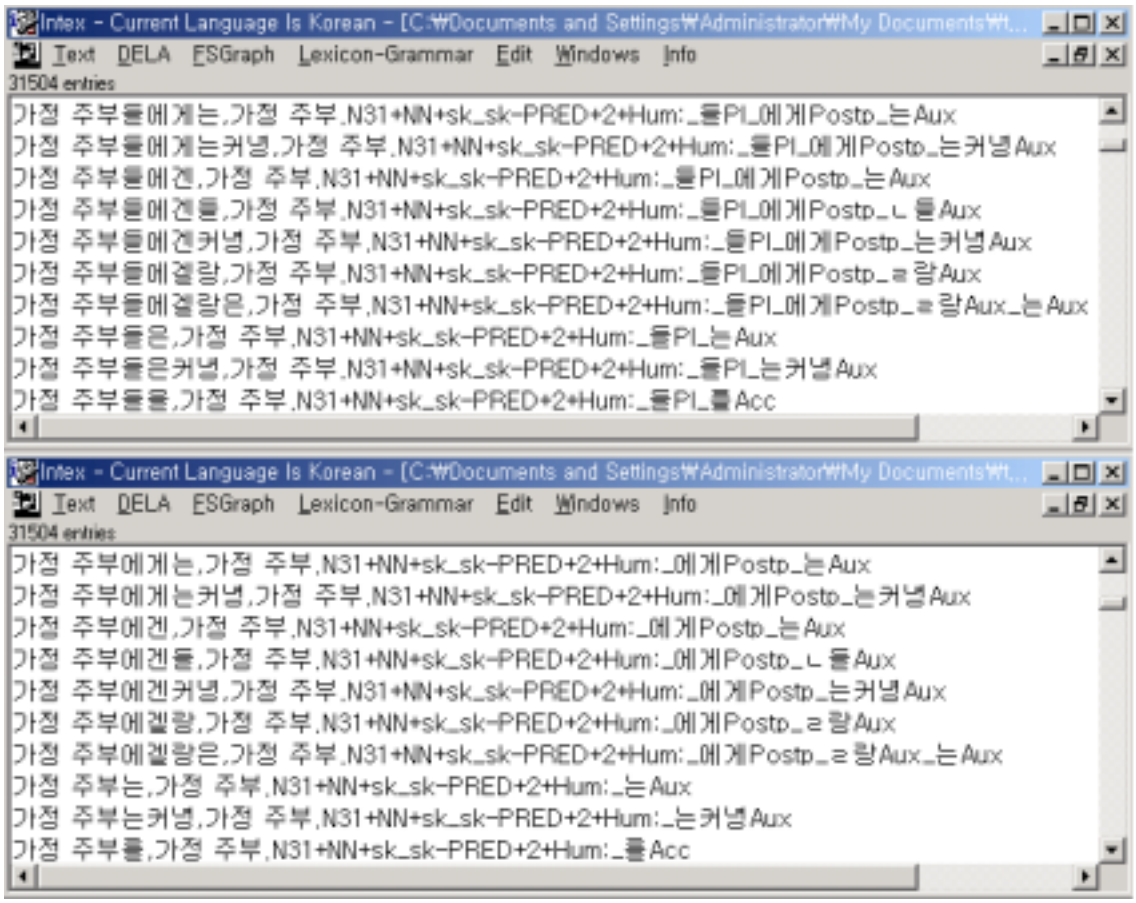


Figure 3. Résultat de la génération des formes fléchies des séquences nominales figées

Les quelques exemples ci-dessous illustrent ces trois difficultés.

(2) a. *gajeong jubu-deu**l-eun** :_deul***Pl**_neun***Aux***
b. *gajeong jub**u-neun** :_neun***Aux***
c. *gajeong jubu-deul-ege-**neun** :_deul***Pl**_ege***Postp**_neun***Aux***
d. *gajeong jubu-deul-ege-**n** :_deul***Pl**_ege***Postp**_neun***Aux***

Les variations phonologiques sont visibles si l'on se réfère aux formes canoniques. Par exemple, dans (2a) et (2b) nous voyons les deux variantes phonologiques de la

postposition nominale <-*neun*> : -*eun* après la consonne *l* et -*neun* après voyelle. Quant aux variantes libres, qui apparaissent dans un contexte phonologique identique (par exemple, -*neun* et -*n* se trouvent après une voyelle et sont des formes d'une même postposition auxiliaire <-*neun*>), nous leur faisons correspondre la même forme canonique, _*ege***Postp**_*neun***Aux** pour les séquences de postpositions nominales -*ege*-*neun* et -*ege*-*n* ((2c) et (2d)). Nous observons aussi une contraction graphique : la forme ***egen*** est une forme graphiquement modifiée à partir de -*ege* et de la forme non syllabique -*n*. La syllabe graphique constituée d'une consonne *n* devient la consonne finale de la syllabe précédente *ge* (cf. figure 12 dans le chapitre 4).

Nous pouvons faire une estimation du nombre de formes fléchies de certaines classes, domaines techniques non compris :

Séquences nominales figées		Séquences de *Postp*	Séquences nominales figées avec des *Postp*
NC Simples	15000	2300	$35x\ 10^6$
Autres NC	?	2300	?
NFF	45000 x 3	2300	311×10^6
NFO	?	2300	?

Figure 4. Estimation du nombre des formes fléchies des séquences nominales figées

Nam Jee-Sun (1994) a recensé 15000 noms compacts simples. Nous avons recensé 45000 noms figés à espacement facultatif. Ce nombre d'entrées doit être multiplié par 3, le nombre moyen de variantes liées à l'espacement. Par exemple, pour un nom figé à espacement facultatif constitué de deux noms (*NN*) le nombre de variantes est de deux. Pour trois noms (*NNN*) il est de quatre, pour quatre noms (*NNNN*) de huit, etc. Il existe des contraintes combinatoires entre séquences nominales figées et postpositions nominales. Le nombre moyen de séquences de postpositions pour un nom est environ 2300. Le nombre des séquences des postpositions pour un nom dépend du nom (cf. la section 2-2 du chapitre 3 et la section 3-3-2 du chapitre 4).

Calculons plus précisément le nombre des formes fléchies des NFF de

notre dictionnaire :

		Nombre de NFF		Nombre des variantes selon l'espacement	Nombre de formes fléchies
NN	N11, N21, N31	2087	42487	2087 x 2	2087 x 2 x 3400 = 14191600
	N12, N22, N32	1239		1239 x 2	1239 x 2 x 1650 = 4088700
	N13, N23, N33	39161		39161 x 2	39161 x 2 x 1900 = 148811800
NNN	N11, N21, N31	148	3277	148 x 4	148 x 4 x 3400 = 2012800
	N12, N22, N32	342		342 x 4	342 x 4 x 1650 = 2257200
	N13, N23, N33	2787		2787 x 4	2787 x 4 x 1900 = 21181200
NNNN	N11, N21, N31	4	202	4 x 8	4 x 8 x 3400 = 108800
	N12, N22, N32	20		20 x 8	20 x 8 x 1650 = 264000
	N13, N23, N33	178		178 x 8	178 x 8 x 1900 = 2705600
NNNNN	N32	2	7	2 x 16	2 x 16 x 1650 = 52800
	N13, N33	5		5 x 16	5 x 16 x 1900 = 152000
NNNNNN	N13	2	2	2 x 32	2 x 32 x 1900 = 121600
Total		45975		99674	195948100 (≅ 200 x 10^6)

Figure 5. Nombre de NFF et de leurs formes fléchies[4]

Le nombre d'entrées doit être multiplié par le nombre d'espaces et de séquences de postposition nominales : par exemple, 2087 (entrées) x 2 (variantes liées à l'espacement) x 3400 (séquences de postpositions nominales) indique le nombre des formes fléchies de NFF constitué de deux noms (*NN*) appartenant aux classes N11, N21 et N31.

1-3. Compression du DELAF et du DELACF de NFF en FST déterministes minimaux

Une fois construit sous la forme d'un DELAF d'INTEX, le dictionnaire de toutes les formes fléchies des NFF, il est placé dans le dossier DELACF de la langue coréenne avec l'extension « .dic ». Or, nous obtenons deux types de formes fléchies de NFF en coréen : les formes sans aucun espace et celles avec au moins un espace. Nous pouvons

[4] Par exemple, les NFF des classes N11, N21 ou N31 prennent le même nombre de séquences de postpositions nominales.

tout mettre dans le dossier DELACF, mais nous avons mis les formes fléchies des NFF sans espace dans le dossier DELAF et celles des NFF avec espace dans le dossier DELACF pour faciliter l'exposition (cf. nous allons comparer le nombre de NFF reconnus par DELAF et celui par DELACF (figure 15, p. 162).

Mais, comme nous l'avons vu dans la figure 4, le nombre des formes fléchies des NFF est 200 x 10^6 et la taille du fichier qui est environ 12 Go est trop énorme pour que le dictionnaire soit utilisé directement dans INTEX. Il est avantageux de convertir ce dictionnaire en transducteur, dont la taille typique est de l'ordre de quelques Mo[5]. INTEX construit automatiquement le transducteur équivalent au dictionnaire DELAF ou DELACF à l'aide du module « Compress into FST » du menu **DELA**. Le processus de compression dans INTEX consiste en deux phases[6] :

(i) La liste textuelle d'entrées du DELAF est convertie en un arbre lexicographique (ou un cas particulier d'automate déterministe acyclique[7]), où l'état initial est la racine de l'arbre et les états terminaux sont des feuilles. La figure 7 présente un arbre lexicographique construit pour un dictionnaire contenant six mots anglais : *ant*, *ants*, *apse*, *apses*, *pat* et *pats*. Le langage reconnu[8] par l'automate de la figure 7 est : L(A)[9]= {*ant*, *ants*, *apse*, *apses*, *pat*,

---

[5] Par exemple, le DELAF français qui contient environ 700 000 formes a une taille de plus de 30 Mo et le DELAF russe qui contient 3,5 millions de formes a une taille de plus de 100 Mo (M. Silberztein, 2000, p.161).

[6] (...) *compact* enters each line from *stdin* into a deterministic finite state automaton, and then minimize it. The minimization process is in O (number of states), because the automaton has no cycle (M. Silberztein1996, p. 86).

[7] Un *automate fini* A est défini par la donnée d'un quintuplet d'ensembles (B,Q,I,T,F) (D.Revuz, 1991) :

- un alphabet fini B;
- un ensemble fini d'états noté Q ;
- une partie I de Q d'états initiaux ;
- une partie T de Q d'états terminaux ;
- un ensemble fini F de triplets (**p**,**a**,**q**), où **p** et **q** sont des états et **a** un symbole de l'alphabet B ; ces triplets sont les transitions de l'automate.

Les *automates déterministes* sont des automates dans lesquels pour tout état **p** de Q et pour tout symbole **a** de B, il existe au plus une transition partant de l'état **p** étiquetée par **a**. On demande en plus que I soit réduit à un unique élément, c'est-à-dire que l'automate ne possède qu'un seul état initial. On parle d'*automates acycliques* lorsqu'il n'existe pas de suite de transitions dans l'automate qui permette de passer deux fois par le même état.

[8] Un mot est reconnu par un automate si et seulement s'il est l'étiquette d'un chemin qui part de l'état initial et arrive à un état terminal. On lit le mot symbole après symbole, tout en se déplaçant dans l'automate en suivant les transitions étiquetées par le symbole courant. Si après avoir lu tout le mot on se trouve dans un état terminal de l'automate, le mot est reconnu ; sinon le mot n'est pas reconnu (M. Silberztein, 1989).

[9] On définit le langage reconnu par un automate, noté L(A), comme l'ensemble des mots reconnus.

pats}

(ii) L'arbre lexicographique est minimisé[10] selon la méthode basée sur la propriété d'équivalence (*E*) d'états dans un automate. Deux états *p* et *q* sont équivalents ((*p*, *q*) ∈ *E*), si et seulement si leurs langages droits sont égaux, le langage droit d'un état étant l'ensemble de tous les suffixes reconnus à partir de cet état jusqu'à l'un des états finals. Par exemple, dans la figure 7 les ensembles d'états qui ont les mêmes langages droits sont {5, 9, 13}, {4, 8, 12}, et {3, 11}. Si tous les états équivalents sont fusionnés, nous obtenons l'automate minimal comme la figure 8.

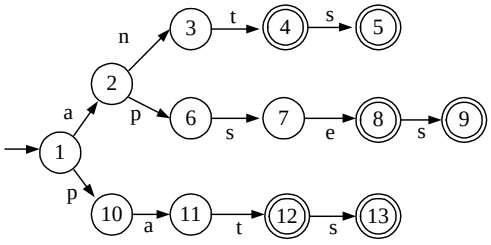


Figure 6. Un arbre lexicographique[11]

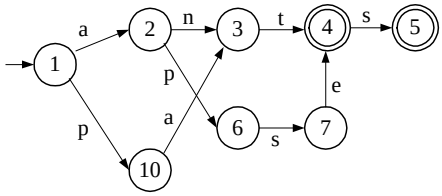


Figure 7. Un automate minimal[12]

L'utilisation d'un arbre lexicographique permet d'obtenir des vitesses intéressantes, puisque la vitesse est dépendante de la hauteur, et non plus directement du nombre d'entrées. Mais leur mise en œuvre nécessite un espace important. Ce point pose un problème pour la compression des formes fléchies pour les langues agglutinantes. Nous y reviendrons plus loin (pp. 156-157).

Dans un DELAF compressé sous INTEX, aucun symbole de sortie n'est attaché aux transitions. La production de l'étiquette grammaticale ne se fait qu'après avoir atteint l'état final. La minimisation est effectuée par l'algorithme ci-dessus. La seule différence est que les états terminaux sont distingués par les différentes productions qui leur sont attribuées. L'état final est associé au numéro qui représente l'indexe de l'information lexicale correspondante dans un tableau. Pour réduire le nombre des informations lexicales, les lemmes sont remplacés par les commandes utilisés lors du processus de la flexion dans INTEX. Si un lemme est égal à son entrée, il est effacé. Voici par exemple trois entrées du DELAF (M. Silberztein 1996, p. 74) :

---

[10] L'ensemble des automates reconnaissant un langage donné admet un plus petit élément par le nombre d'états ; cet automate le plus petit automate est appelé *automate minimal*, qui est unique. Des algorithmes de minimisation transforment un automate donné en l'automate minimal qui reconnaît le même langage.

[11] A. Chrobot 2001, p. 38.

[12] A. Chrobot 2001, p. 38.

aider,aider.V:W
aidions,aider.V:I1p:S1p
volions,voler.V:I1p:S1p

Elles sont représentées comme suit :

aider,.V:W
aidions,4er.V:I1p:S1p
volions,4er.V:I1p:S1p

Par conséquent, les deux lignes suivantes sont stockées dans le fichier d'informations :

.V:W
4er.V:I1p:S1p

Le transducteur dans INTEX est donc représenté par deux fichiers : un fichier « .bin » qui contient les transitions et un fichier « .inf » qui contient les informations lexicales que l'on trouve dans le dictionnaire. Extrayons un dictionnaire français représenté par un transducteur (M. Silberztein, 2000, p.162) :

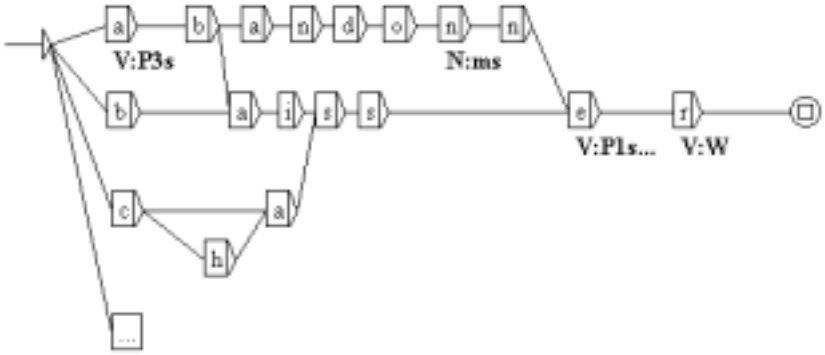


Figure 8. Extrait d'un dictionnaire représenté par un transducteur

Tous les nœuds qui contiennent une production sont traités comme des nœuds terminaux. Ce transducteur reconnaît la forme *a*, le nom *abandon*, les verbes à l'infinitif *abandonner*, *abaisser*, *casser* et *chasser*, les formes conjuguées *abandonne*, *abaisse*, *casse* et *chasse* appartenant au dictionnaire textuel d'origine. Ce transducteur est déterministe minimale, parce que les préfixes et les suffixes de toutes les formes y sont mis en facteur. Plus le dictionnaire contient des entrées avec les préfixes et les suffixes

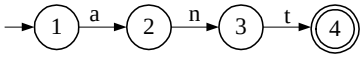
communs, plus le taux de compression d'un dictionnaire textuel vers un automate est élevé. Le temps d'accès est linéaire en fonction de la longueur du mot recherché.

Or, comme nous l'avons remarqué ci-dessus, le processus de compression en deux phases dans INTEX exige un espace mémoire suffisant au moins pour contenir l'arbre lexicographique qui résulte de la première phase (i) (p. 152). Ceci pose un problème de performance pour de gros dictionnaires, et surtout pour des langues à flexion très riche, car l'arbre lexicographique peut s'avérer plus grand que la taille de mémoire disponible. Pour améliorer ce problème, des méthodes plus efficaces sont élaborées : la méthode de D.Revuz (1991) et celle de J.Daciuk et al. (2000).

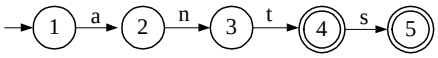
D.Revuz (1991) a proposé un algorithme qui permet, à partir d'un dictionnaire textuel d'obtenir un automate *pseudo-minimal* de taille très réduite par rapport à un arbre lexicographique correspondant (p. 152). Cet algorithme procède à une première minimisation, en comprimant certains suffixes communs à plusieurs mots. Les mots sont ajoutés progressivement dans l'ordre lexicographique inverse : la comparaison de lettre se fait de droite à gauche. Pour chaque nouveau mot, il faut trouver le plus long suffixe commun de ce mot avec tous les mots qui sont déjà dans l'automate. Ce suffixe commun est factorisé. L'insertion du préfixe du nouveau mot, qui reste après l'isolation du suffixe commun, se fait de gauche à droite en utilisant les états déjà existants pour un seul prédécesseur ou en les dédoublant pour plus d'un prédécesseur (D. Revuz 1991, pp. 42-45). Après avoir obtenu l'automate pseudo-minimal, la phase de minimisation a lieu : pour tous les éléments de la partition par hauteur croissante, trier les états par leurs transitions et s'ils sont équivalents, les fusionner. La minimisation est basée sur la propriété d'équivalence de langages droits (cf. voir la phase (ii), p. 153). La complexité de l'algorithme est linéaire en fonction de taille du lexique pour la première phase, et linéaire en fonction de taille du nombre d'états dans l'automate pseudo-minimal pour la deuxième phase.

J. Daciuk et al. (2000) a présenté une méthode plus efficace. Cette méthode ne contient qu'une seule phase. Les mots sont rajoutés dans l'ordre lexicographique de telle façon qu'après chaque nouvelle insertion l'automate obtenu reste minimal. Pour un mot à insérer, il faut d'abord trouver le plus long préfixe commun avec les mots qui sont déjà dans l'automate. A partir du dernier état de ce préfixe, il faut parcourir le chemin du mot précédent (de droite à gauche), et vérifier si certains de ses états ne peuvent pas être remplacés par des états équivalents (du point de vu de leurs langages droits) qui existent déjà dans l'automate. Finalement, le suffixe du mot courant est ajouté à l'automate, et la procédure recommence pour un mot suivant. Reprenons un exemple de construction incrémentale d'un automate minimal pour le même ensemble de mots que ceux de la

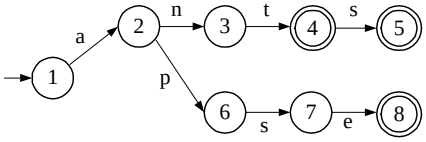
figure 6 (A. Chrobot, 2001, p. 39-40) :



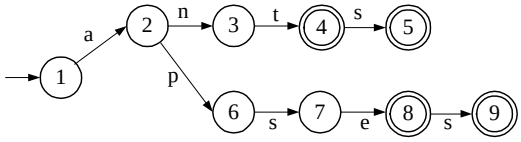
(i) création de l'automate pour *ant*



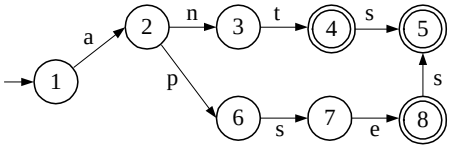
(ii) insertion de *ants*



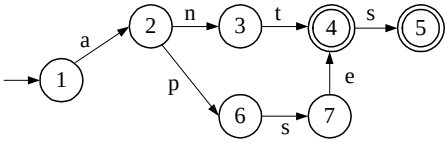
(iii) insertion de *apse*



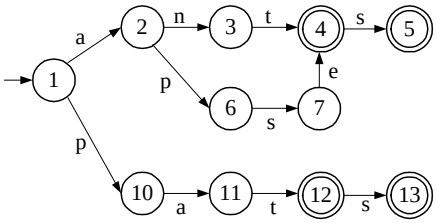
(iv) insertion de *apses*



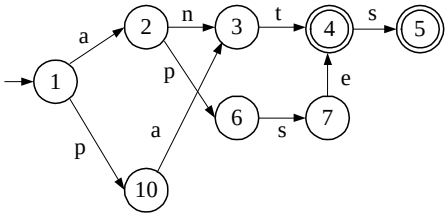
(v) remplacement de l'état 9 par 5



(vi) remplacement de l'état 8 par 4



(vii) insertion de *pat* et *pats*



(viii) remplacement de l'état 13,12, et 11 (dans cet ordre) par 5,4, et 3

Figure 9. Construction incrémentale d'un automate minimal

Maintenant, revenons au problème de la compression des formes fléchies des séquences nominales figées du coréen avec INTEX. L'algorithme d'INTEX pour la compression en deux étapes (pp. 152-153) n'est pas convenable pour celle du DELAF du coréen à la taille trop importante en raison du problème du mémoire, car la fabrication de l'automate déterministe acyclique à partir d'une liste textuelle demande une quantité de mémoire suffisante pour le contenir. Selon notre expérience, alors qu'INTEX peut engendrer les formes fléchies textuelles à quelques Go, il ne peut pas compresser un dictionnaire des formes fléchies textuelles d'une taille supérieure environ 200 Mo en une seule fois. En fait, puisque les algorithmes d'INTEX ne sont pas conçus au départ pour des langues agglutinantes comme le coréen, il existe des défauts de la procédure lors du traitement du coréen avec INTEX. Mais nous les avons utilisés, parce

que notre objectif était seulement de tester les données et les méthodes linguistiques que nous avons élaborées dans notre thèse. En utilisant INTEX pour le coréen, nous sommes allée jusqu'à la limite de la flexibilité permise par INTEX en attendant la réalisation d'une version coréenne d'INTEX. En prenant la méthode de Daciuk et al. (2000), on pourrait améliorer le problème de mémoire lors de la compression des très gros dictionnaires. Cependant, il faut noter que toutes les méthodes que nous avons envisagées partent d'une liste de formes fléchies. Pour la compression du coréen, mieux vaudrait probablement construire le dictionnaire de formes fléchies comprimé directement à partir du dictionnaire de formes canoniques, sans passer par la liste de formes fléchies qui est trop grosse.

Puisqu'INTEX ne peut pas compresser un dictionnaire des formes fléchies textuelles d'une taille supérieure environ 200 Mo en une seule fois, nos dictionnaires DELAF (5.3 Go) et DELACF (6.5 Go) ont une taille trop importante pour être compressés en une seule fois. Nous avons donc décidé de subdiviser, avant génération, le fichier des séquences nominales figées en plusieurs fichiers : 31 fichiers de NFF sans espace et 38 fichiers de NFF pour les noms figés avec espace. Chaque fichier d'environ 170 Mo est réduit à 4.3 Mo (fichier « .bin ») et à 12 Mo (fichier « .inf ») par la compression. Par conséquent, nous avons mis le dictionnaire comprimé des formes fléchies des NFF sans espace dans le dossier DELAF et celui avec au moins un espace dans le dossier DELACF[13] :

- 31 fichiers x 4.3 Mo de « .bin » = 133 Mo et
  31 fichiers x 12 Mo de « .inf » = 372 Mo dans le dossier DELAF
- 38 fichiers x 4.3 Mo de « .bin » = 163 Mo
  38 fichiers x 12 Mo de « .inf » = 456 Mo dans le dossier DELACF

Nous récapitulons notre méthode de construction des dictionnaires des formes fléchies des NFF avec INTEX :

[13] Plus exactement,
31 fichiers x 4.45 Mo de « .bin » ≅ 138 Mo et
31 fichiers x 11.97 Mo de « .inf » ≅ 371 Mo dans le dossier DELAF
38 fichiers x 4.13 Mo de « .bin » ≅ 157 Mo
38 fichiers x 12.1 Mo de « .inf » ≅ 460 Mo dans le dossier DELACF

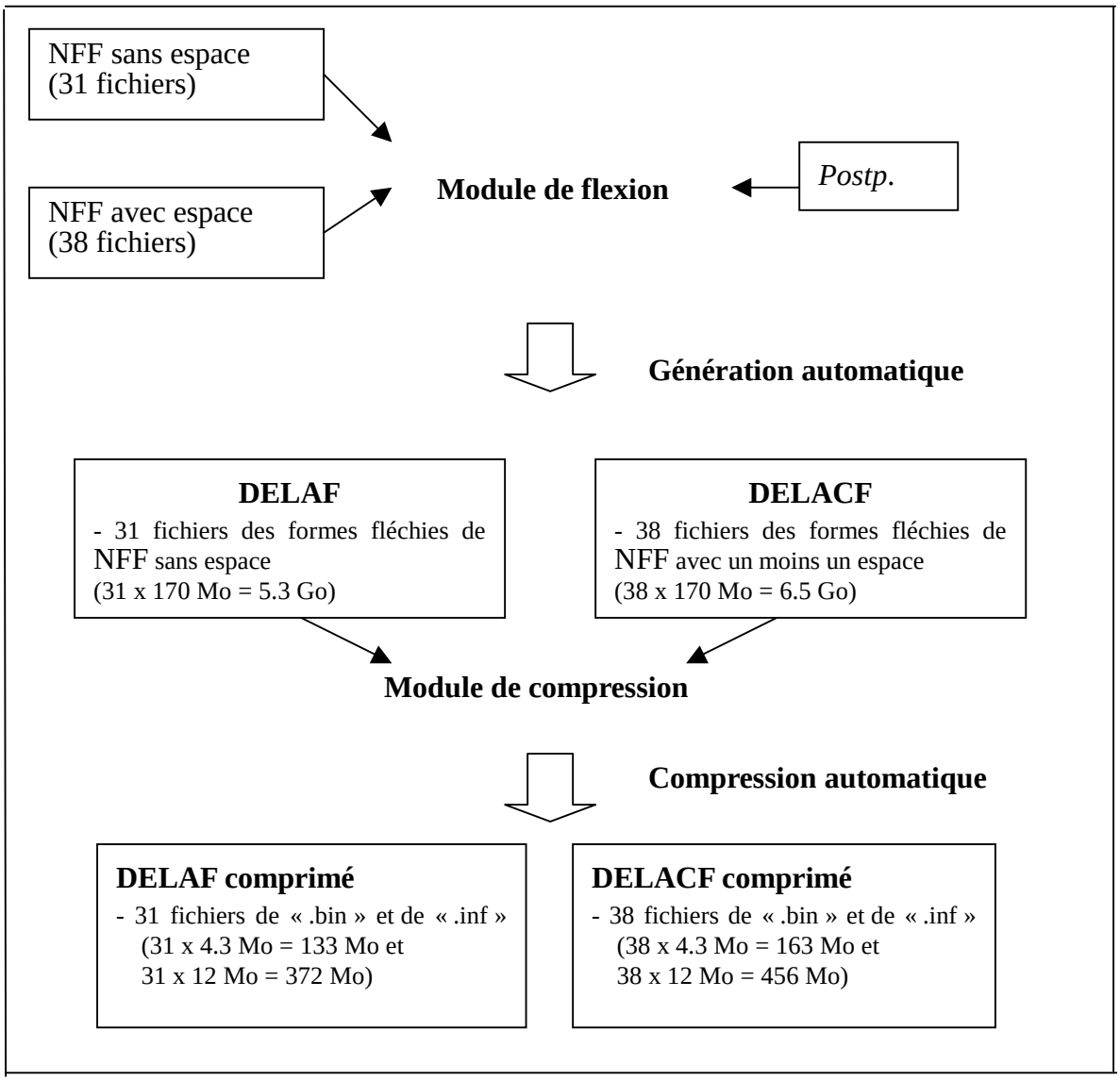


Figure 10. Construction automatique du dictionnaire des formes fléchies de NFF à l'aide d'INTEX

2. Reconnaissance automatique des NFF par dictionnaire électronique

2-1. Reconnaissance des NFF avec INTEX

Pour la reconnaissance des NFF dans les textes coréens par la consultation de notre dictionnaire, nous utilisons le système INTEX et le CD-ROM de corpus écrit du Projet Sejong 21 (http://www.sejong.or.kr).

Or, lors de la consultation du dictionnaire avec INTEX, nous ne pouvons pas

lancer les 69 fichiers « .bin » en même temps ni par l'interface graphique ni par la commande en ligne parce que le nombre de fichiers « .bin » dans le dossier DELACF est limité à 10 fichiers par consultation. Pour le moment, lorsque nous utilisons notre dictionnaire avec INTEX, nous appliquons les quatre parties de DELAF et de DELACF avec le même texte :

1ière partie : « 1sflx.bin ~ 10sflx.bin » et « 1cflx.bin ~ 10cflx.bin »
2ième partie : « 11sflx.bin ~ 20sflx.bin » et « 11cflx.bin ~ 20cflx.bin »
3ième partie : « 21sflx.bin ~ 30sflx.bin » et « 21cflx.bin ~ 30cflx.bin »
4ième partie : « 31sflx.bin » et « 31cflx.bin ~ 38cflx.bin »

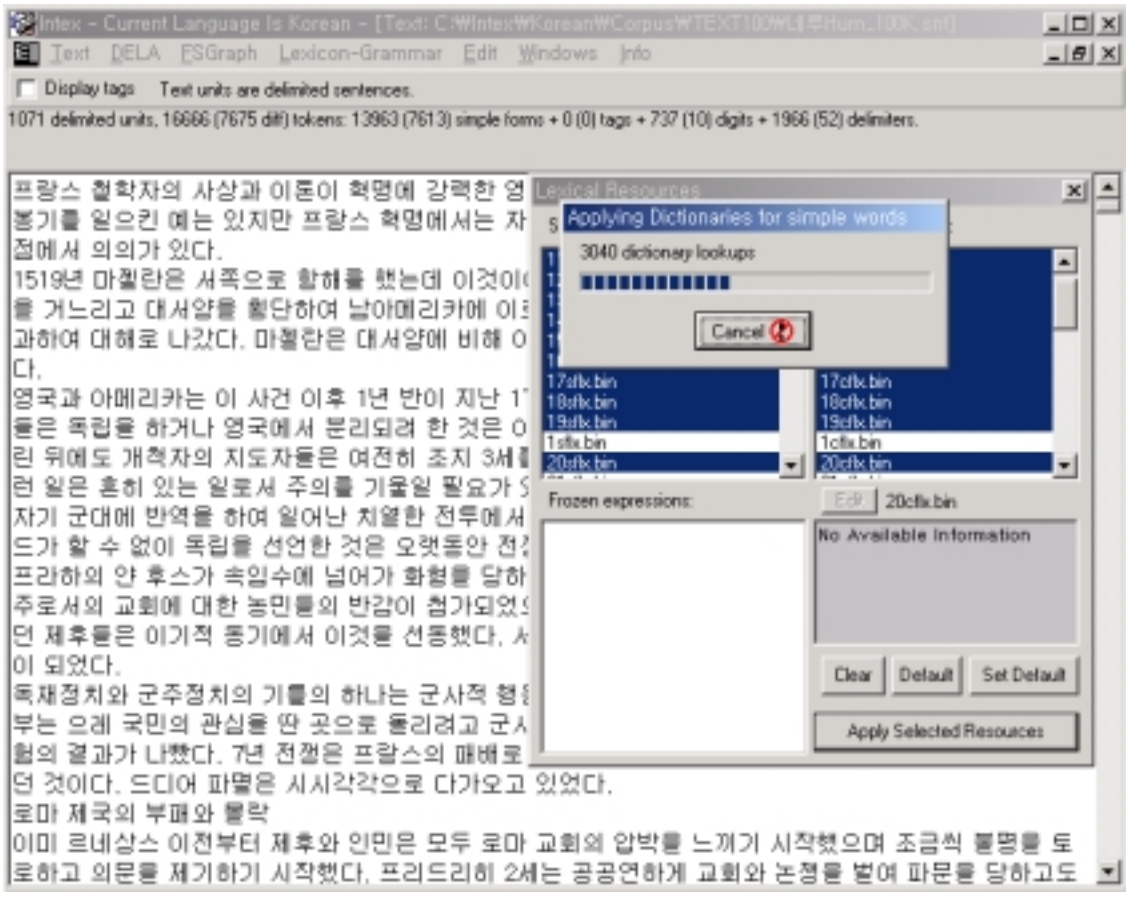


Figure 11. Application de la 2ième partie de DELAF et de DELACF

Nous présentons un échantillon réduit des listes reconnues par la consultation de la 2ième partie du DELAF et du DELACF. Le texte utilisé a une taille de 100 Ko et il s'agit d'un livre sur l'histoire du monde.

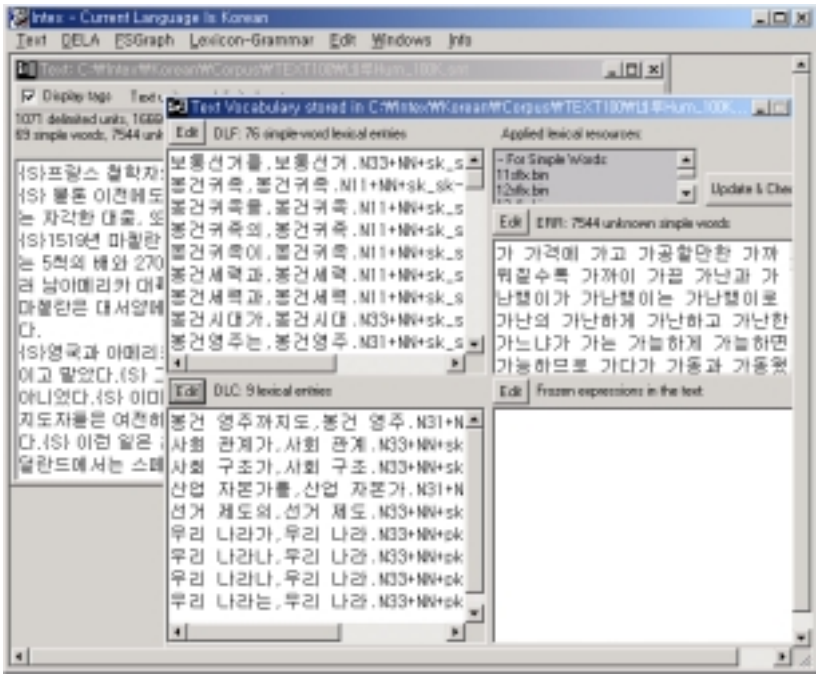


Figure 12. Résultat de consultation de la 2ième partie de DELAF et de DELACF

INTEX peut construire concordance des NFF reconnus par la consultation du dictionnaire à l'aide du module « Locate Pattern ». Lors de cette opération, nous utilisons la variable <N> qui fait référence aux noms décrits dans notre dictionnaire.

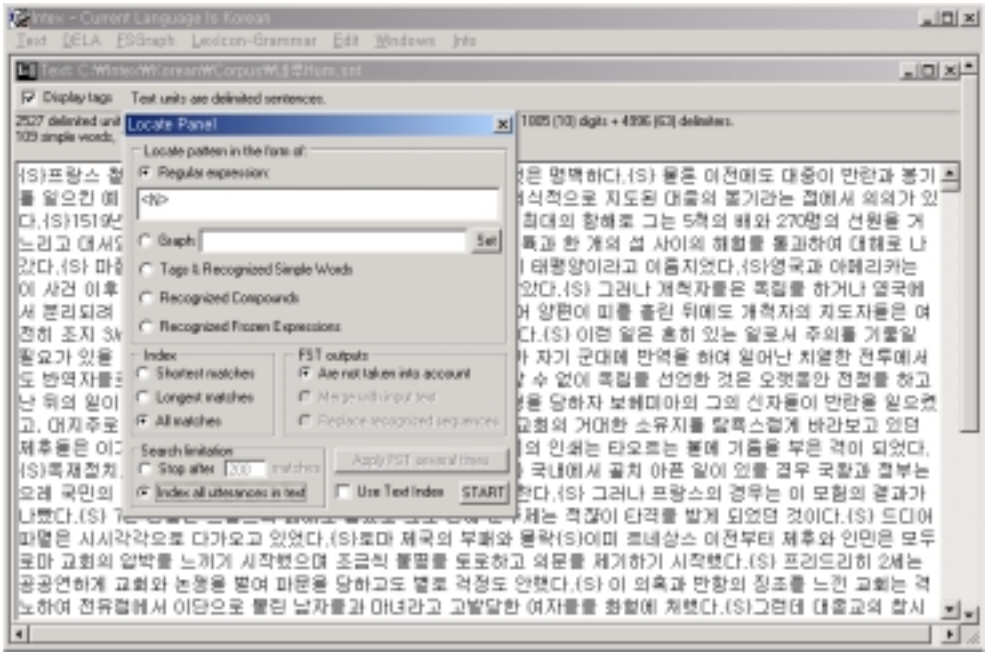


Figure 13. Recherche de NFF dans le texte à l'aide de la variable <N>

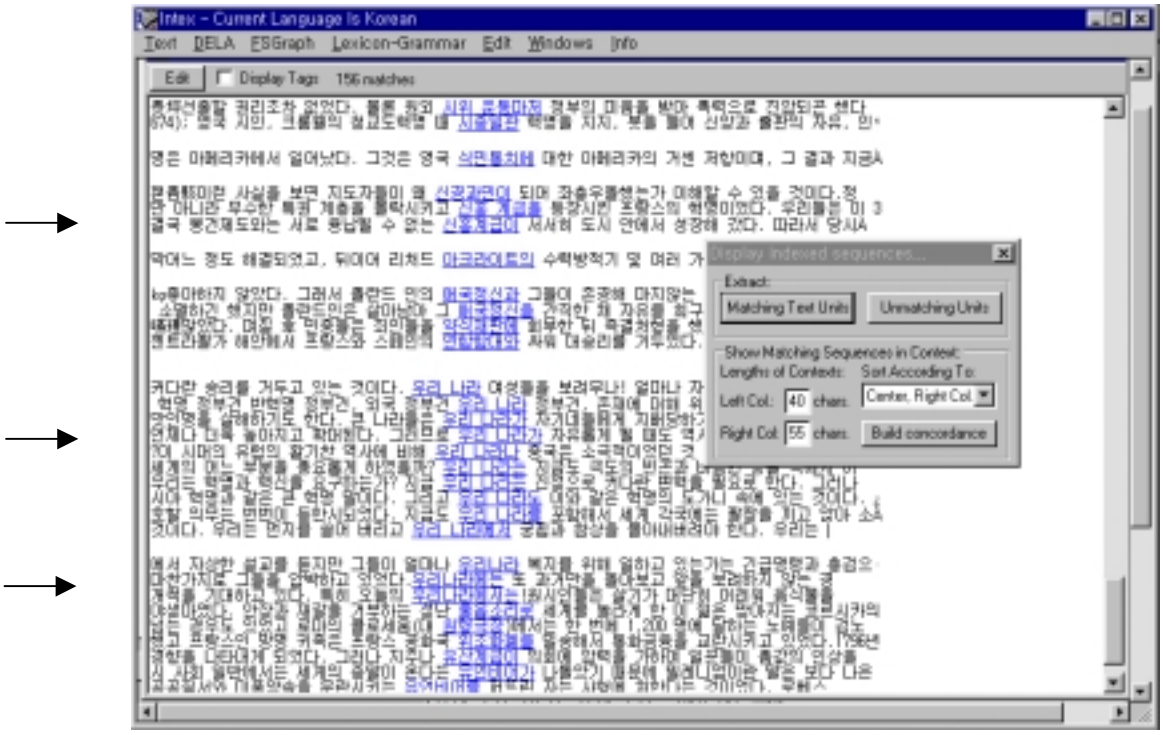


Figure 14. Concordance des noms figés du texte trouvés dans le dictionnaire[14]

Dans la figure 14, nous vérifions que l'auteur alterne fréquemment l'utilisation de l'espace pour un même NFF dans son texte. Par exemple, les noms figés *sinheung jyegeup* « ascension - classe = classe ascendante » ou *uli nala* « nous - pays = la Corée » se réalisent différemment dans le même texte (voir ⟶).

Notre expérience (cf. figure 15) de reconnaissance automatique des NFF avec DELAF et DELACF dans plusieurs types de textes montre bien l'importance des variations liées à l'espacement en coréen. Par exemple, les auteurs des textes 4, 5, 6 et 9 ont tendance à insérer l'espace dans les séquences nominales figées, tandis que les auteurs des autres textes privilégient la soudure. Pour traiter les textes coréens, il est donc indispensable de prendre en considération les variantes liées à l'espacement et de préciser les espacements possibles dans le dictionnaire électronique des séquences nominales en coréen.

[14] Pour le moment, INTEX sous Windows coréen peut afficher le coréen pour plusieurs fonctionnalités, mais ne peut pas afficher directement la concordance. Cependant, nous pouvons afficher indirectement le coréen dans INTEX grâce aux fonctions de « copier » et « coller » du logiciel MicroSoft Word.

Type du texte[15]	Taille du texte	Nombre de NFF reconnus à la main	Nombre de NFF reconnus par DELAF (A)	Nombre de NFF reconnus par DELACF (B)	Nombre de NFF reconnus (A + B)	Portion reconnue par DELAF et DELACF
1 : Essai C	100 Ko	624	172	90	262	42 %
2 : Histoire N	100 Ko	270	154	15	169	63 %
3 : Société H	100 Ko	842	267	7	274	33 %
4 : Journal D1	100 Ko	462	7	270	277	60 %
5 : Journal D2	100 Ko	535	8	265	273	51 %
6 : Journal CH	100 Ko	414	16	192	208	50 %
7 : magazine K	100 Ko	736	169	9	178	24 %
8 : Journal HK	100 Ko	881	199	53	252	29 %
9 : Journal HKL1	100 Ko	515	3	386	389	76 %
10 : Journal HKL2	100 Ko	696	130	66	196	28 %
Total	1 Mo	5975	1125	1353	2478	41 %

Figure 15. Portion reconnue des NFF par DELAF et DELACF[16]

Dans notre expérience, nous avons trouvé 5975 NFF à la main dans les 10 textes (soit un total de 1 Mo). Dans les 10 textes, 2478 NFF sur 5975 NFF sont reconnus par la consultation de nos dictionnaires (soit 41% de NFF). Par conséquent, nous pouvons estimer en moyenne que 6 NFF apparaissent par Ko de texte et nos dictionnaires DELAF et DELACF reconnaissent 2,5 NFF par Ko de texte. Nous pouvons augmenter ce nombre par ajout des noms figés non reconnus dans notre dictionnaire électronique.

2-2. Perspectives pour le traitement des séquences nominales non reconnues

Il existe trois raisons pour expliquer la non-reconnaissance de certaines séquences nominales : la première raison vient de l'insuffisance du dictionnaire, la deuxième des séquences nominales libres soudées et la dernière des noms prédicatifs combinés avec les suffixes verbaux comme *-hada* « faire » ou des noms combinés avec le copule *-ida* « être ». Nous discutons de ces trois problèmes dans le traitement des

[15] cf. Pour la source, voir la Bibliographie.
[16] Nous avons choisi 10 textes au hasard dans le corpus du *Projet Sejong 21*.

séquences nominales.

Par notre expérience, nous constatons que notre dictionnaire de 45000 NFF n'est pas suffisant pour reconnaître tous les NFF dans les textes coréens. Pour compléter le dictionnaire des séquences nominales figées, il est nécessaire d'étendre les dictionnaires des NFF à l'aide de corpus et il sera indispensable de construire les dictionnaires de NC et de NFO après des études systématiques sur les autres types de noms (noms compacts dérivés, noms figés à espacement obligatoire). En outre, il est nécessaire au moins de construire les dictionnaires des noms propres figés (par exemple, les toponymes, les noms de pays, etc.) et les noms techniques figés en considérant leur espacement dans l'usage, bien que nous ne puissions pas mettre tous les termes néologiques dans un dictionnaire électronique. Pour le traitement automatique des séquences nominales, nous avons besoin de 7 types de dictionnaires électroniques des séquences nominales :

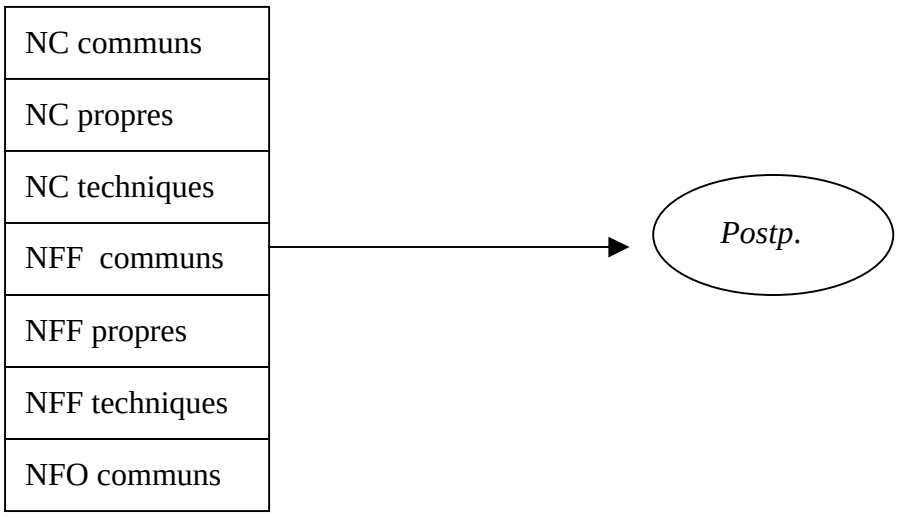


Figure 16. Dictionnaires des séquences nominales[17]

Or, bien que nos dictionnaires soient assez importants en nombre d'entrées, nous sommes confrontés à un autre type de problème lors du traitement automatique des séquences nominales, à savoir la reconnaissance des séquences nominales libres soudées. Ces deux problèmes, la reconnaissance des séquences nominales figées et la reconnaissance des séquences nominales libres soudées, sont liés aux conventions typographiques. Comme nous l'avons vu dans notre expérience (cf. figure 15), les

[17] Il nous semble qu'il existe peu de NFO propres et de NFO techniques.

variantes d'espacement des séquences nominales figées peuvent être identifiées par la construction des dictionnaires des séquences nominales figées qui précisent les variations typographiques. Cependant, le problème des séquences libres soudées constituées de plusieurs noms n'est pas résolu par la construction d'un dictionnaire électronique le plus exhaustif possible. Ce problème est capital dans la mesure où les applications pratiques comme la traduction automatique ou l'extraction de mots-clés demandent une analyse précise des séquences nominales. Pour limiter le nombre des mots non reconnus, il faut segmenter les séquences nominales libres soudées en appliquant les dictionnaires des noms et des séquences nominales figées. Pour obtenir une segmentation précise des séquences nominales libres, a priori, il faut construire un dictionnaire électronique suffisamment exhaustif des séquences nominales figées, car une séquence nominale libre peut contenir une séquence nominale figée comme nous l'avons vu dans le chapitre 2.

Même si l'on segmente les séquences nominales libres soudées en appliquant des dictionnaires de séquences nominales figées suffisamment exhaustifs, on peut trouver des cas de segmentations ambiguës[18]. Nous ne pouvons pas entrer dans les détails, mais nous remarquons le problème des segmentations ambiguës des séquences nominales libres :

(3) a. *dae-hak-saeng-seon-gyo-hoe* (étudiant - réunion de la mission = réunion de la mission des étudiants)

b. *daehaksaeng* / *seongyohoe*
(étudiant / réunion de la mission = réunion de la mission des étudiants)

c. *daehak* / *saengseon* / *gyohoe* (université / poisson / église = ?)

(Yoon B. H et al., 1995)

Par exemple, la séquence libre (3a) peut être segmentée de plusieurs façons sans dictionnaire exhaustif. Avec un dictionnaire suffisamment complet, il existe encore deux possibilités : (3b) et (3c). Le découpage correct est (3b), car (3c) n'a aucun sens en coréen. Pour obtenir la segmentation exacte, il serait indispensable d'en savoir plus sur la grammaire des séquences nominales libres à espacement facultatif.

La troisième raison de l'existence de noms figés inconnus vient du cas où un nom prédicatif est combiné avec un suffixe verbal comme *-hada* « faire » ou du cas où

[18] Pour les solutions statistiques, voir la section 2-2 dans le chapitre 1.

un nom est combiné avec la copule *-ida* « être »[19]. Il s'agit de noms prédicatifs qui entrent dans la structure N_0 *W Npred-hada* et de noms qui entrent dans la structure N_0 *W N-ida*. Si le but est d'extraire tous les noms figés, il faudrait ajouter les formes fléchies de *Npred-hada* et de *N-ida* dans notre dictionnaire des formes fléchies des NFF après des études systématiques sur les suffixes flexionnels des verbes.

3. Avantages de notre dictionnaire électronique pour plusieurs applications du traitement automatique

Nous explorons les avantages de notre dictionnaire électronique pour plusieurs applications du traitement automatique comme la recherche d'informations, l'extraction de mots-clés, la traduction automatique, etc.

3-1. Recherche d'informations et extraction de mots-clés

Les systèmes coréens les plus représentatifs comme l'analyseur morphologique de Kang S.S (1993 et 1998), celui de *KAIST*, celui du *Projet Sejong 21*, etc. ne considèrent pas les variantes liées à l'espacement dans le dictionnaire, et ils segmentent les formes fléchies des séquences nominales présentes dans les textes coréens en une séquence nominale et une postposition nominale avec un dictionnaire de noms et un autre de postpositions nominales sans fusionner ces deux ensembles de données. En cas de segmentation ambiguë, chaque système utilise une technique spécifique : ces cas exceptionnels sont répertoriés à part dans un dictionnaire de cas exceptionnels, ils sont résolus par apprentissage automatique à partir de gros corpus, etc.

Lorsque nous utilisons notre dictionnaire des formes fléchies des NFF dans le domaine de la recherche d'information, le résultat de la recherche est beaucoup plus précis que celui de systèmes préexistants grâce à notre méthodologie de construction du dictionnaire. En vue de la reconnaissance des séquences nominales figées par dictionnaire, nous avons classé celles-ci en trois catégories selon les conventions typographiques : noms compacts, noms figés à espacement facultatif et noms figés à espacement obligatoire, puis nous avons séparément construit le dictionnaire électronique de 45000 NFF et celui des séquences de postpositions nominales et enfin nous fusionnons ces deux ensembles de données. De plus, nous avons décrit des informations linguistiques utiles dans le dictionnaire des NFF.

[19] En coréen, tous les noms peuvent se combiner avec la copule *-ida* « être ». C'est pourquoi selon certaines linguistes, la forme *-ida* est considérée comme une postposition nominale, dite prédicative.

Nous pouvons rechercher toutes les variantes des NFF sans erreur (cf. les figures 8-9). Notre dictionnaire permet de segmenter précisément les formes fléchies des NFF en un NFF et une séquence de postpositions nominales. Par exemple, la forme *ga* peut être le suffixe de « personne » ou la postposition du nominatif.

yeolu hwaga, *yeolu hwaga*.N31+Hum
*yeolu hwaga**ga***, *yeolu hwaga*.N31+Hum:*_ga***Nmtf**

Si nous utilisons notre dictionnaire, nous n'avons pas besoin de lever l'ambiguïté entre les deux homonymes de la même forme *ga*.

Nous pouvons rechercher n'importe quelle postposition nominale ou une séquence de postpositions nominales donnée. Par exemple, nous pouvons trouver les séquences nominales qui contiennent les variantes *-ga*/*-i* de la postposition nominale du nominatif par l'expression régulière <:A>[20]. Remarquons que nous avons codé *A* la postposition du nominatif <*ga*> dans le chapitre 4 :

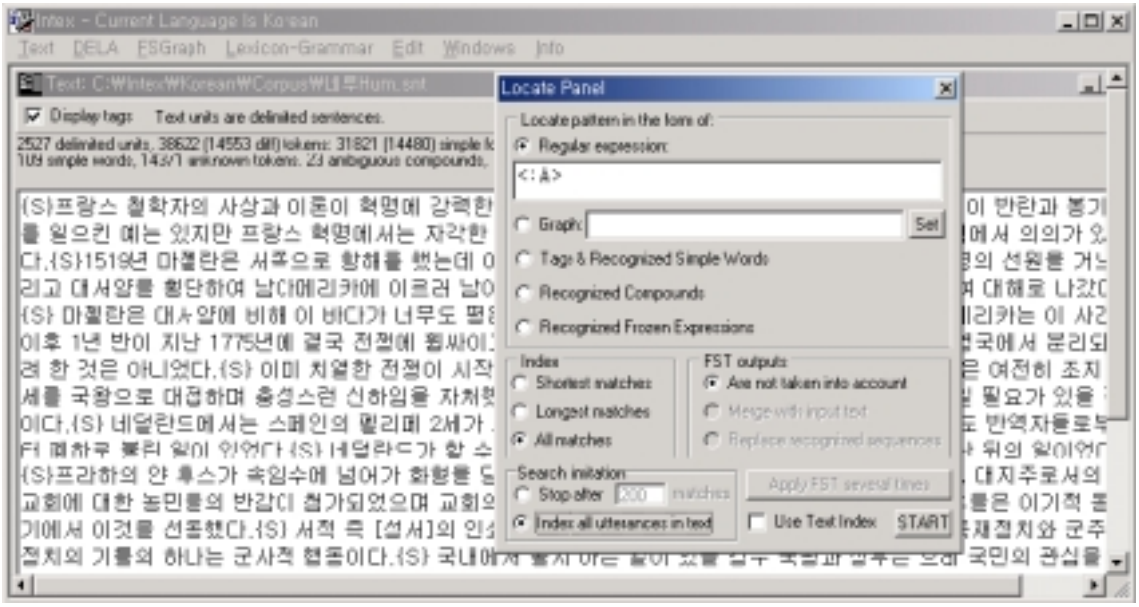


Figure 17. Recherche d'une postposition du nominatif <*ga*>

Nous constatons que notre dictionnaire reconnaît bien les séquences nominales figées qui contiennent les variantes du nominatif <*ga*>.

[20] Les deux points introduisent des informations sur une postposition nominale. *A* est le code de la forme canonique et de la classe linguistique de la postposition nominale *-ga/-i* (<*ga*, **Nmtf**>).

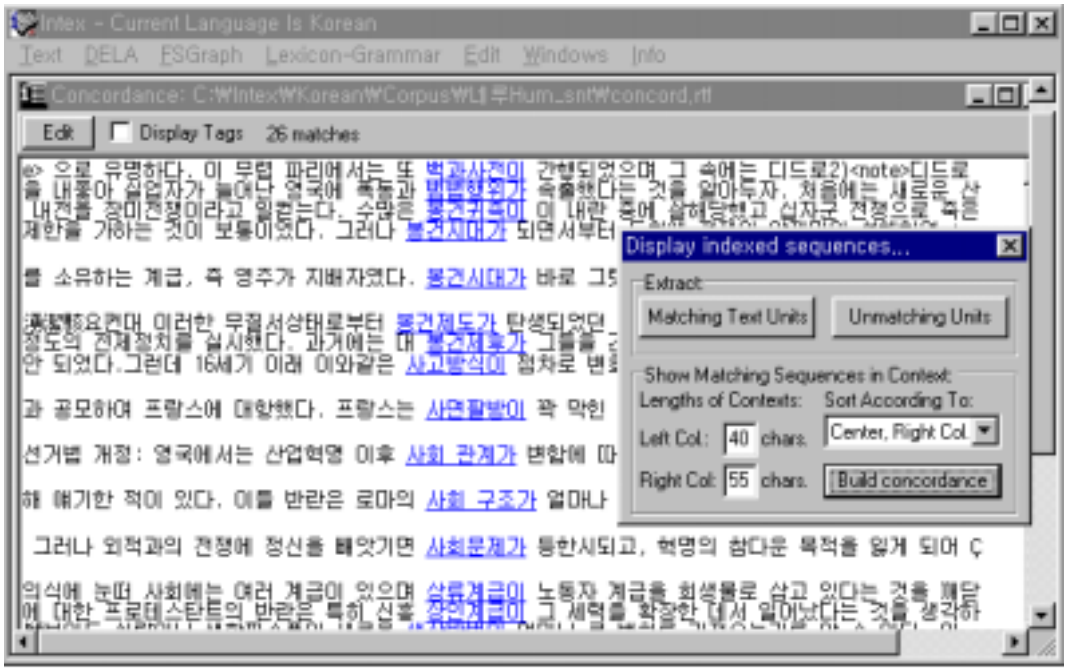


Figure 18. Concordance de la postposition <*ga*> dans un texte

Les autres systèmes ne permettent pas la recherche des séquences qui contiennent une certaine postposition en n'importe quelle position dans les séquences parce que le dictionnaire de postpositions nominales n'est qu'une liste de postpositions nominales (ou de séquences de postpositions nominales) et les séquences de postpositions nominales n'y sont pas segmentées en postpositions nominales. Mais notre dictionnaire le permet : si nous voulons rechercher toutes les séquences (*N-**jocha***, *N-eseo-**jocha***, *N-eseo-**jocha**-do*, etc.) qui contiennent la postposition *-jocha* en n'importe quelle position dans les séquences, nous pouvons les rechercher par l'expression régulière < :3> où 3 est le code de la forme canonique et la classe linguistique de la postposition nominale *-jocha* (<*jocha*, **Aux**>).

Les informations sur le nom tête de chaque séquence sont évidemment utiles pour l'indexation en vue de la recherche d'informations ou l'extraction de mots-clés. La fonction de sous-classifieur de N_1 dans N_1N_2 ou de X dans XN est observée dans le chapitre 2 : les critères *N_1N_2 est un (N_2 + type de N_2)* et *XN est un (N + type de N)* sont appliqués pour l'identification des noms figés. Nous avons codé cette information (cf. chapitre 3) et rappelons ci-dessous un exemple pour chaque type (a) - (h) :

(4) a. *daehap jogae*, ***N+NN+12*** « palourde » « coquillage » = « palourde »

b. *iut sachon*, ***N+NN+1*** « voisin » « cousin » = « voisin plus familier que des

cousins »

c. *chuli soseol*, ***N+NN+2*** « déduction » « roman » = « roman policier »

d. *jigak daejang*, ***N+NN+0*** « retard » « chef » = « personne qui est toujours en retard »

e. *minju jeongchi*, ***N+XN+2*** « démocratique » « politique » = « politique démocratique »

f. *gwagyeok bunja*, ***N+XN+0*** « radical » « élément » = « extrémiste ou radical »

g. *beseuteu seleo*, ***N+XX+0*** « best » « seller » = « succès de librairie »

h. *bitamin di*, ***N+NX+ 1*** « vitamine » « D » = « vitamine D »

Par exemple, lors de l'indexation ou de l'extraction de mots-clés, le code +12 permet d'indexer la séquence figée entière ainsi que chaque nom N_1 et N_2. Le code +1 (ou +2) permet d'indexer la séquence figée entière et le nom N_1 (ou N_2). Le code +0 empêche l'indexation des 2 éléments et autorise seulement indexation de la séquence figée entière. Par exemple :

(5) a. ***daehap jogae***, ***N+NN+12*** : *daehap jogae, daehap, jogae*

b. ***iut*** *sachon*, ***N+NN+1*** : *iut sachon, iut* (**sachon*)

c. *chuli* ***soseol***, ***N+NN+2*** : *chuli soseol, soseol* (**chuli*)

d. ***jigak daejang***, ***N+NN+0*** **:** *jigak daejang* (**jigak, * daejang*)

e. *minju* ***jeongchi***, ***N+XN+1*** : *minju jeongchi, jeongchi* (**minju*)

f. ***gwagyeok bunja***, ***N+XN+0*** : *gwagyeok bunja* (**gwagyeok, *bunja*)

g. ***beseuteu seleo***, ***N+XX+0*** : *beseuteu sele* (**beseuteu,* seleo*)

h. ***bitamin*** *di*, ***N+NX+ 1*** : *bitamin di, bitamin* (**di*)

3-2. Analyse syntaxique

Les codes indiquant la catégorie grammaticale, les informations sémantiques, les informations morpho-syntaxiques et syntaxiques sont indispensables non seulement pour l'analyse morphologique mais aussi pour l'analyse syntaxique.

Nous avons codé les noms humains *Hum*, les noms d'animaux *Anm*, les noms de plantes *Plt*, les noms institutionnels humains *InstHum* et les noms humains collectifs *CollHum*. L'information sémantique joue un rôle très important pour la description syntaxique du lexique-grammaire (cf. figure 19).

Les noms prédicatifs (soit compacts soit figés) qui fonctionnent comme prédicats sémantiques dans les phrases, requièrent une description séparée de leurs

comportements syntaxiques dans le lexique-grammaire. Nous avons codé les noms figés prédicatifs dont le nom tête est morpho-syntaxiquement associé à une construction verbale par la transformation « PREDHN=V-n ». Ce code permet de réutiliser les tables syntaxiques du lexique-grammaire des noms compacts prédicatifs[21]. Nous avons tiré un exemple de la description des noms prédicatifs à verbe support *hada*. Par exemple, *gyeolhon* « mariage » dans la table WNH2SYM[22] est un nom compact prédicatif et *gukje gyeolhon* « international » « mariage » = « mariage avec un étranger » est un nom figé prédicatif dont le nom tête *gyeolhon* « mariage » est morpho-syntaxiquement associé à une construction verbale. *gukje gyeolhon* partage le comportement de *gyeolhon* et nous pouvons donc réutiliser cette table du lexique grammaire.

[21] Han, S.H (2000). ANNEXE.

[22] Han, S.H (2000). ANNEXE, p. 72.

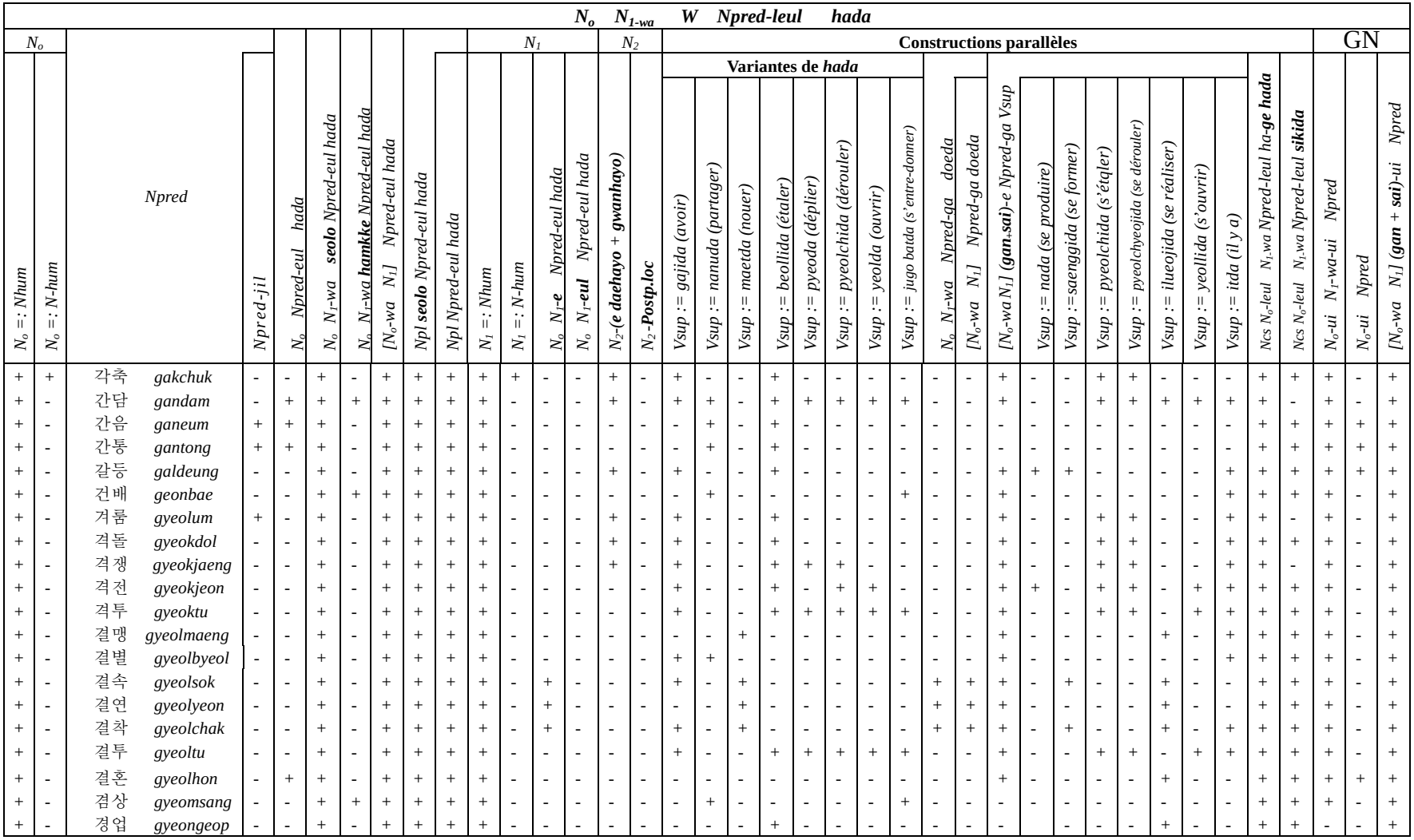


Figure 19. Extrait de Table WNH2SYM

3-3. Traduction automatique

Notre dictionnaire peut être étendu en un dictionnaire bilingue pour la traduction automatique des séquences nominales figées. Par exemple, la forme canonique associée à chaque entrée du dictionnaire de NFF (DELAC) peut être utilisée dans des applications de traduction.

(6) a. *gukchaek eunhaeng*, banque nationale. N12+NN+sk_sk-PRED+2+InstHum
b. *gukchaek eunhaeng*, government owned bank.
N12+NN+sk_sk-PRED+2+InstHum

(6a) est pour la traduction du coréen vers le français et (6b) pour celle du coréen vers l'anglais. Les transducteurs ne peuvent pas être directement appliqués en mode de remplacement pour les noms coréens, parce qu'en général les noms apparaissent sous une forme fléchie dans les textes. Il faut prendre en considération les postpositions nominales attachées aux noms. Pour la traduction automatique des noms coréens, tout d'abord, il faut appliquer le dictionnaire des formes fléchies des séquences nominales figées. Cette étape permet de segmenter les postpositions nominales. Ensuite, les transducteurs des séquences nominales figées doivent être appliqués en mode remplacement.

Quant à la traduction des postpositions nominales qui est plus complexe que celle des séquences nominales figées, elle est différente selon les classes de postpositions nominales. Par exemple, les postpositions auxiliaires qui ajoutent une nuance à la phrase comme *-do* « aussi », *-man* « seulement », etc. peuvent être décrites dans les cas simples dans les transducteurs de traduction, mais les postpositions nominales dites adverbiales comme *-e* « à », *-eseo* « de », etc. requièrent l'utilisation du lexique-grammaire des prédicats parce que leurs emplois sont liés au comportement syntaxique des prédicats. Les postpositions du nominatif ou de l'accusatif donnent une fonction syntaxique au substantif dans la phrase du coréen, et dans la langue cible la fonction du substantif dans la phrase n'est pas nécessairement la même. Il faut étudier chaque postposition nominale pour la traduction automatique.

On pourrait écrire des transducteurs de traduction pour les séquences nominales, mais il est difficile de les utiliser directement pour plusieurs raisons : les variantes liées à l'espacement, la segmentation entre les postpositions nominales et les séquences nominales, etc.

4. Conclusion

En vue de la reconnaissance automatique des NFF, nous avons séparément construit le dictionnaire électronique de 45000 NFF et celui des séquences de postpositions nominales, et puis nous avons fusionné ces deux ensembles de données. Nous récapitulons les avantages principaux de notre dictionnaire des formes fléchies des NFF (DELAF pour les NFF sans espace et DELACF pour NFF avec au moins un espace) :

(i) Il permet de rechercher toutes les variantes de NFF liées à l'espacement
(ii) Il permet de segmenter précisément les formes fléchies des NFF en un NFF et une séquence de postpositions nominales
(iii) Il permet de segmenter précisément les séquences de postpositions nominales

De plus, il comporte les codes utiles pour plusieurs applications du traitement automatique comme la recherche d'informations, l'extraction de mots-clés, la traduction automatique, etc. :

(i) Code indiquant un trait sémantique et le statut de nom prédicatif
(ii) Code indiquant le nom tête de chaque entrée
(iii) Code indiquant l'origine et la catégorie grammaticale