DESCRIPTION ET CONCEPTION D'UNE PLATE-FORME ROBUSTE COMBINANT DES ANALYSEURS D'ENONCE

Brunet-Manquat Francis,
Doctorant en Informatique
Francis.Brunet-Manquat@imag.fr , + 33 4 76 51 43 59

Adresse professionnelle
Laboratoire CLIPS-IMAG, équipe GETA ★ BP 53 ★ 38041 Grenoble Cedex 9

Résumé : L'objectif de cet article est de présenter nos travaux sur la combinaison d'analyseurs pour produire une analyse robuste de la langue. L'idée est de créer une plate-forme robuste d'analyse capable d'intégrer des analyseurs linguistiques existant (analyseurs syntaxiques ou/et de dépendances), et de fusionner leurs résultats dans le but d'obtenir une analyse de dépendances pour des énoncés quelconques.

Summary : The article's goal is to present our work about the combination of parsers to produce a robust parsing. The idea is to create a robust analysis tool capable of integrating existing linguistic parsers (syntactic parsers or dependency parsers), to merge their results and to calculate the best information to obtain a dependency parse of the input sentence.

Mots clés : Traitement automatique de la langue naturelle, analyseur de dépendances, analyseur syntaxique.

Description et conception d'une plate-forme robuste combinant des analyseurs d'énoncé

L'objectif de cet article est de présenter nos travaux sur la combinaison d'analyseurs pour produire une analyse robuste de la langue. L'idée est de créer une plate-forme robuste d'analyse capable d'intégrer des analyseurs linguistiques existant (analyseurs syntaxiques ou/et de dépendances), et de fusionner leurs résultats dans le but d'obtenir une analyse de dépendances pour des énoncés quelconques.

1 - INTRODUCTION

Notre laboratoire est impliqué dans deux projets internationaux importants : CSTAR et son projet européen associé NESPOLE ! (http://nespole.itc.it) pour la traduction simultanée de l'oral et Universal Networking Language, UNL (http://www.unl.ias.unu.edu), pour la traduction de l'écrit. Ces projets se caractérisent notamment par la présence d'une représentation pivot des énoncés et par le fait que l'énoncé à traduire est susceptible d'être *bruité*, c'est-à-dire pas forcément conforme à la grammaire académique de la langue française. Dans un système à pivot, l'énoncé d'une langue source donnée est *analysé* dans la représentation pivot avant d'être *généré* vers une ou plusieurs langues cibles. La nécessité de pouvoir traiter des entrées bruitées implique des outils robustes d'analyse de la langue, capable de fournir une analyse même partielle de la phrase.

L'idée est de concevoir une telle plate-forme en prenant le « meilleur » d'un ensemble d'analyseurs. Le principe étant de combiner différents résultats d'analyse pour une même phrase, puis de calculer les meilleures informations pour obtenir la ou les meilleures analyses possibles. Notre idée s'inspire du système ROVER, Recognizer Output Voting Error Reduction [Fis97], qui combine des résultats de systèmes de reconnaissance de parole pour obtenir la meilleure reconnaissance possible.

Notre approche se base principalement sur la méthode dite du « vote à la majorité », plus une information sera commune aux différents analyseurs, plus le poids de cette information sera fort, mais également sur un apprentissage permettant d'adapter les poids associés aux informations fournis en fonction de l'énoncé (par exemple, entrée bruitée ou non). À partir d'une telle approche, nous comptons construire un analyseur de dépendances robuste car regroupant un maximum d'informations issues d'une combinaison d'analyseurs.

Nous décidons de fonder notre plate-forme sur une représentation par dépendances, décrivant les relations entre mots (les relations peuvent être de tous types : syntaxique, logique ou sémantique). Fort des différents projets réalisés, ce type de représentation nous semble plus adapté pour une analyse robuste. Elle nous permet, par exemple, de représenter clairement et simplement l'analyse partielle d'une phrase mal formée.

2 – SPECIFICATIONS DE LA PLATE-FORME D'ANALYSE

La plate-forme d'analyse ne doit pas intégrer les analyseurs, mais elle doit être capable d'extraire les informations linguistiques des résultats qu'ils produisent, de les interpréter, de les fusionner et enfin de produire un arbre de dépendances (ou plusieurs) combinant le meilleur des informations extraites.

2.1 - Les différentes étapes du traitement

La plate-forme comporte deux étapes : la *normalisation* des résultats d'analyse et la *construction* des analyses de dépendances.

L'étape de normalisation se compose de deux phases :

- La phase d'*extraction* des informations linguistiques des résultats obtenus avec chaque analyseur linguistique ;
- La phase de *projection* qui traite les informations extraites pour obtenir un ensemble de structures de dépendances dite normalisée, conforme à notre norme de description linguistique (voir 3.2.3 – Vers une norme de description linguistique). Chaque information normalisée se verra associé un indice de confiance donné en

fonction de l'analyseur l'ayant fournie et de l'énoncé traité (Cet indice est pré-calculé par un processus d'apprentissage sur un corpus de références).

L'étape de construction se compose de deux phases :

- La phase de *fusion* qui permet, dans un premier temps, de faire *correspondre* les différentes structures normalisées fournies par l'étape précédente, et, dans un second temps, de *fusionner* les informations de ces structures, à l'aide des correspondances établies précédemment, pour donner une unique représentation de dépendances contenant toutes les informations linguistiques possibles, même les informations contradictoires. Toutes ces informations ainsi fusionnées verront leurs indices de confiance recalculés ;
- La phase de *production* qui permet de construire une ou plusieurs structures de dépendances en fonction des informations fusionnées, des indices de confiance recalculés et de contraintes linguistiques et structurelles de production.

2.2 – Architecture fonctionnelle

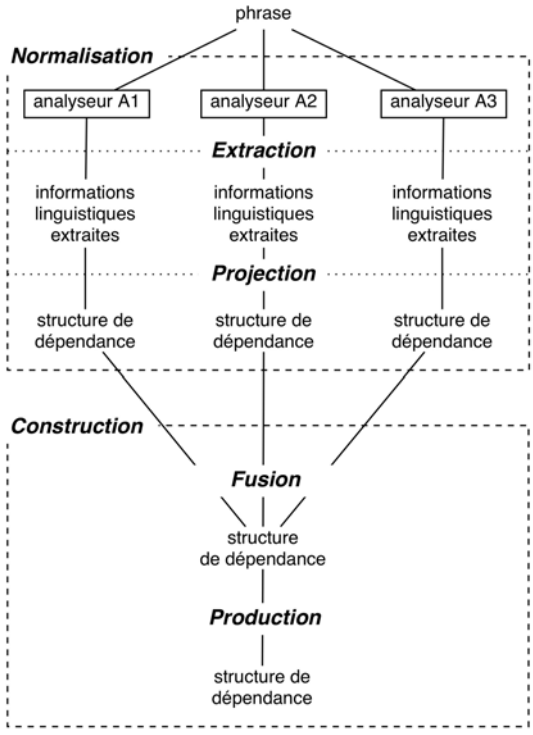


figure 1 : Architecture fonctionnelle de la plate-forme d'analyse

3 – NORMALISATION DES RESULTATS D'ANALYSE

La première étape consiste à normaliser chaque résultat d'analyse, c'est-à-dire transformer les différents résultats d'analyse obtenus (constituants et/ou dépendances) en un ensemble de structures de dépendances.

3.1 - Structure de dépendances

3.1.1 - Définition

Une structure de dépendances permet de décrire les relations entre les mots d'un énoncé. Les relations peuvent être de tous types : syntaxique, logique ou sémantique. Une structure de dépendances est composés d'un ensemble de nœuds et d'un ensemble d'arcs :

- Un *nœud* est associé à un mot d'un énoncé. Il est composé d'un ensemble d'informations linguistiques correspondant à ce mot (catégorie morphosyntaxique, variables grammaticales, variables sémantiques et variables logiques) ;
- Un *arc* entre deux nœuds correspond à une relation linguistique entre deux mots d'un énoncé (relation syntaxique, sémantique ou logique).

Exemple de structure de dépendances syntaxiques représentant la phrase (a) « le vieux chat attrape la souris. »

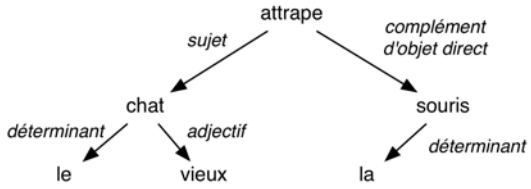


figure 2 : arbre de dépendances syntaxiques

Nous pouvons également ajouter d'autres types d'informations linguistiques à cette structure comme des informations sémantiques :

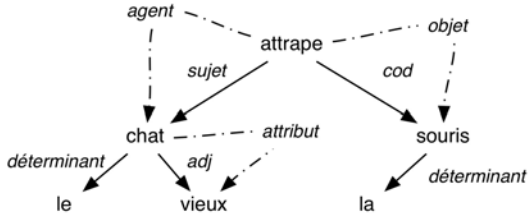


figure 3 : graphe de dépendances syntaxiques et sémantiques

3.1.2 – Matrice de dépendances (MD)

Les structures de dépendances sont décrits dans notre plate-forme par une représentation matricielle. Notre représentation, nommée *matrice de dépendances MD*, est un couple <*L, M*> composée :

- D'une *liste de nœuds (L)*, un nœud étant composé d'informations linguistiques relatives aux mots qu'il décrit ;
- D'une *matrice carrée (M)* permettant de décrire les arcs entre nœuds. La case (i, j) contient l'ensemble des arcs (relations) entre le nœud i et le nœud j de la liste de nœuds.

La représentation matricielle correspondant au graphe de dépendances multi-niveau (figure 3) de la phrase (a) sera :

L =

le	::	cat=déterminant
vieux	::	cat=adjectif
chat	::	cat=nom
attrape	::	cat=verbe, temps=présent
la	::	cat=déterminant
souris	::	cat=nom

M =

relation →

	le	vieux	chat	attrape	la	souris
le						
vieux						
chat	det	adjectif *attribut*				
attrape			sujet *agent*			cod *objet*
la						
souris					det	

Une représentation matricielle des données présente de nombreux avantages pour le traitement informatique :

- *Maniabilité* : De nombreux outils mathématiques sont associés aux matrices : ajout, suppression, comparaison, etc. Tous ces outils permettent un traitement simple de l'information contenue dans une matrice ;
- *Efficacité* : Les méthodes utilisant les matrices comme structure de données, telle que la reconnaissance de motifs ou la fusion de matrice, se montre très efficaces et très simple à mettre en place.

Notre représentation matricielle permet également de représenter différents contenus linguistiques simplement :

- *Multi-niveau* : elle peut regrouper plusieurs niveaux linguistiques (syntaxique, sémantique, logique, etc.) comme dans l'exemple précédent ;
- *Ambiguïtés* : elle peut regrouper des informations contradictoires avant la phase de production des arbres de dépendances. Par exemple, suite à la phase de fusion, deux dépendances syntaxiques, *sujet* et *objet*, entre les mots *vieux* et *chat* ont été combinés dans la MD. Ces informations seront traitées lors de la phase suivante, la phase de production (voir 4.3 - Production des arbres de dépendances).

3.2.3 – Vers une norme de description linguistique

Pour construire notre plate-forme combinant différents analyseurs, il est nécessaire de connaître les caractéristiques de chaque analyseur et particulièrement les caractéristiques de leur résultat. Pour ce faire, une première étude a été réalisée permettant de classer les différentes informations linguistiques produites par chaque analyseur. Cette classification nous permet d'établir une norme de description linguistique à utiliser lors de l'étape de normalisation des informations.

Les analyseurs syntaxiques

Pour cette première étude, nous avons utilisé 3 analyseurs du Français : IFSP (Incremental Finite-State Parser) développé par XEROX, l'analyseur du GREYC développé par Jacques Vergne et l'analyseur du projet LIDIA du GETA.

Nous pouvons distinguer deux types d'analyseurs : les analyseurs linguistiques, fondés sur des formalismes grammaticaux, et les analyseurs probabilistes, fondés sur l'apprentissage à partir de corpus. Pour notre étude et la réalisation de notre plate-forme, nous avons choisi d'utiliser dans un premier temps des analyseurs linguistiques. La plupart de ces analyseurs se répartissent en trois catégories en fonction des résultats qu'ils fournissent [Mon02] :

- Les *analyseurs fondés sur les constituants* qui retournent une segmentation en groupes.

- Les *analyseurs fondés sur les dépendances* qui retournent les dépendances entre mots d'une phrase.
- Les *analyseurs fondés sur les constituants et les dépendances* qui retournent une segmentation en groupes et des relations de dépendances entre ces groupes et/ou entre les mots.

Les 3 analyseurs utilisés pour notre étude se situent dans cette dernière catégorie.

Exemples de résultats à traiter :

Relation de dépendances :

SUBJ(Pierre, travaille)

VMODOBJ(travaille, dans, bureau)

Groupe :

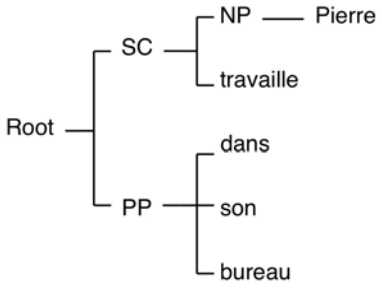


figure 4 : Résultat filtré de l'analyseur IFSP pour la phrase "Pierre travaille dans son bureau."

Groupe :

Groupe nominal : Pierre

Groupe verbal : travaille

Groupe nominal : dans son bureau
[bureau gouverneur]

Relation de dépendances :

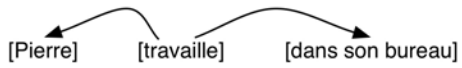


figure 5 : Résultat filtré de l'analyseur développé par J. Vergne pour la phrase "Pierre travaille dans son bureau."

La norme

Cette étude nous a permis de répertorier les différentes informations produites par les analyseurs (relations entre mots, groupements de mots, etc.) et de comparer les différentes particularités de chaque analyseur afin de vérifier la présence d'informations utiles pour réaliser une analyse de dépendances. Cette étude nous permet surtout d'évaluer la complémentarité des analyseurs et de définir certains des critères à utiliser lors de la fusion des résultats des différents analyseurs. Nous avons classé toutes ces informations linguistiques en 6 catégories. Notre norme est donc constituée de :

- *Syntagmes* : groupe nominal, groupe verbal, groupe adjectival, groupe adverbial, groupe prépositionnel, proposition relative, proposition subordonnée, etc.
- *Relations syntaxiques* : sujet, objet, gouverneur, complément d'agent, complément circonstanciel, coordination, apposition, inclusion, etc.
- *Relations logiques* (relations prédicatives) : numérotation des arguments, etc.
- *Relations sémantiques* : contexte, but, cause, conséquence, instrument, bénéficiaire, conséquence, manière, matière, lieu, qualification, etc.
- *Catégories morpho-syntaxiques* : nom, verbe, adjectif, adverbe, subordonnant, coordonnant, ponctuation, mot inconnu, pronom, préposition, déterminant, participe, etc.
- *Variables grammaticales* : genre, nombre, temps, mode, personne, etc.

	Analyseur Vergne	Analyseur Xelda	Analyseur Lidia
Syntagmes			
Groupe nominal	x	x	x
Groupe verbal	x	x	x
Groupe prépositionnel	x	x	x
Groupe adjectival		x	x
Groupe adverbial			x
Proposition relative	x		x
Proposition infinitive			x
Proposition participale	x	x	x
Proposition subordonnée	x		x
Relations syntaxiques			
Sujet	x	x	x
Objet	x	x	x
Complément d'agent			x
Complément circonstanciel			x
Coordination	x		x
⋮			

figure 6 : Extrait d'un tableau de classement

Nous avons cherché à détailler chacune de ces six catégories puis de faire correspondre, si possible, à chaque information détaillée les informations fournies par les analyseurs (voir figure 6). Ceci nous permet de voir plus clairement quelles informations sont disponibles, quelles informations pourront être contradictoires et surtout quelles informations seront complémentaires.

3.3 – Extraction des informations contenues dans les résultats d'analyse

La première phase consiste à extraire les informations des résultats fournis par chaque analyseur. Dans un premier temps, il faut être capable de décrire le format de représentation de ces résultats, dans notre cas nous décrivons ce format sous la forme d'une grammaire JAVACC (un générateur d'analyseurs syntaxiques pour JAVA), nommé grammaire de description. Puis à l'intérieur de cette grammaire, nous introduisons des méthodes d'extraction en fonction de la catégorie du résultat (constituant et/ou dépendances) :

- AjoutRelation(TypeRelation, Mot1, Mot2)
- AjoutSyntagme(TypeSyntagme, Ensemble de Mots)
- AjoutCategorie(Mot, Information Linguistique).

3.3 – Projection des informations extraites

Après avoir extrait toutes les informations linguistiques des résultats, la seconde phase consiste à *normaliser* toutes ces données, c'est-à-dire les faire correspondre à une structure de dépendances (un ensemble de relations entre nœuds) normalisée, correspondant à la norme établie précédemment.

3.3.1 – Règles de normalisation

Pour normaliser toutes ces informations extraites de chaque analyseur, nous nous aidons du tableau de classement décrit précédemment (voir figure 6). Ce tableau fait correspondre à chaque donnée extraite son information normalisée (catégorie, variable grammaticale ou relations entre deux nœuds). Par exemple, pour le résultat fourni par IFSP, la relation SUBJ sera normalisée pour correspondre à la relation SUJET et ainsi de suite pour toutes les relations et les catégories simples à normaliser. Mais ce tableau devra également contenir, pour certaines informations, des renseignements supplémentaires concernant la normalisation de celle-ci. Par exemple, la relation SUBPASS (sujet passif) extraite de l'analyse de IFSP pourra être transformée en une relation sujet grâce à la règle de normalisation qui permet d'inverser les nœuds de la relation :

```
SUBPASS($var1[1], $var2) := SUJET($var2, $var1)
```

D'autres relations posent des problèmes plus difficiles par exemple la relation VMODOBJ extraite de l'analyse de IFSP pourrait avoir plusieurs significations syntaxiques : complément circonstanciel, complément d'objet direct, etc. L'idée est de fournir toutes les projections possibles vers notre norme et d'associer à chacune de ces informations normalisées un indice de confiance[2]:

```
VMODOBJ($var1, $prep, $var2) :=
 COMPLEMENT_CIRCONSTANCIEL($var1, $var2)
 indice_de_confiance = 0,8

VMODOBJ($var1, $prep, $var2) :=
 COMPLEMENT_OBJET_DIRECT($var1, $var2)
 indice_de_confiance = 0,2
```

Cet indice sera pré-calculé à l'aide d'une phase d'apprentissage réalisée à partir d'un corpus de référence[3]. Cet indice exprime la confiance relative à l'information en fonction de l'analyseur et de la typologie de l'énoncé. Cet indice sera recalculé par la suite lors de la phase de fusion (voir 4.2 – Fusion des informations linguistiques).

3.3.2 - Première expérimentation

Cette expérimentation nous a surtout permis de définir clairement la spécification et les différents processus à utiliser pour concevoir notre plate-forme.

Nous avons réalisé une première maquette de notre outil ne comportant que les deux premiers phases : extraction et projection en utilisant une représentation classique en arbre pour représenter nos données. Cette maquette

[1] La variable $var représente soit un mot soit un syntagme.

[2] L'indice varie de 0 à 1, 1 étant l'indice de confiance maximum.

[3] Le module d'apprentissage est en cours de réalisation et sera basé sur une méthode de correction d'indices utilisant des techniques de rétro-propagation de données. Pour le moment, les indices de confiance sont fixés manuellement.

ne charge que deux analyseurs : l'analyseur de la plateforme Xelda et l'analyseur développé par Jacques Vergne. Cette première réalisation avait pour but de vérifier nos hypothèses de départ, c'est-à-dire : la possibilité d'extraire facilement de l'information et de la normaliser, ainsi que la possibilité d'intégrer facilement de nouveaux analyseurs à une plateforme d'analyse [Bru03].

Le corpus de phrases utilisé provient d'un corpus UNL, Universal Networking Language, contenant une phrase et son hypergraphe UNL associé. Ce corpus était constitué de quarante phrases courtes, neuf mots par phrase en moyenne (minimum 3 mots, maximum 56 mots), décrivant de nombreux phénomènes linguistiques : coordination, négation, relative, etc. Par exemple :

- Une tulipe est plus belle qu'une rose.
- Il sait que tu ne viendras pas et il ne le regrette pas.

Dans un premier temps, nous avons donc traité ces quarante phrases avec l'analyseur de Xelda et avec l'analyseur développé par J. Vergne. Ce premier traitement, appliqué à toutes les phrases du corpus, nous a permis de déterminer quelques règles de normalisation à associer aux informations fournies par les analyseurs. Pour cette expérimentation, nous avons 15 règles de normalisation associées à chaque analyseur concernant essentiellement les relations syntaxiques.

Pour intégrer les deux analyseurs à notre plate-forme, nous avons donc réalisé leurs grammaires de description puis associé les méthodes d'extraction. Les règles de normalisation pour chaque analyseur ont ensuite été développées. Notre outil a extrait toutes les informations linguistiques fournies par les résultats d'analyse. Avec seulement 15 règles de normalisation, notre outil a déterminé 78% des liaisons entre mots (relations non étiquetées) et 67% de ces relations ont pu être étiqueté. Les premiers résultats normalisés fournis pour notre outil sont encourageants, même s'ils restent limités à quelques phrases et quelques règles de normalisation, mais surtout les hypothèses de départ ont pu être vérifiées.

Mais plusieurs problèmes se sont présentés, tout d'abord, la lourdeur de la représentation arborescente utilisée. En effet, nous nous sommes vite rendu compte de l'utilité de changer de représentation pour les deux premières phases mais surtout pour la future phase de fusion et d'adopter la représentation de type matricielle présentée précédemment, plus maniable et plus adaptée à nos besoins (voir 3.1.2 – Matrice de dépendances (MD)). Le second problème concerne le besoin d'informations linguistiques supplémentaires dans le cas des analyseurs fondés sur les constituants. Comment trouver les relations entre mots à partir des syntagmes si aucune information n'est pas fournie concernant le gouverneur du syntagme. Dans le cas de l'analyseur développé par J. Vergne, l'information concernant le gouverneur du syntagme est fournie mais pas dans le cas de l'analyseur de Xelda. Nous aurons donc besoin d'une base de connaissances linguistiques simple permettant de résoudre ce genre de problèmes.

4 – CONSTRUCTION D'UNE ANALYSE DE DEPENDANCES

À la fin de l'étape de normalisation, un ensemble d'arbres de dépendances est associé à chaque phrase. La phase suivante consiste à fusionner toutes ces structures arborescentes pour obtenir une unique représentation de dépendances contenant toutes les informations linguistiques présentes dans ces structures (catégories, variables grammaticales, relations). Pour réaliser cette fusion, il faut, dans un premier temps, mettre en *correspondance* ces structures arborescentes. En effet, l'un des problèmes rencontré lors de la fusion est le problème des différentes segmentations possibles fournies par chaque analyseur pour une phrase donnée.

4.1 - Correspondance de structures

4.1.1 - Arbre de dépendance étendu

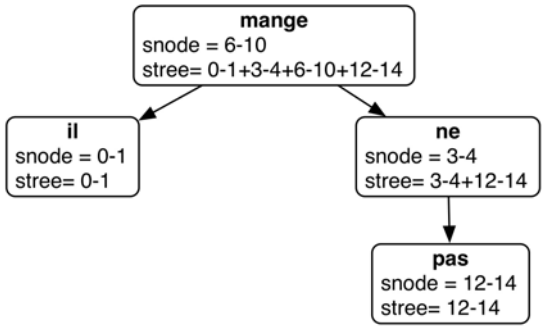
Pour réaliser la correspondance entre plusieurs arbres de dépendance, il faut être capable de déterminer si deux nœuds de structures différentes représentes la même segmentation. Pour se faire, nous étendons la définition de nos arbres de dépendance en nous basant sur la proposition de [Boi88], *Structured String-Tree Correspondences (SSTC)*. Cette proposition nous permet de réaliser la correspondance entre la phrase et l'arbre de dépendance associé et, par extension, nous permet de déterminer facilement si deux nœuds

correspondent à la même segmentation dans la phrase (même chaîne).

Nous ajoutons donc à la structure de dépendance des informations relatives à la correspondance entre la phrase et la structure arborescente associée. Nous attachons à chaque nœud N de l'arbre deux séquences d'intervalles :

- *SNODE(N)* : séquence d'intervalles représentant la sous-chaîne correspondante au nœud N.
- *STREE(N)* : séquence d'intervalles représentant les sous-chaînes correspondantes aux nœuds contenu dans le sous-arbre ayant comme tête le nœud N.

Pour la phrase « Il ne mange pas. », un arbre de dépendance étendu possible sera :



La plupart du temps, les analyseurs fournissent assez d'informations relatives aux mots pour que la correspondance chaîne-nœud se réalise sans difficulté : unité lexicale (chaîne), lemme et ses variables grammaticales associées, position dans la phrase.

Les informations SNODE et STREE nous permettrons également de vérifier certaine propriété relative aux arbres de dépendance lors de la dernière phase du traitement, la production (voir 4.3 - Production des arbres de dépendances) :

L'arbre est *non-recouvrant* si :

- STREE(N1) et STREE(N2) ont une intersection vide si N1 et N2 sont indépendant.
- SNODE(N1) et STREE(N2) ont une intersection vide si N2 est un fils de N1.

L'arbre est *projectif* si :

- L'arbre est *non-recouvrant*.
- Pour tous nœuds frères N1 et N2, N1 étant à gauche de N2, STREE(N1) est complétement à gauche de STREE(N2). Cela signifie :

Si STREE(N1) = w(i1_j1)…w(ip_jp) ou ∅
et STREE(N2) = w(k1_l1)…w(kq_lq) ou ∅
alors jp<=k1

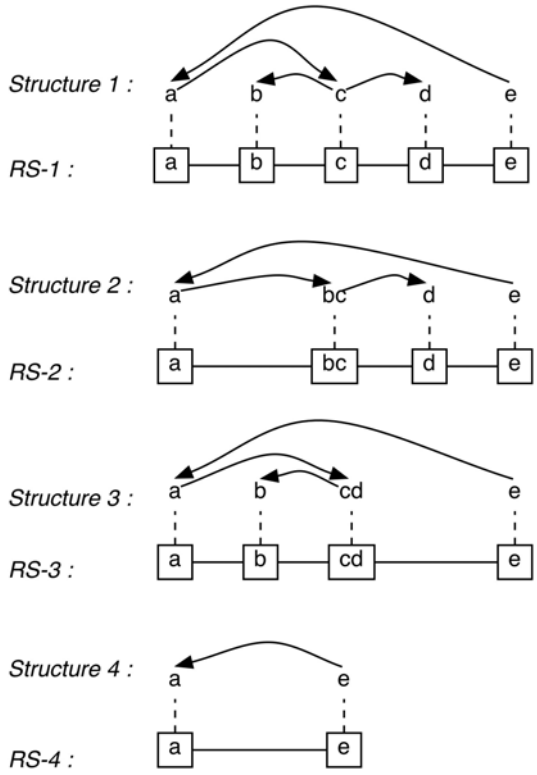
4.1.2 - Réseau de segmentation

La correspondance de structures consiste à regrouper les nœuds représentant la même segmentation dans la phrase. Mais elle consiste également à représenter les discordances issues des différentes segmentations.

Pour ce faire, nous créons une structure, appelée *réseau de segmentation (RS)*, représentant les différentes segmentations de la phrase et permettant de lier les nœuds des structures normalisées. Ce réseau peut être vu comme un « *pivot de liaison* » entre ces structures. Ce réseau est un treillis, chaque nœud du réseau représentant une segmentation possible d'un mot et servant de liaison entre les nœuds des structures de dépendances. Concrètement, un nœud Nrs d'un RS contient deux informations :

- SNODE(Nrs) : séquence d'intervalles représentant la segmentation dans la phrase, basée sur la proposition de [Boi88] *Structured String-Tree Correspondences* (SSTC).
- L : un ensemble contenant les nœuds des structures normalisées liés au nœud Nrs.

La première étape consiste à créer un réseau de segmentation initial pour chaque arbre de dépendances. Chaque nœud Nrs du RS initial est créé en fonction d'un nœud N1 de l'arbre A1 : SNODE(Nrs) = SNODE(A1.N1) et L(Nrs) = {A1.N1}. Les nœuds du RS seront insérés dans le treillis selon l'ordre d'apparition dans la phrase (en fonction du SNODE). Dans la suite du traitement, nous prendrons comme exemple les quatre arbres de dépendances et leurs RS initiaux suivants :



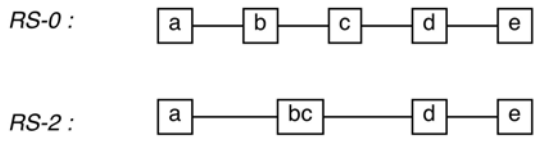
Le premier réseau initial RS-1 est désigné comme le réseau de base, RS-0, qui servira tout au long du traitement. La suite consiste à introduire les particularités des autres RS initiaux dans le réseau de base. Pour ce faire, nous utilisons deux règles de construction :

- **Règle 1)** *Correspondance* : Si le nœud Ni de RS-i est *équivalent* à l'un des nœuds N0 de RS-0 (l'équivalence est vraie si SNODE(Ni) == SNODE(N0)), N0 sera lié aux nœuds de la structure que représente Ni :

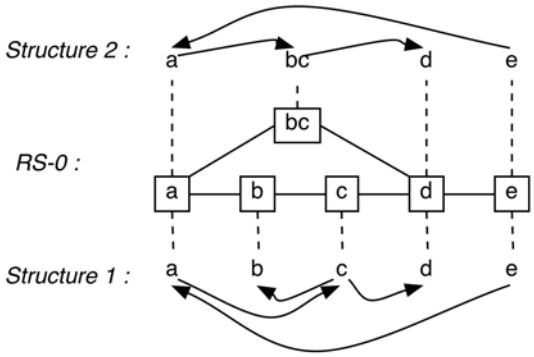
  $$L(N0) = L(N0) \cup L(Ni).$$

- **Règle 2)** *Insertion* : Si le nœud Ni de RS-i n'est équivalent à aucun nœud de RS-0, le nœud Ni est inséré dans RS-0 en fonction de son SNODE (ordre d'apparition dans la phrase).

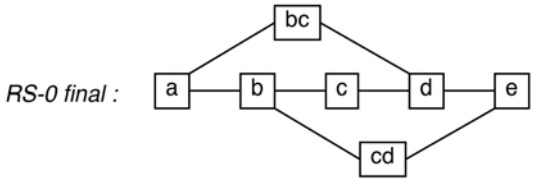
Faisons la correspondance entre RS-0 et RS-2 :



Le premier nœud *a* de RS-2 vérifie la première règle, il correspond au nœud *a* de RS-0. Le second nœud *bc* de RS-2 vérifie la seconde règle, il est donc inséré dans RS-0. Le reste des nœuds de RS-2 vérifie la première règle. Nous obtenons donc le réseau suivant :



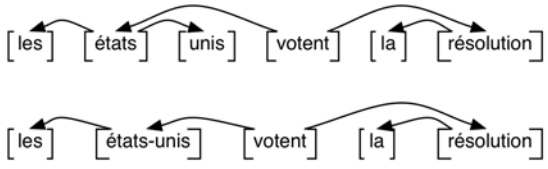
Le traitement se répète en ajoutant RS-3 et RS-4. Le réseau de correspondance final sera donc :



Le réseau de segmentation final obtenu représente les segmentations possibles et lie les nœuds des structures entre eux. Maintenant que la correspondance entre les nœuds des structures est établie, nous pouvons fusionner ces structures pour fournir une unique représentation de dépendances combinant toutes les informations linguistiques relatives aux structures : catégories, variables grammaticales et relations (voir 4.2 – Fusion des informations linguistiques).

4.1.3 - Traitement des mots composés

Une segmentation discordante entre structures peut provenir d'un mot composé. Un mot composé est assimilé à une différence de segmentation si, comme dans l'exemple ci-dessous, un autre analyseur ne reconnaît pas la composition du mot :



Nous voulons que notre plate-forme soit capable de déterminer la correspondance entre le mot composé et le sous-arbre fourni par un autre analyseur. La solution envisagée est de définir une *règle de correspondance (RC)* qui décrira des contraintes linguistiques et

structurelles permettant de déterminer une telle correspondance.

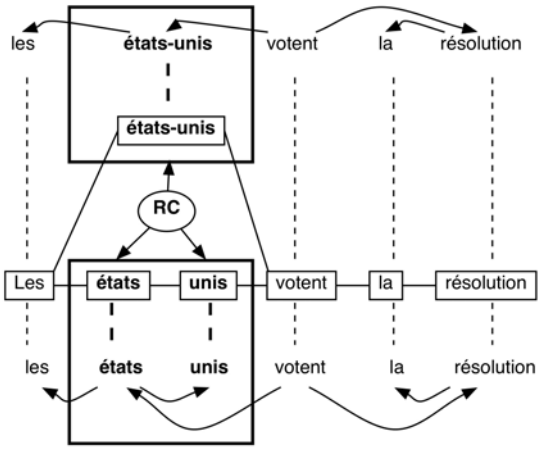
Une règle de correspondance est constituée :

- D'un modèle de correspondance permettant de déterminer les nœuds des structures à traiter (en rapport avec les nœuds du RS).
- De contraintes linguistiques sur les nœuds et les arcs des structures.
- De contraintes structurelles

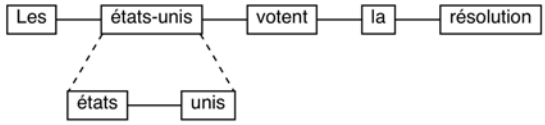
Exemple simple de règle de correspondance permettant de vérifier l'exemple pécédent :

```
// description du modèle
RC[Nmc ⇔ (N1,N2)] =

    // contraintes linguistiques
    Nmc.catégorie == "nom" ;
    N1.catégorie == "nom" ;
    N2.catégorie == "adjectif" ;

    // N2 dépendant de N1
    N1 → N2 ;

    // N2 dépendant d'aucun autre noeud
    N* → N2, N*/N1 = ∅ ;

    // N2 n'a pas de dépendants
    N2 → N*, N* = ∅ ;
```

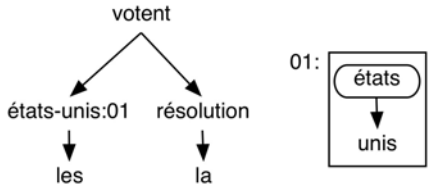
Les règles de correspondance s'appliquent sur les nœuds discordants du réseau de segmentation. Ci-dessous, un exemple d'application de la règle décrite précédemment :



Si une règle est vérifiée, la correspondance est établie et le réseau de segmentation sera modifié pour prendre en compte cette nouvelle information. Nous ajouterons une dimension supplémentaire au réseau établissant ainsi une correspondance particulière entre ces nœuds :



Ce type d'information nous permettra, lors de la phase de fusion, de combiner les informations relatives aux deux sous-structures mis en correspondance, mais également, lors de la phase de production, de fournir une information structurelle sur le mot composé. Par exemple, pour l'exemple vu précédemment, la plate-forme d'analyse fournira comme résultat : (01 étant la référence du sous-arbre de tête le nœud *état*)



4.2 – Fusion des informations linguistiques

Les correspondances entre les différentes structures étant établies, la fusion des informations. La méthode utilisé lors de cette phase est basée sur la méthode dite de « vote à la majorité » : *plus une information sera commune aux différents analyseurs, plus son poids augmentera*, dans notre cas plus son indice de confiance augmentera. Chaque indice pourra être vu comme le *vote pondéré* de l'analyseur pour l'information, ce vote étant *adapté* aux différentes possibilités de l'analyseur en fonction de l'énoncé (par exemple, entrée bruitée ou non) lors de la phase d'apprentissage.

À la fin de la phase de correspondance, à chaque phrase est associée un ensemble de matrices de dépendances et un réseau de segmentation permettant de les lier. La suite du traitement consiste à créer pour chaque réseau de segmentation une matrice de dépendances, nommée *matrice de fusion*, (les nœuds du RS serviront de nœuds pour cette représentation), puis à la compléter en fusionnant toutes les informations linguistiques contenues dans les matrices de dépendances associées. Certaines

informations linguistiques seront équivalentes, d'autres contradictoires. Il ne s'agit pas simplement de regrouper toutes ces informations, il faut également calculer de nouveaux indices de confiance, *les indices de fusion*, pour chaque information en fonction des indices de confiance fournis par l'étape de normalisation.

Calcul simple[4]

L'indice de fusion de l'information I pour un nœud ou un arc donné est égal à la somme des indices de confiance des informations I pour ce nœud ou cette arc :

$$\text{indice(I)}_{\text{fusion}} = \sum_{\substack{\text{i = analyseur fournissant} \\ \text{l'information I}}} (I_i)$$

Par exemple, calculons l'indice de fusion à associer à l'information catégorie ADJ pour le mot x, fournie à la fois par l'analyseur A1 et par l'analyseur A2. L'indice de fusion associé à ADJ(x) est égal à la somme des deux indices de confiance indice(ADJ::A1)=0,6 et indice(ADJ::A2)=0,8. L'indice de fusion associé à ADJ(x) est égal à (0,6+0,8) = 1,4.

Calcul normalisé

L'indice de fusion de l'information I est égal à la somme des indices de confiance divisé par le nombre d'analyseurs pouvant fournir cette information :

$$\text{indice(I)}_{\text{fusion}} = \frac{\left(\sum_{\substack{\text{i = analyseur fournissant} \\ \text{l'information I}}} (I_i) \right)}{n}$$

Par exemple, calculons l'indice de fusion à associer à l'information relation OBJ entre les mots x et y, fournie à la fois par l'analyseur A1 et par l'analyseur A2. L'indice de fusion associé à OBJ(x,y) est égal à la somme des deux indices de confiance indice(OBJ::A1)=0,5 et indice(OBJ::A2)=0,7 divisée par le nombre d'analyseurs pouvant fournir ce type d'information (ici trois pour l'exemple), (0,5+0,7+0)/3 = 0,4. Si le troisième analyseur fournit une information de type SUBJ entre les mots x et y, et que l'indice de confiance relatif à cette information est de 0,8. L'indice de fusion associé à SUBJ(x,y) est égal à (0+0+0,8)/3 = 0,26.

Calcul corrigé[5] **:**

L'indice de fusion de l'information I est égal à la somme des indices de confiance des informations I moins la somme (multipliée par un coefficient de correction) des indices de confiance des informations contradictoires à l'information I, le tout divisé par le nombre d'analyseurs pouvant fournir l'information I :

$$\text{indice(I)}_{\text{fusion}} = \frac{\left(\sum_{\substack{\text{i = analyseur fournissant} \\ \text{l'information I}}} (I_i) - \beta \times \sum_{\substack{\text{p = analyseur fournissant une} \\ \text{information contradictoire à I}}} (I_p) \right)}{n}$$

Pour l'exemple précédent, les relations syntaxiques entre les mots x et y sont contradictoires : soit OBJ(x,y), soit SUBJ(x,y). L'indice de fusion associé à OBJ(x,y) est égal à ((0,5+0,7) – (0,4 * 0,8))/3 = 0,29. (en prenant comme coefficient de correction 0,4) et l'indice de fusion associé à SUBJ(x,y) est égal à (0,8 – 0,4 * (0,5+0,7))/3 = 0,1

On peut constater que ces calculs favorisent les informations fournies par le plus grand nombre d'analyseurs. Ces nouveaux indices de fusion ainsi calculés serviront lors de la phase de production.

4.3 - Production des arbres de dépendances

Cette dernière phase permet d'obtenir une ou plusieurs structures de dépendances grâce à toutes les informations recueillies. Les structures de dépendances résultats sont produites à partir des indices de confiance associés aux informations, de contraintes linguistiques et de contraintes structurelles.

Nous pouvons voir cette phase comme une phase de satisfaction de contraintes comportant 4 règles :

- Seuls les indices de confiance élevés seront conservés (au-dessus d'un certain seuil).
- Aucune discordance de segmentation entre les nœuds.
- Respect des contraintes linguistiques imposées par l'utilisateur pour éviter les contradictions.

[4] n : nombre d'analyseurs pouvant fournir l'information I

[5] β : coefficient de correction

- Respect des contraintes structurelles, comme la projectivité de la structure résultat et le fait de n'avoir qu'une et une seule relation de même type (syntaxique, sémantique, logique) entre deux nœuds.

Notre plate-forme produira donc plusieurs structures résultats pour un énoncé, à chaque structure résultat sera associé un indice de pertinence, moyenne pondérée des indices de fusion présents dans la structure.

5 - PERSPECTIVES

Une seconde maquette est en cours de réalisation et sera évaluée sur 5 analyseurs : l'analyseur de la plateforme Xelda, l'analyseur Xip développé par Xerox, l'analyseur developpé par J. Vergne, l'analyseur du projet Lidia, l'analyseur developpé par J. Chauché. Cette nouvelle maquette intégrera tous les modules et fournira en résultat une ou plusieurs structures de dépendances.

À plus long terme, nous comptons utiliser notre plate-forme d'analyse de dépendances, pour générer des hypergraphes UNL (Universal Networking Language). Un hypergraphe décrit le *sens* de l'énoncé dans un contexte donné. Il est composé d'arcs représentant des relations sémantiques (tels qu'agent, objet, but,etc.) et de nœuds représentant les UW (« Universal Word », ou acceptions interlingues) auxquelles sont associés des attributs sémantiques [Sér00]. Pour générer ce type de graphe, la structure de dépendances fournie par notre plate-forme d'analyse nous semble particulièrement adéquate, vu qu'elle représente également un ensemble de relations entre mots. Une telle structure associée à un dictionnaire spécifique nous permettra dans un premier temps de générer des hypergraphes simples, composés d'informations syntaxiques et sémantiques de surface.

REMERCIEMENTS

Je tiens à remercier XEROX et Jacques Vergne pour m'avoir permis d'utiliser leurs analyseurs.

BIBLIOGRAPHIE

[Boi88] Boitet Ch. And Zaharin Y. (1988), "Representation trees and string-tree correspondences", published in COLING-88, pp 59-64

[Bru03] Brunet-Manquat F. (2003), "Fusionner pour mieux analyser: quelques idées et une première expérience", published in RECITAL'03

[Fis97] Fiscus J.G. (1997), "A post-processing system to yield reduced error word rates: Recognizer output voting error reduction (ROVER)", published in IEEE Workshop on Automatic Speech Recognizer and Understanding, pp 347-354

[Ill99] Illouz G. (1999), "Méta-étiqueteur adaptatif: vers une utilisation pragmatique des resources linguistiques", published in TALN'99

[Mon02] Monceaux L., Isabelle Robba I. (2002), "Les analyseurs syntaxiques : atouts pour une analyse des questions dans un système de question-réponse ? ", Actes de *TALN'2003*, pp.195-204.

[Sch00] Schwenk H. and Gauvain J.L. (2000), "Combining multiple speech recognizers using voting and language model information", published in IEEE International Conference on Speech and Language Processing (ICSLP), pp. II:915-918

[Sér00] Sérasset G., Boitet Ch. (2000), "On UNL as the future "html of the linguistic content" & the reuse of existing NLP components in UNL-related applications with the example of a UNL-French deconverter", in *COLING 2000*, Saarebruecken, Germany.