Université Pierre Mendès France
Master2 Professionnel ICA-ICPS

Stage chez Xerox
du 3 avril au 29 septembre 2006

Responsable : Anne Schiller

XEROX®
Research Centre Europe

Développement d'une grammaire de l'allemand pour XIP

Rapport de stage

Sigrid Maurel

Sigrid.Maurel@xrce.xerox.com

Table des matières

Introduction

Dans le cadre de la formation du Master 2 professionnel ICA-ICPS[1] de l'université Pierre Mendès France (Grenoble II), nous sommes amenés à réaliser un stage en entreprise d'une durée de trois mois minimum pour acquérir une expérience professionnelle.

Je travaille depuis le 3 avril 2006 au sein de l'équipe ParSem - Parsing and Semantics. ParSem fait partie du XRCE - Xerox Research Centre Europe, qui est un laboratoire de recherche de l'entreprise américaine Xerox, à Meylan en France.

Je fais partie d'un projet plus vaste qui consiste à développer des grammaires en plusieurs langues (allemand, italien, espagnol et portugais) pour l'analyseur syntaxique *XIP* (Xerox Incremental Parser). Ces grammaires générales (et celle du français qui existe déjà depuis un moment) doivent ensuite être adaptées pour une application de recherche d'information (*Pocket Engineer (PE)*) pour les bases de données d'aide en ligne des imprimantes de Xerox.

Mon stage durera en tout six mois (jusque fin septembre), c'est pour cette raison que le travail a été divisé en deux parties. J'ai commencé par le développement de la grammaire (générale) pour l'allemand, et ensuite je vais travailler sur la spécification de la grammaire pour l'application PE et faire une évaluation des deux parties.

Ce rapport comporte donc également deux parties, la première traitant du travail effectué jusqu'à maintenant (fin juin). Je commence avec une présentation de l'entreprise d'accueil, puis je donne une introduction à XIP, l'outil avec lequel j'ai travaillé pour développer la grammaire. Mon travail proprement dit sera détaillé ensuite, et dans la deuxième partie je vais décrire ce qui reste à faire, notamment pour PE, pendant les mois à venir.

Le présent document se termine avec une conclusion, les perspectives pour la deuxième partie du stage et un annexe qui contient une partie des règles que j'ai développées et une sortie typique de XIP d'une petite phrase allemande. Il a pour objet de rapporter l'ensemble des activités et travaux menés à bien lors du stage de fin d'études réalisé.

[1] Master d' Information, Cognition et Apprentissage, spécialité Ingénierie de la Communication Personne-Système

1 Contexte de travail

Cette section situe mon stage au sein de l'équipe ParSem du centre de recherche européen Xerox et donne un aperçu de l'entreprise. La première partie est consacrée à l'entreprise Xerox, et la deuxième au centre de recherche XRCE, où je présenterai également l'équipe ParSem.

À la fin de cette partie se trouve le cahier des charges avec les tâches que j'ai réalisées pour ce projet.

1.1 Xerox

Xerox est une société américaine qui produit principalement des imprimantes (couleur et noir/blanc), faxes et photocopieurs. Elle est implantée un peu partout dans le monde. Le siège aux États-Unis est à Stamford, dans l'État du Connecticut, et le siège européen n'est pas loin de Londres, en Angleterre.

Cette entreprise a été créée en 1906 (**The Haloid Company**), appelée **Xerox Corporation** en 1961. La société s'est installée en Europe sous le nom de **Rank Xerox** en 1956 et est devenue **Xerox** en 1997.

Ses principales activités sont centrées autour des technologies de traitement du document et des services de conseil associés. On distingue notamment :

- imprimantes de bureau (couleurs et noir & blanc),
- outils multifonctionnels et copieurs numériques,
- presses numériques pour l'impression de masse,
- fournitures, logiciels et maintenance.

En 2005, Xerox employait près de 58 000 personnes dans le monde entier (à peu près 32 000 aux États-Unis, 16 000 en Europe et 9 000 dans le reste du monde) pour un chiffre d'affaires de 15,7 milliards de dollars.

Xerox est une entreprise de technologies et de services, leader dans son domaine et plus particulièrement quand il s'agit de la mise en œuvre de stratégies pour une gestion plus intelligente de documents. L'objectif principal est de rendre plus facile aux clients la façon de travailler avec eux (ie. des réponses plus rapides, une seule interface, ...).

Xerox fait partie des développeurs les plus importants dans le monde entier en ce qui concerne les nouvelles technologies. L'entreprise entretient des laboratoires de recherche aux États-Unis, au Canada et en Europe, et investie environ un milliard de dollars américains chaque année pour des activités de développement et de recherche.

Les noyaux de la recherche sont des thèmes comme la science des couleurs, les procédures d'illustration numériques, les méthodes de travail, les systèmes

électromécaniques, la recherche de matériaux, ainsi que d'autres disciplines qui sont en relation avec la gestion de document et leur impression.

Les résultats des travaux de recherche de Xerox sont coordonnés étroitement avec les activités d'affaires de l'entreprise. Ainsi les innovations des laboratoires de recherche Xerox sont intégrées dans de nouveaux produits et solutions.

1.2 XRCE

Le **XRCE** - Xerox Research Centre Europe, situé pas loin de Grenoble à Meylan en France, est le centre de recherche européen de la société américaine **Xerox**. Il travaille à l'élaboration d'outils permettant la manipulation efficace du document, quel qu'il soit, en vue d'en faciliter l'accès, l'usage et le partage. XRCE nourrit en effet une vision du document où ni le langage, ni la localisation, ni le support de ce dernier ne font obstacle à l'accès à l'information et au savoir qu'il contient, se situant ainsi au plus près des besoins d'une société se dirigeant vers le tout numérique.

Il fait partie du **XIG** (Xerox Innovation Group) qui emploie environ 500 personnes dans la recherche et coopère beaucoup avec d'autres groupes d'entreprise (p.e. Fuji). Environ 5% du revenu sont investi dans XIG, actuellement plus de 16 000 brevets ont été déposés.

XRCE a été créé en 1993 et emploie actuellement environ 100 personnes, réparties en plusieurs groupes de travail, dont **ParSem** (Parsing and Semantics) au sein duquel j'effectue mon stage. Il est en coopération avec d'autres centres de recherche, notamment à Grenoble comme l'INRIA, le Synchroton, le CEA et les universités, et aussi avec l'industrie (Caterpillar, Sun, HP, Bull, ...).

Les différents domaines de recherche sont les suivants :

- Image Processing : fait du traitement de l'image, par exemple pour des systèmes de *machine learning* (apprentissage automatique) en se servant de la catégorisation d'images.
- Document Structure : travaille sur la transformation d'une information sans structure en un document structuré (p.e. en XML).
- Work Practice Technology : un groupe multidisciplinaire qui développe de nouvelles technologies et services basés sur des études extensives du poste de travail.
- Learning & Content Analysis : ce groupe développe une nouvelle génération d'outils de prognostiques, diagnostiques, visualisation, ... pour des documents mono- ou multilingues.
- **Parsing & Semantics** : s'occupe de la reconnaissance d'entités, en passant par l'extraction d'évènements, jusqu'à la caractérisation fine

d'un texte.

Les employés du XRCE sont d'une multitudes de nationalités, l'anglais comme langue de communication est vraiment omniprésent. Le centre est très ouvert et invite des chercheurs et des professionnels pour présenter leurs travaux. Le XRCE donne aussi à beaucoup d'étudiants (pas seulement de Grenoble, mais un peu de partout) la possibilité d'effectuer leur stage de fin d'études dans son sein.

Vers la fin de la première partie de mon stage un *Intern Day* (journée des stagiaires) a eu lieu. Pendant cette journée chaque stagiaire a présenté, en anglais, son projet. Comme je faisais partie d'un projet en groupe, nous avons fait une présentation à quatre, où chacun a parlé d'un aspect précis de la problématique.

Il y a une fois par mois à peu près un *General Meeting* durant lequel tous les membres du XRCE se réunissent. C'est le moment de discuter de nouveaux projets, de présenter les brevets obtenus et d'accueillir les nouveaux arrivants.

Le schéma de la figure 1 montre l'organisation du XRCE.

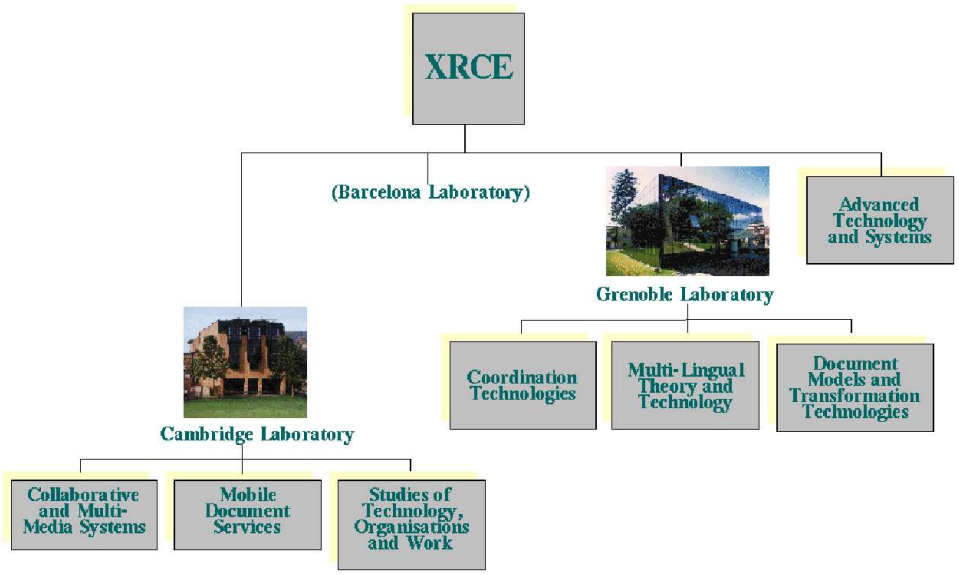


Fig. 1 – Ce schéma montre l'organisation du XRCE, le centre de recherche européen de Xerox. Le laboratoire de Grenoble regroupe la plupart des chercheurs. (Source : Intranet du XRCE)

1.2.1 ParSem

Le groupe **ParSem** - Parsing and Semantics (analyse syntaxique et sémantique) est un groupe de recherche qui s'est focalisé sur le traitement de documents (écrits) à l'aide de logiciels de syntaxe et de sémantique. Il regroupe une dizaine de chercheurs (et presque autant de stagiaires et doctorants) de nationalités différentes.

Le groupe est divisé en chercheurs qui font surtout de la linguistique et en informaticiens qui développent des logiciels, le plus connu étant XIP, un analyseur syntaxique que je vais décrire en détail dans la section 2. Mais il y a d'autres applications qui ont été développées ici, comme la recherche d'information dans un texte, la création automatique d'un texte multilingue et l'extraction d'entités nommées.

En vue d'élaborer des solutions de compréhension automatique du contenu du document, ParSem œuvre à la découverte et à l'amélioration de théories, de méthodes et d'outils de traitement du langage naturel. Ces solutions peuvent être relatives à l'examen de la pertinence d'un document par rapport à une information donnée (classification et catégorisation de document, recherche d'information), à la compréhension détaillée du texte (analyse syntaxique, extraction d'entités, détermination de relations diverses) ou encore à la modélisation du contenu du document, toujours selon l'objectif final de la compréhension poussée du texte.

Ces réalisations font appel à plusieurs compétences de différents domaines, parmi lesquelles il importe de signaler les automates à états finis (*Finite State Technology*), l'analyse syntaxique robuste et l'analyse sémantique (*Robust Parsing and Semantics*) tout comme l'apprentissage automatique (*Machine Learning*).

ParSem se concentre sur la compréhension automatique des documents électroniques, en les analysant sémantiquement. Le groupe se focalise sur deux lignes principales de recherche de traitement de langage naturel : analyse robuste et sémantique.

Mon stage (développement d'une grammaire) se situe dans la première ligne de recherche.

Les membres de ParSem participent régulièrement à des conférences et invitent des gens de l'extérieur pour rester au courant de ce qui se fait ailleurs dans leur domaine de recherche.

Chaque semaine a lieu une réunion de groupe où on parle du travail effectué et des projets à venir. Ceci permet au groupe de savoir à peu près sur quels problèmes les autres travaillent en ce moment et de connaître les nouveautés des différentes conférences auxquelles certains membres du groupe ont assisté.

En plus de ces réunions de groupe nous nous sommes réunis aussi une fois par semaine pour parler de l'avancement des grammaires. De cette façon chaque stagiaire a pu profiter des solutions des autres. Nous avons discuté ensemble les problèmes rencontrés et réfléchi comment réaliser l'application Pocket Engineer.

1.3 Présentation et objectif du stage

Mon stage a été découpé en deux parties, en conséquence de sa longueur. La première partie (de avril à juin) comprend surtout la partie développement de la grammaire, et la deuxième partie (de juillet à septembre) sera consacrée à l'évaluation de la grammaire et à l'adaptation pour une application appelée Pocket Engineer, dont je parlerai plus bas.

La tâche principale de ce stage consistait à développer une grammaire XIP de l'allemand pour l'application Pocket Engineer. Une grammaire XIP, comme je montrerai plus en détail dans la section suivante, consiste en un ensemble de fichiers qui à l'aide du parseur XIP (Xerox Incremental Parser) vont permettre de désambiguïser un texte étiqueté, construire l'arbre des unités (linguistiques) et d'en extraire les relations des dépendances entre les différentes unités identifiées dans ce texte.

1.3.1 Cahier des charges

Mon stage se situe dans une entreprise qui est en même temps un centre de recherche. Comme il s'agit d'un travail de développement, mais due à l'environnement aussi de recherche, le cahier des charges contenait seulement le résultat, les étapes pour y arriver étaient libres de choix.

L'objectif principal était de réaliser une grammaire XIP pour l'allemand et de l'adapter à Pocket Engineer. Pour ce faire il est prévu que j'adapte les règles existantes et en crée des nouvelles si les existantes ne suffisent pas. En parallèle le lexique doit être également adapté à PE, il ne contient pas encore tous les termes techniques propres à cette application. Finalement la grammaire doit être testée et évaluée, aussi bien sur des textes généraux que sur des textes spécifiques à PE.

Selon l'avancement du projet, des règles de synonymes et de paraphrases seront développées pour Pocket Engineer. Ces règles permettront un fonctionnement plus efficace de l'application, la recherche d'information dans l'aide en ligne.

L'outil de développement a été fixé, il s'agit de XIP, l'analyseur syntaxique de Xerox, pour lequel finalement cette grammaire est développée.

(1)	grammaire générale de l'allemand	
1.1	définition des dépendances	fin mai
1.2	contruction d'un corpus	mi-juin
1.3	développement des règles	fin-juin
1.4	évaluation sur corpus	mi-juillet
(2)	adaptation de la grammaire pour Pocket Engineer	
2.1	analyse du corpus PE	fin juillet
2.2	extension du lexique	fin août
2.3	adaptation des règles	fin août
2.4	évaluation	mi-septembre
2.5	définition de synonymes	fin septembre
(3)	documentation	
3.1	description des dépendances	fin mai
3.2	rapport de stage (pour l'université)	fin juin
3.3	documentation des règles	fin septembre
3.4	rapport d'évaluation	fin septembre

Tab. 1 – Organisation temporelle du travail

Le travail se fait sous linux. Avec CVS[2] (Concurrent Versions System) chaque partie terminée est stockée sur un serveur, et les autres développeurs du groupe peuvent télécharger la version actuelle de l'avancement.

Le tableau 1 montre l'organisation des tâches à effectuer jusqu'à quelle date.

[2]CVS est un logiciel libre de gestion de versions, qui permet de dépister et coordonner les changements réalisés par une équipe de dévéloppement. Un serveur gère une base de donnée centrale contenant toutes les sources, et garde en mémoire les changements effectués.

2 XIP

L'analyse par accroissement de Xerox (**XIP** - Xerox Incremental Parser) est un formalisme qui intègre sans difficultés un certain nombre de mécanismes de description pour l'analyse robuste peu profonde et profonde. Elle s'étend de la désambiguïsation des partie du discours (nom, verbe, adjectif, ...), l'identification d'entités et chunking aux grammaires de dépendance et au traitement supplémentaire.

Des grammaires de XIP ont été développées pour un certain nombre de langues, y compris le français et l'anglais. Les applications principales incluent l'identification contextuelle d'entités, la désambiguïsation lexicologique et structurale, la résolution de coréférence et plus globalement l'extraction d'information.

2.1 Présentation

XIP est un analyseur par accroissement, c'est-à-dire qu'il ne contient pas de récursion. Chaque règle appartient à une couche. Lors du traitement, ces couches sont appliquées l'une après l'autre. Si pour une raison ou une autre une règle ne s'applique pas dans le texte donné en entrée, XIP passe à la règle suivante et continue le traitement d'analyse.

Ceci implique qu'il n'y aura pas d'échec global, XIP livre toujours une sortie. Celle-ci peut-être incomplète si un problème est apparu en cours de traitement, mais le traitement ne sera pas abandonné à cause de ce problème. C'est pour cette raison que l'on parle d'analyseur robuste.

Pour chaque étape l'entrée est la sortie de l'étape précédente. L'ordre des étapes est fixé dans un fichier de contrôle.

2.1.1 Un exemple

Pour avoir tout de suite une idée du traitement de XIP, voici un exemple, qui va être repris (en le simplifiant un peu) dans les parties suivantes. Il s'agit d'une phrase « normale » avec un sujet et un objet.

Prenons la phrase suivante : « Marie hat einen neuen Drucker im Supermarkt gekauft. » (*Marie a acheté une nouvelle imprimante au supermarché.*). XIP fait le traitement et nous donne les informations suivantes (la sortie au complet se trouve dans l'annexe B) :

1. La désambiguïsation détecte que *einen* est un article et non un verbe (ni un adjectif).

2. Le chunking regroupe *Marie*, *ein neuer Drucker*[3] et *Supermarkt* en phrases nominales, *im Supermarkt* en phrase prépositionnelle et *neu* en phrase adjectivale.
3. Les dépendances extraites sont les suivantes : le sujet de la phrase est *Marie*, l'objet du verbe est *ein neuer Drucker* et *neu* est le modifieur du nom *Drucker*.

Les détails de ces étapes seront expliquées dans les paragraphes suivants, après la présentation de l'architecture de XIP.

2.2 Architecture

L'architecture de XIP comprend trois parties principales : la désambiguïsation, le chunking et les dépendances.

L'entrée peut être du texte brut ou formaté (p.e. en XML), ou encore du texte pré-traité. Le traitement de XIP proprement dit vient ensuite, c'est-à-dire la transformation et l'analyse. Le texte est d'abord divisé en phrases et en mots. Puis, ce texte est étiqueté, c'est-à-dire les mots sont classés en catégories (POS - *part of speech*, partie du discours) qui reçoivent des traits (un trait étant une paire attribut-valeur, p.e. nombre-singulier).

L'unité de base est une unité lexicale qui a une forme et qui appartient à une catégorie. Elle peut avoir un certain nombre de traits (dont celui de sa catégorie) pour spécifier la catégorie. Le tout est déclaré comme un nœud et va former une partie de l'arbre de chunk.

Le traitement proprement dit vient après la phase d'étiquetage. La première étape du traitement consiste à désambiguïser le texte, c'est-à-dire, les parties du discours sont détectées et fixées (et éventuellement corrigées le cas échéant). La phase suivante construit l'arbre de chunk, les mots qui appartiennent au même groupe syntaxique sont regroupés ensemble. Et finalement la troisième partie extrait les dépendances existantes dans le texte et termine l'analyse.

La sortie contient un arbre de chunk avec des nœuds terminaux (les mots) et des nœuds non-terminaux (les étiquettes des catégories et de chunks) et une liste des dépendances extraites.

Le schéma de la figure 2 montre l'architecture de XIP. Par la suite, trois parties suivants la partie sur le prétraitement expliquent les modules de désambiguïsation, de chunking et de dépendance.

[3] *ein neuer Drucker* est la forme lemmatisée (normalisée) de *einen neuen Drucker*. En général, XIP fait la normalisation en sortie.

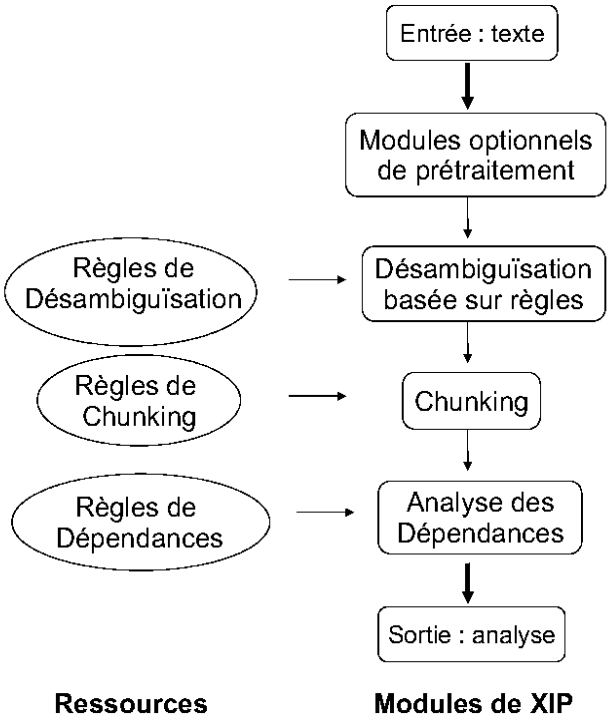


FIG. 2 – Ce schéma montre l'architecture de XIP avec les modules de désambiguïsation, de chunking et de dépendance. L'entrée peut être un document de texte brut ou pré-traité, la sortie livre l'analyse du traitement (un arbre de chunk et une liste de dépendances).

Entre-temps les règles s'appliquent dans l'ordre qui a été défini auparavent. Optionnellement des modules de prétraitement peuvent être insérés avant la désambiguïsation.

2.3 Prétraitement

La toute première analyse est le découpage du texte en phrases et des phrases en mots. Il s'agit d'une étape beaucoup moins triviale que ce que l'on pourrait penser, due à la ponctuation (une abbréviation par exemple ne doit pas provoquer une fin de phrase). Quand les mots sont identifiés, ils sont passés au prétraitement. Celui-ci inclut une analyse morphologique et l'étiquetage.

L'analyse morphologique prend chaque mot et vérifie dans le dictionnaire (qui contient toutes les formes possibles de mots allemands) quelles lectures pourraient être possibles pour ce mot. Par exemple, un verbe peut avoir une lecture pour chaque forme conjuguée qui correspond à la forme du mot

(l'unité lexicale).

Voici un exemple de l'analyse morphologique du mot « einen »(*un*) :

```
einen ein +0+5+PAdj+Indef+Neut+Sg+Gen+Wk+INDADJ
einen ein +0+5+PAdj+Indef+Fem+Sg+Dat+Wk+INDADJ
einen ein +0+5+Det+Indef+Masc+Sg+Acc+AgrWk+ART
einen ein +0+5+Pron+Indef+Masc+Sg+Acc+INDPRO
einen einen +0+5+Verb+IndcSubj+1P+Pl+Pres+VVFIN
einen einen +0+5+Verb+IndcSubj+3P+Pl+Pres+VVFIN
einen einen +0+5+Verb+Inf+VVINF
```

Il ne s'agit ici que des huit premières analyses, en tout il y a pour ce mot 24 analyses possibles ! Dans cet extrait sont présents les lectures *adjectif* (`+PAdj`), *déterminant* (`+Det`), *pronom* (`+Pron`) et *verbe* (`+Verb`), le verbe pouvant être conjugué (`+IndcSubjc`) ou à l'infinitif (`+Inf`).

L'analyse morphologique donne en résultat la catégorie syntaxique et les traits associés au mot, tel que dans le dictionnaire. Pour chaque trait une étiquette est attribuée au mot analysé.

2.4 Désambiguïsation

L'entrée de ce module est la sortie de l'analyse morpgologique. Les phrases du texte sont séparées les unes des autres (la fin d'une phrase reçoit le trait `sent`), ainsi que les mots. Chaque mot porte les traits des catégories attribuées et les traits qui vont avec ces catégories.

Pour l'allemand, la phase d'étiquetage est faite par un formalisme HMM (*Hidden Markov Model*) qui est un algorithme statistique. Il choisi la lecture la plus probable dans le contexte (les mots avant et après le mot en question) et lui attribut une étiquette pour la catégorie.

Mais ce formalisme traitant principalement les parties de discours il ne désambiguïse pas les cas, chose fondamentale pour l'allemand (voir 3.3.1).

Dans la plupart des cas il s'agit dans le module de désambiguïsation donc d'une correction ou d'un raffinement de l'étape d'étiquetage. Ce travail se fait sur la base de contextes. Ces contextes peuvent être plus larges que le contexte pour l'HMM qui ne considère que le mot juste à côté. Les contextes pour la désambiguïsation peuvent inclure plusieurs mots ou groupes de mots de chaque côté.

Selon les mots (ou groupe de mots) à gauche et à droite du mot en question, une seule lecture possible est choisie. Par exemple, dans la phrase « Junge und alte Kinder freuen sich auf die Ferien. » (*Les enfants jeunes et vieux sont contents de l'arrivée des vacances.*) le mot « Junge » peut être un nom ou un adjectif.

Le contexte (une coordination et un adjectif suivi d'un nom) rend la lecture comme adjectif plus probable et ce sera celle-ci qui sera adoptée. Dans ce cas il s'agit d'un choix que j'ai fait, vu que l'on se retrouve avec une phrase ambigue de toute façon. La lecture comme nom serait également correcte, mais moins probable.

Remarque : Il existe bien sûr d'autres règles où la sélection n'est pas arbitraire du tout ! J'ai choisi ici cet exemple pour illustrer la difficulté (et l'arbitrarité du développeur) à faire des choix.

Un exemple non-arbitraire est la désambiguïsation entre un préfixe verbal et une préposition. Dans la phrase « Er kommt an. » (*Il arrive.*), le mot « an » en fin de phrase ne peut être que le préfixe d'un verbe, en aucun cas il peut s'agir d'une préposition.

La désambiguïsation permet aussi de raffiner les traits, par exemple des traits supplémentaires (plus spécifiques) peuvent être ajoutés ou au contraire être supprimés. Par exemple, les noms de personnes peuvent recevoir le trait `individu`.

Chaque phrase reçoit automatiquement les traits `start` et `end` sur le premier et dernier élément. Le module suivant (chunking) ajoutera les traits `first` et `last` sur le premier et dernier élément de chaque chunk.

Les figures 3 et 4 (pour la phrase « Marie kauft einen Drucker. » (*Marie achète une imprimante.*)) montrent qu'avant la désambiguïsation le mot « einen » a entre autres aussi les traits d'un adjectif. C'est seulement après la désambiguïsation que le mot est correctement étiqueté comme déterminant, tous les traits de l'adjectif sont supprimés.

FIG. 3 – Cette figure montre les mots de la phrase donné en entrée, chaque nœud (mot) est représenté avec sa catégorie syntaxique. Avant la désambiguïsation le mot « einen » peut être adjectif, verbe ou déterminant. Les traits de chaque catégorie sont disponibles pour le traitement.

FIG. 4 – Dans cette figure il y a les nœuds de la même phrase après l'étape de désambiguïsation. La catégorie du mot « einen » est fixée. Il s'agit ici d'un déterminant, les traits des autres catégories sont effacés.

2.5 Chunking

Définition d'un chunk

Le chunking consiste à simplement découper les phrases à analyser en « morceaux » non récursifs. Ces « constituants minimaux non récursifs » correspondent grossièrement à la notion linguistique de constituant[4]. Par exemple les chunks verbaux n'incluent jamais leurs arguments mais se contentent d'identifier la complexité de la structure verbale (chunk verbal infinitif, interrogatif, etc.). Les chunks sont décrits par un ensemble de règles exprimées au moyen d'expressions régulières.

L'étape de chunking prend les mots désambiguïsés et les regroupe en chunks. Ces analyses ne produisent pas un arbre syntaxique complet, mais identifient par exemple certains constituants comme les NP (*noun phrase*, phrase nominale), sans indiquer leur structure interne et leur fonction dans la phrase.

Les règles de chunking sont organisées en couches. Un chunk créé par une couche ne sera ni détruit ni modifié par les règles des couches suivantes (rappel : XIP ne contient pas de récursivité).

Évidemment les chunks de base (comme les chunks nominaux, verbaux, ...) sont créés en premier, et ensuite viennent les chunks plus complexes (comme les subordonnées par exemple).

Un chunk nominal par exemple est toute suite de mots qui contient optionnellement un article, suivi optionnellement d'une phrase adjectivale et un nom (ou plusieurs). Le dernier élément d'un chunk, avec lequel se fera l'accord, porte le trait `last`.

La sortie du chunking est un arbre syntaxique avec une racine (le nœud fictif `TOP`), des branches (les chunks) et finalement les feuilles (les mots de la

[4] c.f. *Chunk Stylebook* S. Abney, 1996.

phrase en entrée).

La figure 5 montre un arbre de chunks pour la même phrase que lors de la désambiguïsation.

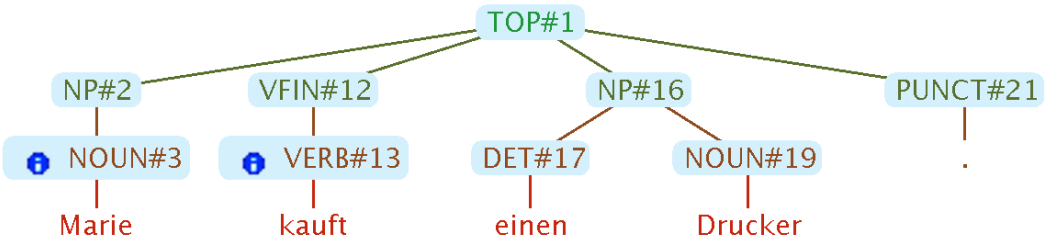


FIG. 5 – Cette figure montre l'arbre de chunk créé à partir de la phrase donnée en entrée. Le chunking regroupe le nom propre, et l'article et le nom dans une phrase nominale (NP) et le verbe en verbe fini (VFIN). Chaque chunk a reçu le trait de sa catégorie plus éventuellement des traits spécifiques.

2.6 Dépendances

Les dépendances entre les mots/nœuds sont extraites à l'aide de l'arbre de chunk, créé par l'étape précédente.

Une dépendance est une relation n-aire qui connecte des nœuds qui entretiennent une relation spécifique. Il y a d'un côté les dépendances standard syntaxiques (p.e. le sujet ou l'objet d'une phrase, les modifieurs des noms, verbes et adjectifs, les temps et modes du verbe, etc.) et de l'autre côté aussi des dépendances plus vastes qui incluent des relations entre différentes phrases (p.e. la coréférence).

Les règles de dépendance sont traitées de manière séquentielle. Une règle plus bas dans le fichier peut modifier ou supprimer une dépendance déjà établie si le contexte s'applique. La création d'une dépendance peut aussi avoir une autre dépendance parmi les conditions, alors elle doit venir après celle-ci.

La sortie du module est une liste qui contient toutes les dépendances extraites. Chaque dépendance a la forme : *Nom de la dépendance (paramètre1, paramètre2, ... , paramètre n)* où les paramètres sont en fait les arguments de la dépendance, c'est-à-dire les mots de la phrase. Par exemple, la dépendance sujet sera `Subj(kaufen, Marie)`, `Marie` étant le sujet du verbe `kaufen`.

La figure 6 montre la liste de dépendances de la phrase déjà donnée à la désambiguïsation et au chunking[5].

Dependency	parameter 0	parameter 1
SUBJ	kauft kauft#13	Marie Marie Marie Marie Marie Marie Marie Marie#3
OBJ_ACC	kauft kauft#13	Drucker#19
VMAIN	kauft kauft#13	
DETERM	Drucker#19	einen#17

FIG. 6 – La dernière figure présente la liste des dépendances extraites de la phrase à analyser. Elle montre que pour cette phrase **Marie** est le sujet de la phrase, que **Drucker** est l'objet, le verbe principal est **kaufen** et le nom est accompagné d'un article, ici **einen**.

Les détails techniques des règles de désambiguïsation, de chunking et de dépendance seront développés dans la section 3.

2.7 Langues

2.7.1 Langues disponibles

Pour les langues anglais et français des grammaires pour XIP ont été développées ces dernières années. Il existe, notamment pour l'anglais, diverses applications sur la base de XIP, par exemple l'extraction d'entités nommées (des noms de personnes, de lieu, de dates, ...) ou des liens de co-référence (reprise d'un sujet ou objet connu dans le texte par un pronom par exemple).

XIP a été développé tout d'abord pour le français, et ensuite pour l'anglais. Son moteur évolue constamment et est adapté si nécessaire avec les nouvelles versions des langues et des applications.

Aujourd'hui il y a quelques universités qui ont une licence pour XIP pour leurs recherches individuelles.

2.7.2 Langues en cours de développement

Pour les langues allemand, espagnol, italien et portugais des grammaires sont en cours de développement. Deux autres stagiaires sont embauchés par

[5]Cette figure, ainsi que les trois précédentes, est une copie d'écran de l'interface graphique (voir 3.7). Dans cette interface, par défaut les éléments de la liste de dépendances sont des formes de surface, c'est-à-dire telles quelles apparaissent dans la phrase. Une sortie du terminal (cf. annexe B) est paramétrée de manière à ce quelle donne au contraire les formes lemmatisées (normalisées). Ces paramètres peuvent être changés par l'utilisateur.

Xerox en ce moment qui sont chargés du développement de grammaires pour l'espagnol et l'italien.

L'objectif du projet est de développer des grammaires générales qui peuvent analyser un vaste choix de sortes de texte et qui peuvent servir ensuite à d'autres applications. L'adaptation à l'application Pocket Engineer se fera par la suite. Les textes de cette application n'ont pas le même format que les textes avec lesquels la grammaire est développée, par exemple des articles de journaux.

Ces textes sont beaucoup plus techniques car la base de données de PE contient surtout des questions d'utilisateurs d'imprimantes et de faxes, et des réponses des responsables de l'aide en ligne pour résoudre ces problèmes. Il s'agit donc d'un type de texte bien particulier qui nécessite une grammaire spéciale.

L'application *Pocket Engineer* sera détaillée dans la deuxième partie du rapport (4.2).

3 La grammaire

3.1 Corpus

Pour développer et tester l'avancement de la grammaire un corpus[6] a été créé. Il contient deux parties, une partie (*corpus d'entraînement*) avec des textes pour développer la grammaire, et une deuxième (*corpus de test*) pour tester si les règles apportent bien le résultat désiré. Le corpus d'entraînement a servi à identifier les problèmes et améliorer les règles quand un problème a été détecté.

La grammaire a été développée au fur et à mesure des difficultés rencontrées. Le corpus de test permettra ensuite de vérifier que la grammaire n'a pas d'effets de bord sur d'autres phénomènes. Il servira aussi pour l'évaluation qui sera réalisée à la fin du stage.

Une partie de mon travail consistait donc à constituer ces corpus. Pour diversifier la structure et le contenu (c'est-à-dire pour bien représenter la langue) ils contiennent d'une part des articles de journaux quotidiens et hebdomadaire comme Spiegel Online, Die Zeit et FAZ, tous tirés de l'internet pendant plusieurs jours et provenants de différents ressorts.

Remarque : La FAZ étant un journal qui a décidé de continuer d'écrire avec l'ancienne orthographe, il est d'un grand avantage que les deux versions de l'orthographe allemande soient gérées dans le dictionnaire. Pour ceci les mots qui existent en deux versions maintenant (ancien et nouveau) portent des traits spécifiques (`old` et `new`).

D'autre part il y a dans ce corpus des extraits de lois et de manuels techniques qui ont une structure spécifique, différente de celle des articles de journaux. Le corpus de PE ressemble plutôt à ces textes-là.

En plus de textes tout faits, j'ai créé des fichiers de texte avec des phrases construites. Ceux-ci m'ont aidé pour le développement de règles très spécifiques pour une situation précise. À la rencontre de problèmes très pointus, j'ai cherché (inventé) un maximum de phrases possibles pour ce cas précis, et c'est ainsi que j'ai constitué des fichiers avec des phrases typiques pour un problème donné, par exemple la coordination de NP, d'AP, etc., l'énumération, les phrases subordonnées, et bien d'autres encore.

Le corpus de PE n'a pas été utilisé pour le développement de la grammaire générale. Il va servir dans la deuxième partie de mon stage à développer la grammaire spécifique à cette application. Ce corpus aussi a été divisé en un corpus d'entraînement et un corpus de test.

[6] Un corpus est une collection de données textuelles qui a été collectée pour servir d'échantillon représentatif de la langue.

Ce qui est très important pour le développement d'une grammaire d'une langue, c'est que ces deux corpus ne doivent jamais être mélangés ou confondus ! En aucun cas le corpus de test doit servir pour développer ou améliorer des règles. Sinon on doit constituer un nouveau corpus de test pour rester correct dans le développement de la grammaire.

3.2 La partie déjà disponible

Le dictionnaire pour faire l'analyse d'étiquetage était disponible à mon arrivée. Il a déjà servi pour d'autres travaux et a juste dû être adapté à quelques endroits pour de petits détails. C'est avec ce dictionnaire que se fait le prétraitement décrit dans la section 2.3.

Le dictionnaire contient pour chaque mot (c'est-à-dire pour chaque forme de mot possible, p.e. le singulier et le pluriel d'un nom, ou toutes les formes conjuguées d'un verbe) son **lemme** (sa forme de base, pour un nom ce sera le singulier, pour un verbe l'infinitif, etc.) et **tous les traits** possibles, dont un trait pour la catégorie. XIP traitera le tout comme un **nœud**.

Ce dictionnaire est un transducteur à états finis (FST, *finite state transducer* en anglais). Chaque lecture possible d'un mot y est codée. Ce dictionnaire est très complet, à part des mots spécifiques pour PE (voir 4.2.1), je n'ai presque rien dû y ajouter.

Je n'ai pas dû commencer mon travail de développement à zéro. Il existait déjà une partie de la désambiguïsation et du chunker, faite par Anne Schiller, ma responsable, avant mon arrivée. Dans un premier temps je me suis donc concentrée sur les dépendances entre les mots et les chunks.

C'est ensuite, en travaillant sur les dépendances, que j'ai modifié, corrigé et raffiné les règles de désambiguïsation et de chunking.

3.3 Le développement

Suite à l'introduction et aux préliminaires, les prochaines sections décrivent mon travail proprement dit pour ParSem.

Une première approche du développement de la grammaire partait du principe d'essayer de faire comme dans les autres langues, c'est-à-dire d'abord la partie désambiguïsation, ensuite le chunking et puis les dépendances. Mais tout de suite la désambiguïsation s'est révélée problématique.

La désambiguïsation des parties du discours est incomplète pour l'allemand par rapport au cas, nombre et genre. Par exemple pour le mot « einen », il ne suffit pas de savoir que c'est un article, mais il faut aussi connaître son cas dans le contexte donné (c.f. la section sur le prétraitement, 2.3).

L'allemand n'étant pas une langue où l'essentiel se joue sur l'ordre des mots, la notion de cas est très importante. Le paragraphe suivant présente le problème pour mieux comprendre la suite.

3.3.1 Les cas

Il existe quatre cas en allemand, le **nominatif** (le sujet de la phrase), et le **génitif**, le **datif** et l'**accusatif** (les objets du verbe). L'ordre des mots dans la phrase n'est pas aussi importante que dans des langues sans cas, en théorie on reconnaît le cas d'un mot, et donc sa fonction dans la phrase, selon sa terminaison.

Malheureusement les cas ne sont pas très marqués en allemand, contrairement au latin p.e. où chaque cas a sa propre terminaison, mais en allemand souvent un mot a la même forme pour plusieurs cas. Ceci implique beaucoup d'ambiguïtés, car même si les mots ont souvent la même forme, l'ordre des mots dans la phrase est quand même plus ou moins libre en allemand.

Par exemple, dans la phrase « Die Frauen haben die Kinder gesehen. » (*Les femmes ont vu les enfants.*) il n'est pas clair qui a vu qui. La phrase peut se traduire aussi par *Les enfants ont vu les femmes.* Certes, l'habitude fait que la première lecture de la phrase est ressentie comme plus probable que la deuxième, mais le problème de la désambiguïsation ne peut pas se résoudre intuitivement.

Cette problématique implique que même après l'étape de désambiguïsation des parties du discours (l'étiquetage faite par l'HMM) beaucoup de choix demeuraient possibles, car les mots pouvaient encore porter les traits de tous les cas.

Les règles « normales » de désambiguïsation, utilisées pour les autres langues, ne sont pas adaptées à l'allemand. Car pour désambiguïser les différentes lectures possibles d'un nom dues aux cas il faudrait prévoir une règle pour chaque combinaison de cas, genre et nombre. Ceci rendrait la grammaire très compliquée.

3.3.2 Double Reduction Rules

Une des solutions au problème de désambiguïsation pour le développement de la grammaire consistait dans le fait d'utiliser les *Double Reduction Rules*, une fonctionnalité de XIP qui permet pour l'allemand de désambiguïser les différents cas, grâce notamment aux articles et adjectifs qui précèdent un nom. Ce sont surtout les articles qui portent le plus de marques explicites pour les différents cas, ils sont donc plus facilement repérables que les noms tout seuls.

Avant de donner la définition de ce type de règle, voici un exemple :

Un exemple La phrase nominale « der neue Drucker » (*la nouvelle imprimante)* peut être désambiguïsée comme suit. Seule la ligne rouge montre le bon acccord entre le déterminant « der », l'adjectif « neue » et le nom « Drucker ». Grâce à la désambiguïsation avec les règles de double réduction toutes les autres lectures peuvent être écartées.

der	**neue**	**Drucker**
+Det+Def+Neut+Pl+Gen	+Adj+Neut+Pl+Nom	+Noun+Masc+Pl+Gen
+Det+Def+Masc+Pl+Gen	+Adj+Neut+Sg+Acc	+Noun+Masc+Pl+Nom
+Det+Def+Fem+Pl+Gen	+Adj+Neut+Sg+Nom	+Noun+Masc+Sg+Acc
+Det+Def+Masc+Sg+Nom	**+Adj+Masc+Sg+Nom**	**+Noun+Masc+Sg+Nom**
+Det+Def+Fem+Sg+Dat	+Adj+Fem+Pl+Acc	+Noun+Masc+Sg+Dat
+Det+Def+Fem+Sg+Gen	+Adj+Fem+Pl+Nom	+Noun+Masc+Pl+Acc
+Pron+Dem+Masc+Sg+Nom	+Adj+Neut+Pl+Acc	
+Pron+Dem+Fem+Sg+Dat	+Adj+Masc+Pl+Acc	
+Pron+Rel+Masc+Sg+Nom	+Adj+Masc+Pl+Nom	
+Pron+Rel+Fem+Sg+Dat	+Adj+Fem+Sg+Acc	
	+Adj+Fem+Sg+Nom	

Définition Les règles de double réduction permettent une simplification parallèle pour deux entrées. Une telle règle comprend deux parties, l'une est appliquée sur les catégories sœurs à la racine de l'arbre de chunk[7], et l'autre applique une expression booléenne sur les nœuds qui forment le patron pour sélectionner les lectures qui sont en conformité avec le test booléen.

Un exemple :

```
DoubleReduction:
|det#1, ?*[verb:~, noun:~], noun#2 | => #1[agr] :: #2[agr].
```

Le mot-clé `DoubleReduction` indique que la suite du traitement concerne des règles de double réduction. La première partie de la règle (jusqu'à la flèche) extrait tous les noms qui sont précédés par un déterminant (et optionnellement aussi par des mots qui ne portent ni le trait `verb`, ni le trait `noun`, comme p.e. un ou des adjectif(s)) tandis que la deuxième partie ne garde que les entrées qui partagent les mêmes valeurs du trait `agr` (accord

[7] Vu que dans notre cas il n'existe à ce stade (la désambiguïsation) pas encore d'arbre de chunk, tous les nœuds sont sœurs, elles dépendent toutes du nœud racine `TOP`.

entre le grenre, le nombre et le cas), ce fait étant testé par l'opérateur deux double points.

Ici, deux double points assurent que les valeurs de l'accord des deux entrées sont **exactement** les mêmes (un seul double point teste seulement si les deux entrées partagent **entre autre** les mêmes valeurs).

Le principe de ces règles est que seules les lectures sont gardées où l'accord entre nom, (adjectif) et déterminant est cohérent. Bien sûr, ceci ne fonctionne que dans les cas où le nom est accompagné soit d'un article ou d'un (ou plusieurs) adjectif(s), ou dans le meilleur cas, des deux.

Si les mots portent les mêmes traits pour le cas, le nombre et le genre, alors ils sont regroupés et toute autre lecture possible est écartée (c.f. le tableau de la page précédente). Sinon l'ambiguïté reste préservée.

Les trois prochaines sections montreront des exemples de règles de grammaire XIP pour l'allemand avec des explications. D'autres règles se trouvent dans l'annexe A. Le ou les mots intéressants auxquels s'applique la règle présentée sont écrits en majuscules dans l'exemple qui précède la règle.

3.4 La désambiguïsation pour l'allemand

La désambiguïsation résoud les ambiguïtés restantes après l'étiquetage. La syntaxe d'une règle de désambiguïsation est la suivante :

```
layer > readings_filter = |left_context| selected_readings
 |right_context|.
```

où `layer` est le numéro de la couche dans laquelle la règle s'applique, `readings_filter` est une expression qui spécifie un sous-ensemble de catégories et les traits associés à ce mot.

Le `|left_context|` et `|right_context|` sont les contextes gauche et droit (un contexte est toujours noté entre deux barres verticales) qui sont une séquence de nœuds optionnelle, il arrive qu'il n'y ait qu'un des deux contextes. Le `selected_readings` finalement est la lecture qui va être adoptée si le patron de la règle correspond à l'entrée donnée au traitement.

Pour l'allemand surtout, un point important de la désambiguïsation est l'attribution du cas juste au nœud nominal. La désambiguïsation corrige et raffine ici la sortie de l'étiquetage.

À part le problème de cas pour les noms, il reste souvent une ambiguïté entre les verbes et les adjectifs d'un côté, et les verbes et les infinitifs de l'autre (les infinitifs sont toujours ambigus avec le verbe conjugué au présent à la troisième personne du pluriel) après l'étape d'étiquetage.

Pour fins d'illustration, les règles suivantes traitent la désambiguïsation entre les verbes finis (un verbe fini exprime le temps, le mode, le nombre et la personne, c'est une forme conjuguée du verbe.) et les infinitifs.

Prenons comme exemple les règles listées plus bas. La première partie de la règle énumère les traits auxquels nous nous intéressons. Il s'agit ici des traits `finite` et `inf` que le mot a reçu par l'étiquetage. La deuxième partie contient le contexte gauche dans lequel ce mot se trouve et ensuite la lecture qui doit être écartée (le tilde « ∼ » exprime la négation).

La double barre oblique introduit un commentaire qui permet de donner un exemple de phrase. Remarque : Xerox étant une entreprise américaine toute la documentation interne se fait en anglais, ce qui vaut aussi pour les commentaires de codes.

```
// finite verb or infinitif
// "Du musst deine Eltern ANRUFEN." vs. "Sie RUFEN uns AN."
// "tu dois APPELLER tes parents."
4> ?[finite, inf, v2] = | noun[acc:+, pl:+]; pron[acc:+, pl:+] |
 ?[finite:~] .
```

Cette règle vérifie en plus des traits `finite` et `inf` le trait `v2` qui est attribué à des verbes comme *an-rufen* où le préfixe *an* peut être séparé de la racine dans la phrase. Sauf dans les subordonnées (« weil sie anrufen », *parce qu'ils appellent*), les participes (« sie haben angerufen », *ils ont appellé*) et les infinitifs (« sie werden anrufen », *ils vont appeller*) le verbe sera toujours séparé du préfixe dans les formes conjuguées. Il ne peut donc pas s'agir ici d'un verbe fini.

```
// "Die Kinder SPIELEN gerne." vs. "Die Kinder wollen SPIELEN."
// "les enfants aiment JOUER."
4> ?[finite, inf] = | ~verb, noun[nom:+, pl:+]; pron[nom:+, pl:+],
 (det[gen:+], noun[gen:+]), (adv) | ?[inf:~] .
```

Ici on prend le contexte d'un nom ou pronom au pluriel qui précède directement le mot (optionnellement il peut y avoir encore un nom et article au génitif (souvent pour le possessif), et/ou un adverbe avant le verbe en question.) et la condition qu'il n'existe pas un autre verbe plus en amont dans la phrase.

On se retrouve en présence de l'ambiguïté entre infinitif et verbe conjugué à la troisième personne au pluriel du présent, mentionnée plus haut. Pour le mot en entrée, il s'agit du verbe fini au pluriel quand il suit un nom au pluriel. Ici c'est le prédicat du nom, sujet de la phrase.

```
// "Oma und Opa KOMMEN." vs. "Oma mag heute nicht KOMMEN."
// "Grand-mère et grand-père VIENNENT."
4> ?[finite, inf] = | noun[nom:+]; pron[nom:+], conj[coord],
 noun[nom:+]; pron[nom:+], (adv) | ?[inf:~] .
```

Cette dernière règle traite la coordination (*UND*). Deux noms (au singulier) au nominatif, reliés par une conjonction de coordination requièrent un verbe fini (au pluriel). Nous ne testons pas toutes les contraintes (comme le nombre) pour ne pas surcharger le traitement. L'expérience a montré que la règle s'applique telle quelle dans les cas qui sont les bons.

Si nécessaire (pour un cas plus complexe) il n'y aura pas de difficultés à adapter les règles. Plus de règles, traitants d'autres problèmes, notamment l'ambiguïté entre adjectif et nom en début de phrase, se trouvent dans l'annexe A.1.

Un autre aspect spécifique de la désambiguïsation de l'allemand est le bon découpage des mots composés[8]. Les langues latines créent des sens complexes en enchaînant les mots par des prépositions en « de ». L'allemand regroupe les sens en un mot avec un sens très complexe s'il contient plus de deux ou trois parties. Un lecteur humain décompose assez facilement et surtout correctement ces mots, sans problèmes majeurs.

Pour le traitement automatique cette tâche est plus compliquée. En résultat on reçoit souvent une multitude de découpages morpho-syntaxiquement possibles, mais sémantiquement impossibles ou abérrants. Il faut donc des règles pour que XIP découpe aux bons endroits.

Une approche est d'attribuer à ces mots un trait `rank` avec une valeur numérique pendant le prétraitement. Les mots simples auront la valeur 1, un mot composé de deux segments aura 2, etc. Au moment de la désambiguïsation un groupe de règles finalement très simples donne la priorité au découpage avec la valeur du trait `rank` au minimum.

Des mots comme « Verbraucher » (*consommateur*), « Gründung » (*fondation*) et même un anglicisme comme « Teenager » (*adolescent*) seront donc analysés comme mots simples, alors qu'une autre lecture possible, mais sémantiquement très douteuse, serait « Verb-raucher » (*fumeur de verbes*), « Grün-dung » (*engrais vert*) et « Tee-nager » (*rongeur de thé*).

Les règles de sélection de segmentation se trouvent également dans l'annexe A.1. (Dans la grammaire ça va jusqu'à 9, ici j'en ai mis que les premières.)

La prochaine étape du traitement est le chunking, expliqué par la suite.

[8] c.f. *German compound analysis with wfsc* A. Schiller, 2005.

3.5 Le chunking pour l'allemand

Le chunking regroupe les nœuds qui forment un groupe et crée un arbre de chunks. La syntaxe d'une règle de chunking est la suivante :

```
layer> new_node = list_of_lexical_nodes.
```

où `layer` est (comme pour les règles de désambiguïsation) le numéro de la couche dans laquelle la règle s'applique, `new_node` est le nouveau nœud créé par la règle et `list_of_lexical_nodes` est la liste des nœuds qui doivent être groupés.

Les chunks les plus importants sont les chunks nominaux (appellés NP (*nominal phrase* en anglais), la tête est un nom), les chunks verbaux (VFIN, la tête est un verbe fini), les chunks adjectivaux (AP, la tête est un adjectif) et les chunks prépositionnels (PP, la tête est le nom du NP relié à la préposition).

La tête d'un chunk est toujours le dernier élément du groupe, c'est avec celui-ci que se fera l'accord, par exemple entre un NP et un VFIN pour la dépendance sujet ou entre un NP et un AP pour la dépendance NMOD (un modifieur du nom). Dans ce cas le dernier adjectif du AP doit avoir les mêmes traits pour le nombre et le cas que le dernier nom du NP suivant.

Dans un VFIN la tête est le verbe conjugué, même si ce n'est que l'auxiliaire. Le verbe qui porte l'information sémantique et qui vient après, n'est en général pas regroupé sous le VFIN, il reste un nœud VERB tout court et sera relié à sa tête morphologique (le verbe conjugué) par une règle de dépendance.

Pour les autres langues, il existe aussi des chunks pour la phrase principale (SC, *sentence clause*) ou des chunks pour marquer le début d'une subordonnée (BG, *begin*), et d'autres encore peuvent être créés si besoin est. Le chunk porte toujours au moins le trait de sa classe, par exemple, un chunk nominal aura le trait `NP`.

À l'heure actuelle la grammaire allemande contient des règles pour la création de chunks nominaux (NP), de chunk verbaux (VFIN), de chunk adjectivaux (AP) et de chunk prépositionnels (PP). J'ai commencé des règles pour la création des BG, mais vu qu'il n'y a pas de SC pour l'allemand, cette tâche s'est avérée assez complexe.

La raison pour laquelle il n'y a pas (encore) de SC pour l'allemand vient encore du fait que l'ordre des mots (c.f. 3.3.1) est (trop) libre pour la modélisation de telles règles. Car un NP qui précède un VFIN, n'est pas forcément le bon pour former un SC, qui contient en général le sujet, donc un NP au nominatif.

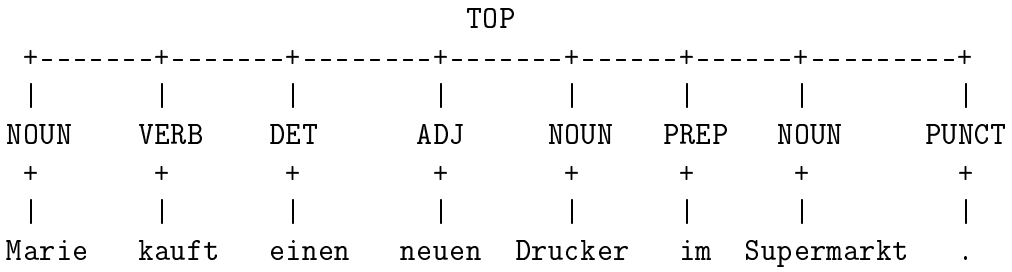
La question qui reste à discuter est si on a vraiment besoin de chunks SC pour l'allemand. Car grâce aux traits pour les cas, on peut reconnaître un

chunk nominal au nominatif immédiatement et le relier avec une dépendance au chunk verbal, il n'est pas besoin d'alourdir l'arbre avec une couche de chunks en plus qui ralentirait le traitement et qui rendrait plus difficile la recherche d'information sur les nœuds (un niveau de plus ou de moins à traverser fait une grande différence).

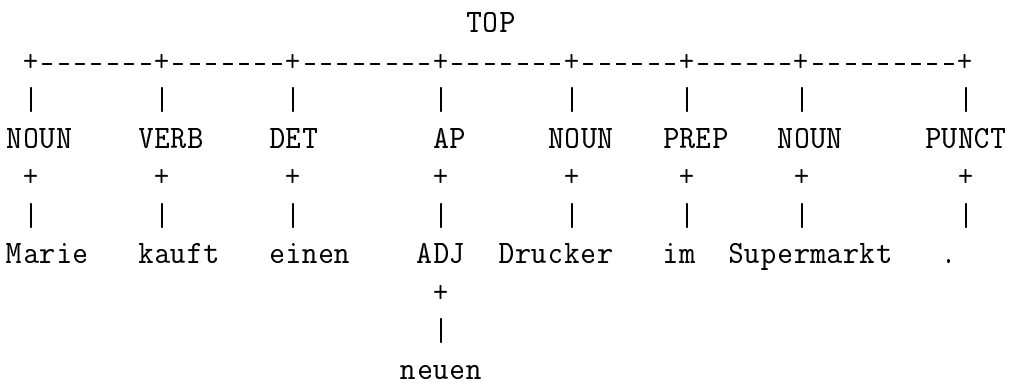
Dans la partie du stage à venir je ne vais donc pas seulement travailler sur Pocket Engineer, mais aussi compléter la grammaire générale, en ajoutant par exemple des chunks plus complexes pour les verbes. On pourrait imaginer des chunks spécifiques pour les infinitifs, les participes, etc. comme il en existe pour l'anglais et le français.

Durant la phase de chunking, on crée d'abord les chunks adjectivaux, ensuite les chunks nominaux, et puis les chunks prépositionnels. Dans un deuxième temps viennent les chunks verbaux, et à la fin les chunks BG. (Les chunks SC (si on décide de les faire) seront probablement introduits entre les VFIN et les BG.)

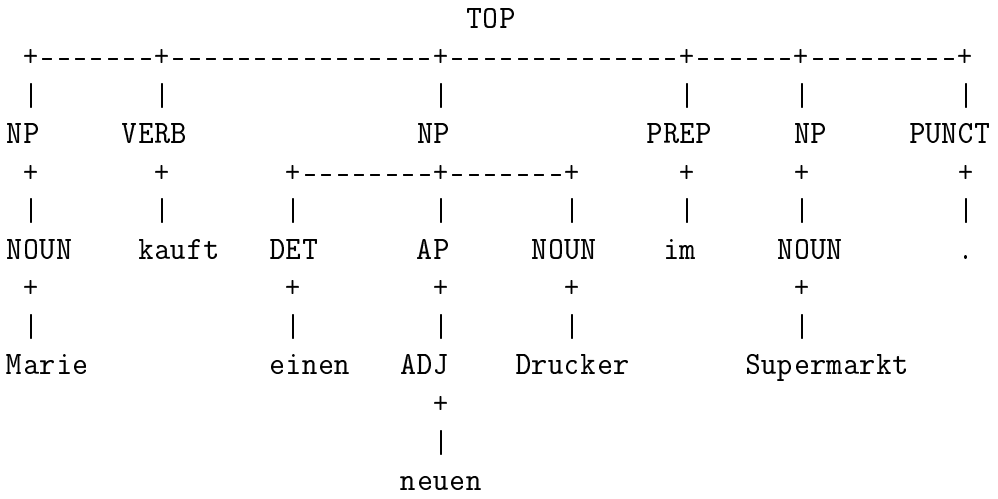
Reprenons la phrase donnée en exemple dans l'introduction sur XIP (c.f. 2.1.1), « Marie kauft einen neuen Drucker im Supermarkt. ». Dans un premier temps la phrase a une structure plate, tous les nœuds sont au même niveau, comme ceci :



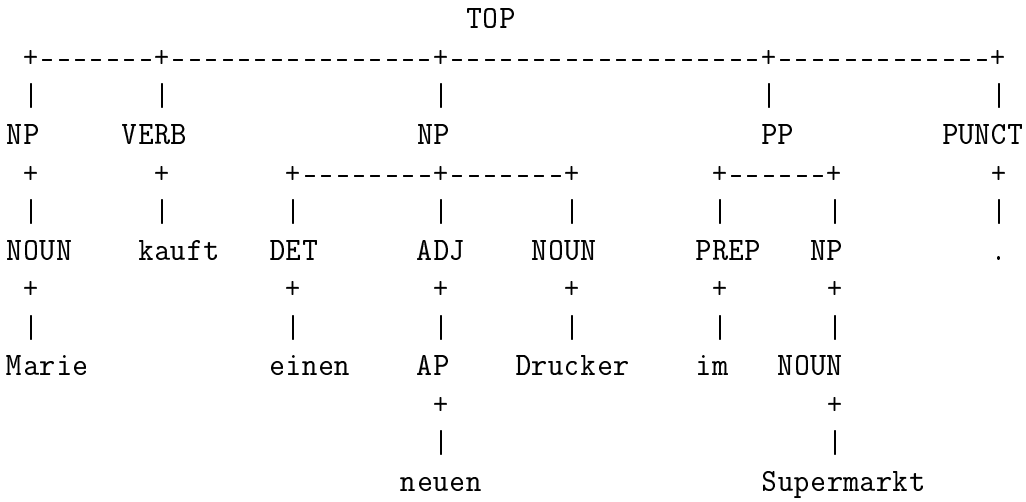
Les premiers chunks qui sont créés sont les AP, ce qui donne l'arbre suivant :



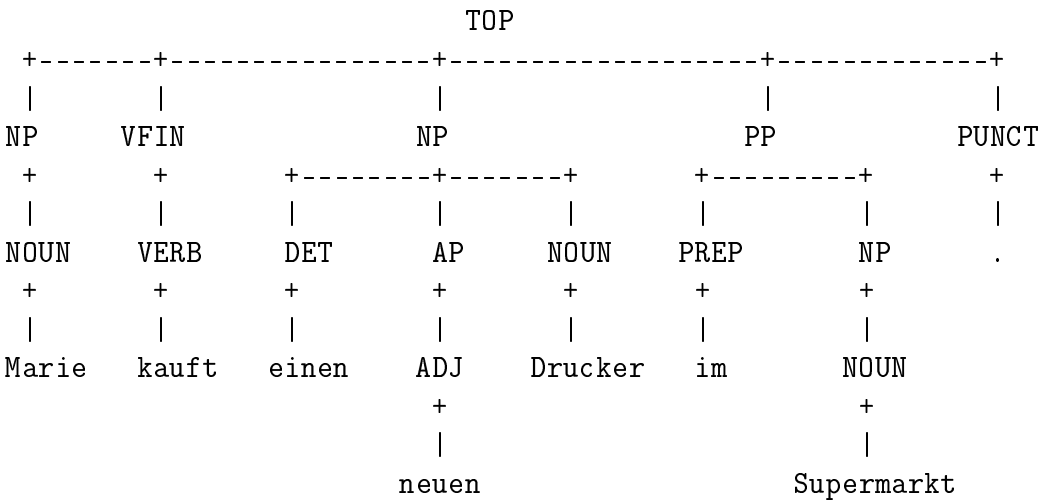
Puis, les NP sont créés et la phrase aura la structure suivante :



Par la suite les PP regroupent un NP suivant une préposition pour donner l'arbre si-dessous :



Finalement, les VFIN sont construits ce qui donne l'arbre final :



Après cet aperçu, voici un exemple concret pour une règle de chunking de la grammaire allemande :

```
// simple NP
// "SEHR GROSSE BÄUME"
// "DES ARBRES TRÈS GRANDS"
115> np[p3=+] =  ( ap[ !agr:! ]), noun[ !agr:! ].
```

Un chunk nominal (NP) est créé d'un nom précédé optionnellement d'une phrase adjectivale (AP). L'accord (`!agr :!`) doit correspondre pour que la règle s'applique. Le nouveau NP reçoit ici le trait `p3` pour la troisième personne. Cette *assignation* de trait se fait par le biais du signe égal (`=`), un test de traits se fait par le double point.

L'exemple suivant montre la création d'un chunk prépositionnel. Le mot-clé `where` force le test du cas, seulement si le NP est du même cas que la préposition, le tout forme un groupe. La préposition doit être du type préposé (`pre`), une préposition postposée ne formera pas de chunk prépositionnel.

Le test dans la partie `where` se fait à l'aide de variables (de la forme `#` et un numéro). La préposition a reçu le nom de variable `#1` et le NP est marquée par `#2`. De cette manière on peut facilement référer à des nœuds précis dans le chunk pour faire des tests (ou établir des dépendances, dont parle la prochaine section).

Voici la règle :

```
// PP
// "IN DEN FERIEN"
// "PENDANT LES VACANCES"
150> pp = prep#1[pre:+], np#2, where ( #1[case]:#2[case] ).
```

D'autres exemples de règles, p.e. pour les noms propres et les chunks verbaux, se trouvent dans l'annexe A.2.

Un point qui pose des problèmes dans toutes les langues, l'allemand inclut, est la coordination. La coordination simple, entre deux noms reliés avec une conjonction de coordination, est assez facile à traiter. On peut en faire un chunk nominal, ou la définir comme dépendance comme c'est le cas à l'heure actuelle.

Mais comment traiter les énumérations, ou les coordinations plus complexes, par exemple quand il y a d'autres éléments entre les mots qui doivent être coordonnés. Là encore, c'est un problème auquel je vais me consacrer durant la deuxième partie du stage.

3.6 Les dépendances pour l'allemand

Une dépendance est une relation entre différents nœuds d'une phrase. Il peut s'agir de dépendances standards, comme *sujet*, *objet*, ... ou de dépendances de base comme le modifieur d'un nom, verbe ou adjectif. Les dépendances *intersentential* (entre deux phrases) seront l'objet de la deuxième partie du stage.

La syntaxe d'une règle de dépendance est la suivante :

```
| pattern | if <condition> <dependency_terms>.
```

où `pattern` est le motif de la structure de la phrase, en général il s'agit (d'une partie) de l'arbre de chunk de l'entrée. Les mots entre lesquels la dépendance sera établie doivent être marqués par des variables (une variable étant un dièse suivi d'un numéro (p.e. `#1`)) pour pouvoir les reprendre ensuite dans la dépendance qui établie la relation.

Des `condition` optionnelles sont testées (p.e. l'accord en genre et nombre), puis la dépendance `dependency_terms` est établie. Une règle de dépendance peut contenir aussi seulement une ou plusieurs conditions, le motif étant facultatif dans ce cas. La dépendance est établie si le test a du succès, ou s'il n'y a pas de conditions à tester.

La dépendance aura toujours la forme :

```
label[features](arg1, arg2, ..., argn)
```

où `label` est le nom de la dépendance, elle peut porter des traits spécifiques (`[features]`). Elle est toujours accompagnée des ses arguments (`(arg1, arg2, ..., argn)`). Le nombre d'arguments peut changer d'une dépendance à l'autre, mais doit toujours être le même pour une même dépendance.

Avant de commencer la création des règles, j'ai réfléchi d'abord sur la conception des dépendances souhaitables et rédigé un document qui décrit et détaille les dépendances que l'on veut extraire avec la grammaire. Il contient en partie les mêmes dépendances que pour le français par exemple (SUBJ, OBJ, NMOD, etc.), et puis un certain nombre d'autres, spécifiques à l'allemand (comme OBJ_GEN, pour l'objet indirect au génitif).

Pour le développement de cette grammaire, j'ai commencé avec les dépendances dites *simples* : les modifieurs du verbe, du nom, de l'adjectif, et les différentes dépendances du verbe : verbe au passé, verbe au passif, verbe auxiliaire, verbe modale, etc. Puis je me suis intéressée à la coordination, j'ai créé des dépendances pour des coordinations entre noms, verbes, adjectifs et pour les noms dans les phrases prépositionnelles.

Pour reprendre la phrase ambigue du paragraphe 3.3.1, (« Die Frauen haben die Kinder gesehen. ») voici la règle qui extrait le sujet dans la position classique, c'est-à-dire avant le verbe, et l'objet après le verbe.

Le NP doit être au nominatif et avoir le même trait pour le nombre que le verbe. Le nom ne doit pas, par ailleurs, être relié par la dépendance MOD_GEN avec un autre nom.

```
// subject before the verb
// "Die FRAUEN haben die Kinder gesehen."
| np#3[nom:+, rel:~]{?*, #1[last]}, vfin{?*, #2[last]} |
        if ( #2[number]:#3[number] & ~MOD_GEN(#10, #1) )
        SUBJ(#2, #1).
```

La règle suivante par contre détecte un nominatif qui suit un verbe comme le sujet de la phrase, avec toujours la même condition que le nom et le verbe doivent porter le même trait pour le nombre :

```
// subject after the verb
// "Die Frauen haben die KINDER gesehen."
| vfin[start:~]{?*, #2[last]}, np#3[nom:+]{?*, #1[last]} |
        if ( #2[number]:#3[number] )
        SUBJ(#2, #1).
```

Ici, le verbe ne doit pas avoir le trait `start`, sinon il s'agirait d'une phrase interrogative. Les règles pour trouver l'objet (direct, un accusatif) dans les deux cas se trouvent dans l'annexe A.3.

Ces exemples montrent bien que l'ordre des mots n'est pas primordial pour trouver le sujet. Un autre exemple serait une inversion de la phrase de l'introduction de XIP (« Marie hat einen neuen Drucker im Supermarkt gekauft. », c.f 2.1.1). Si elle apparaît sous la forme « Im Supermarkt hat

Marie einen neuen Drucker gekauft. », toujours correcte, le sujet extrait doit quand même être *Marie*, et surtout pas *Supermarkt* du PP !

Les paragraphes suivants présentent un autre exemple de règles pour la dépendance **VMOD**, un mot qui modifie le verbe. Il peut s'agir d'un adjectif ou d'un adverbe, ou encore du nom d'un PP. Ces règles ne s'appliquent qu'aux verbes pleins. Les verbes modales ou auxilaires ne provoquent pas cette dépendance.

La première règle ci-dessous crée la relation si un verbe est suivi d'un adjectif, optionnellement précédé par un adverbe. La deuxième est prévue pour la même situation, mais niée. La dépendance aura un argument (la particule de négation) en plus.

```
// modifier of the verb
// "er läuft SEHR SCHNELL"
// "il court TRÈS VITE"
| vfin{verb#1[modal:~, aux:~]}, (adv), adj#2 |
        VMOD(#1, #2).

// "es regnet NICHT STARK"
// "il NE pleut PAS FORT"
| vfin{verb#1[modal:~, aux:~]}, ptcl#2[!neg:!], (adv), adj#3 |
        VMOD(#1, #2, #3).
```

Les points d'exclamation qui entourent le trait `neg` provoquent que la dépendance `VMOD` aura elle aussi le trait `neg`, qui sera rajouté à la sortie à la fin de son nom : `VMOD_NEG(#1, #2, #3)`. Le même effet aurait eu la règle suivante :

```
| vfin{verb#1[modal:~, aux:~]}, (adv), ptcl#2[neg:+], adj#3 |
        VMOD[neg=+](#1, #2, #3).
```

Le phénomène que le trait remonte « tout seul » s'appelle *percolation* et est très commode.

Cette fonction peut aussi être utilisée au niveau du chunking. L'avantage de travailler plutôt avec la percolation qu'avec l'assignation de trait explicite est que le trait en question remonte à tous les niveaux, jusqu'au niveau le plus haut, juste en-dessous du nœud racine `TOP`. Nous nous servons également de la percolation pour fixer les cas, au moment de la création des NP (c.f. 3.5).

La prochaine règle établie la dépendance VMOD quand le verbe est suivi d'une phrase prépositionnelle. Le modifieur du verbe est ici le nom du NP relié dans le PP. La préposition de cette phrase prépositionnelle ne doit pas porter le trait `novmod` qui est associé à certaines prépositions qui ne doivent

pas provoquer cette dépendance (comme p.e. « wegen » (*à cause de*) ou « während» (*pendant que*)).

Comme beaucoup d'autres règles, celle-ci contient également des éléments optionnels, par exemple un adverbe ou un autre PP, ou encore un NP (qui sera dans ce cas (très probablement) le sujet de la phrase).

```
// "sie fahren nach BERLIN"
// "ils vont à BERLIN"
| vfin{verb#1[modal:~, aux:~]}, (np[nom]; pp; adv),
 pp{prep[novmod:~], ?*, np{?*, #2[last]}} |
        VMOD(#1, #2).
```

Le deuxième argument de cette dépendance, celui qui modifie le verbe, sera le nom (la tête) du chunk PP.

Après les dépendances simples j'ai écrit des règles pour les dépendances plus complexes. À l'heure actuelle il y a des règles pour la dépendance sujet, pour les différents objets (génitif, datif, accusatif et de phrase, c'est-à-dire une conjonction qui introduit une subordonnée). Pour le sujet il y a des règles pour le cas où le sujet est sujet de plusieurs verbes, ou s'il y a plusieurs sujets pour un verbe (au pluriel).

Surtout pour les dépendances du sujet et des objets le fait qu'un NP puisse avoir encore plusieurs traits de cas différents était très problématique. Au début de mon travail il y avait souvent autant de « sujets » à la sortie que de NP détectés, et autant d'objets directs et indirects. Suite à ce constat j'ai commencé à me servir de la possibilité qu'on pouvait aussi effacer des dépendances erronées ou en trop.

Il existe bien sûr des règles pour la coordination qui devront être raffinées dans la deuxième partie du stage. Ce constat vaut aussi pour les dépendances pour le sujet d'une subordonnée, pour l'instant il n'est extrait que si la phrase correspond bien au motif donné. Et finalement les objets d'une subordonnée devront être extraits également.

Et pour conclure, toutes les dépendances qui seront nécessaires à Pocket Engineer feront l'objet des travaux des mois à venir.

D'autres exemples de dépendances, notamment les plus complexes comme sujet et objet, se trouvent dans l'annexe A.3.

3.7 Utilisation de l'interface

J'ai effectué presque tout le développement de la grammaire avec une interface graphique qui a été implémentée par un stagiaire du XRCE de l'année dernière[9].

[9] c.f. *Graphical User Interface for XIP* V. Grassaud, 2005.

Cette interface propose deux modes de mise en œuvre, la première (**Parse**) donne exactement le même résultat que la sortie du terminal. Ce mode sert pour une vérification générale des règles ou pour l'application finale où les pas intermédiaires ne sont pas pertinents.

La deuxième, appellée **Debug**, affiche un arbre graphique dans lequel on peut faire afficher les traits de chaque nœud en cliquant dessus. Elle permet aussi de suivre au pas à pas comment le traitement se déroule et de voir quelle règle s'applique à quel moment.

Ceci a été particulièrement utile pour le développement, surtout pour les règles plus complexes. Quand le résultat n'est pas celui qu'on attend on peut tout de suite voir à quel moment les règles posent des problèmes et corriger l'erreur.

Et en utilisant quotidiennement cette interface, nos expériences de travail ont participé d'un autre côté à améliorer celle-ci considérablement (pendant les trois derniers mois il y a eu plusieurs nouvelles versions). Les copies d'écran présentes dans ce rapport sont des extraits de cette interface.

Le même travail de développement est possible en ne se servant que de la ligne de commande, mais il ne propose que le résultat final en sortie. (Il est possible de définir des étapes dans les règles et de les appliquer seulement jusqu'à une étape précise. Mais ces étapes doivent être sélectionnées et définies clairement avant le traitement, alors qu'on ne sait pas encore exactement où on veut voir des détails.) L'interface a vraiment rendu le développement plus agréable, plus clair et plus rapide.

Une autre fonctionnalité de l'interface est le fait qu'elle propose de s'organiser en « projets ». J'ai donc créé deux projets, un premier pour le développement de la grammaire générale, avec tous les fichiers nécessaires, et un deuxième pour l'application PE avec les fichiers spécifiques à cette application en plus de ceux pour la grammaire générale.

La figure 7 montre l'interface graphique avec laquelle j'ai développé la grammaire de l'allemand.

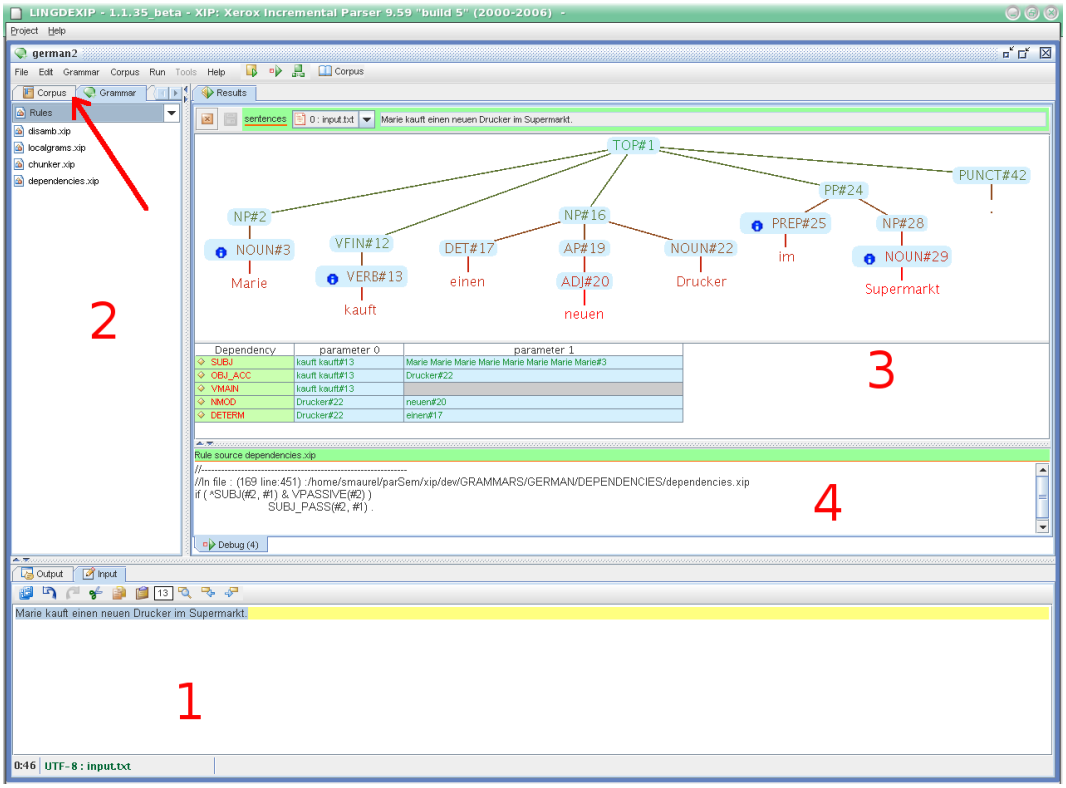


FIG. 7 – Voici une copie d'écran de l'interface graphique avec laquelle s'est faite le développement. Dans la partie en bas (1) on peut saisir des phrases pour les tester séparément. Si tout un texte doit être analysé il peut être sélectionné dans la partie gauche (2) sous l'onglet *Corpus*, caché dans cette figure par l'onglet *Grammaire* (derrière la flèche). Cet onglet montre tous les fichiers de grammaire qui participent à l'analyse, dans leur ordre de traitement. Un troisième onglet *Debug*, également caché, permet de faire du pas à pas. Un bouton *next* donne la possibilité de faire s'appliquer les règles une à une et de suivre ainsi le traitement.

La partie centrale de l'interface (3) est occupée par l'arbre de chunk créé et la liste de dépendances extraites. En dessous (4) se trouve la dernière règle appliquée, lors du pas à pas on peut voir ici tout de suite quelle règle s'applique à quel moment et dans quelles conditions.

La phrase donnée en entrée est toujours « Marie kauft einen neuen Drucker im Supermarkt. ».

4 Évaluation et PE

Cette deuxième partie de mon rapport présente les tâches que je vais réaliser les trois prochains mois. Il s'agit de l'évaluation de la grammaire générale de l'allemand et du développement de la grammaire spécifique pour l'application Pocket Engineer. Pour cette deuxième grammaire je vais faire également une évaluation quand elle sera réalisée. Une évaluation externe par des utilisateurs sera effectuée ensuite dans le cadre du projet.

Dans un premier temps, le terme d'« évaluation » est introduit. Par la suite je présenterai le projet de l'application *Pocket Engineer* et les premiers pas effectués jusqu'à maintenant pour ce projet.

4.1 L'évaluation

En matière de génie logiciel, évaluer les performances d'un outil est une tâche qui apparaît comme incontournable. Quelle que soit la discipline ou l'application, l'utilisateur comme le développeur ont en effet besoin de connaître les capacités effectives d'un logiciel avant de l'acquérir ou de le produire. Une grammaire XIP ne fait pas exception à ce besoin.

L'objectif d'une évaluation est de mesurer le niveau de performance du système et d'apprécier la justesse des résultats produits. Largement utilisées en TALN (Traitement Automatique des Langues Naturelles), les mesures de *Précision* et de *Rappel* permettent une appréciation globale et rapide des compétences d'un outil. Connues et utilisées par tous, ces mesures apparaissent comme les indices incontournables de toute évaluation, autorisant la comparaison avec d'autres systèmes et garantissant la compréhension de la part d'autres concepteurs-évaluateurs.

Il n'est pas très aisé de se faire une idée intuitive de ce que représentent ces mesures. De manière formelle, il s'agit des rapports suivants :

$$Précision = \frac{Nombre\ de\ réponses\ pertinentes\ extraites\ par\ le\ système}{Nombre\ total\ de\ réponses\ sélectionnées\ par\ le\ système}$$

$$Rappel = \frac{Nombre\ de\ réponses\ pertinentes\ extraites\ par\ le\ système}{Nombre\ total\ de\ réponses\ pertinentes\ de\ référence}$$

Le Rappel correspond au taux de couverture du système, à sa capacité d'analyser correctement tous les phénomènes présents dans le texte donné en entrée. La Précision permet quant à elle de se faire une idée de la pertinence du système, de sa capacité à déclarer correctement l'analyse effectuée : cette mesure est inversement proportionnelle à la précédente.

Pour évaluer la grammaire générale de l'allemand que je suis en train de développer, il faudra définir dans un premier temps les critères d'évaluation. Par la suite, le corpus de test, spécialement créé à cet effet, sera analysé par la grammaire et le résultat jugé, peut être en comparaison avec les grammaires des autres langues.

4.1.1 L'évaluation d'une grammaire

L'évaluation d'une grammaire entière pour une langue est un projet très vaste. Est-ce qu'on s'intéresse essentiellement à la création correcte de chunks, ou plutôt à l'extraction correcte des dépendances, ou encore, plus basique, à la désambiguïsation correcte des nœuds ?

Pour tester une grammaire il faut avoir à sa disposition un corpus dit *annoté*, c'est-à-dire un texte au format XML par exemple qui contient toutes les informations nécessaires à l'évaluation. Comme nous ne disposons pas d'un tel corpus pour l'allemand, l'annotation devra être faite à la main. Il s'agit ici d'un travail long et fastidieux qui pourra être effectué seulement sur un échantillon limité de textes, dû au manque de temps. Cet échantillon ne pourra donc pas servir pour un test de référence. Une évaluation thématique nous paraît donc plus intéressante.

Pour nos fins il sera plus fructueux de faire une évaluation d'un échantillon réduit de textes pour avoir un aperçu global de la grammaire, et par la suite une évaluation thématique sur un point précis, par exemple l'extraction correcte de tous les sujets d'un texte. Vu qu'il ne s'agit là que d'une seule dépendance, le fait qu'elle a correctement été extraite pour toutes les phrases pourra être facilement vérifiée à la main.

Cette évaluation de la grammaire allemande sera comparée aux évaluations des grammaires des autres langues. Pour l'instant les trois grammaires (allemande, espagnole et italienne) sont à peu près avancées au même niveau.

4.2 L'application Pocket Engineer

Il s'agit pour PE d'un projet qui cherche à améliorer la recherche d'information dans l'aide en ligne des imprimantes Xerox. L'idée est d'analyser le corpus d'aide en ligne avant de l'indexer pour en extraire des relations syntaxiques et sémantiques pertinentes qui puissent ensuite contribuer à obtenir des réponses plus adéquates aux questions de l'utilisateur.

Cette application est déjà assez avancée pour l'anglais. L'utilisateur peut taper une question dans un champ d'une interface graphique, puis cette question est analysée par XIP. Par la suite la base de données est balayée pour

trouver des réponses qui contiennent les mots pertinents de la requête, mais aussi ses synonymes, qui pourraient donner également une bonne réponse.

L'utilisateur se voit ensuite afficher une sorte d'arbre dans lequel il peut spécifier son problème. Prenons un exemple : quand il imprime un document à partir de son ordinateur, et que l'impression sort froissée de l'imprimante. Il va donc taper les mots « impression », « froissée » et « poste de travail » pour trouver une solution.

L'arbre de raffinement va lui proposer de préciser de quelle sorte de poste de travail il s'agit (PC, PDA, ...), ensuite il peut sélectionner le type de papier, de l'imprimante, etc. De cette manière il peut détailler de plus en plus finement son problème et trouver ainsi rapidement la solution, grâce au codage des mots-clés en synonymes et paraphrases.

L'avantage indéniable de ce procédé est que l'utilisateur n'a pas à visionner une grande quantité de réponses, dont de toute façon la plupart ne couvrent pas exactement son problème. Alors que dans un tel cas, un problème technique avec une machine, la réponse exacte est vraiment primordiale, une réponse vague ne servira à rien l'utilisateur.

4.2.1 Premiers pas

J'ai déjà commencé le développement de la grammaire spécifique pour l'application Pocket Engineer, mais à l'heure actuelle son état est au stade initial.

Les premiers pas étaient de rajouter des mots comme *Web*, *Touchscreen*, *CopyBox*, ... au dictionnaire, c'est-à-dire dans le FST. Pour ceci le mot doit être déclaré avec sa classe de déclinaison exacte, ensuite toutes les formes possibles sont générées et le mot est ajouté au dictionnaire.

Autre chose faite est la création d'un fichier de grammaire dite « locale » qui contient des règles pour reconnaître des suites de mots comme *Windows 2000*, *Windows XP*, *Xerox Document Centre*, etc. et les déclarer en un nom seulement (n'ayant qu'un seul nœud dans l'arbre de chunk). Il s'agit pour la plupart de nom propre de systèmes d'exploitation, nom d'imprimantes, etc. très fréquents dans le corpus de PE.

Les numéros de version (d'imprimantes ou de logiciels utilisés pour l'impression) sont traités d'une façon similaire, c'est-à-dire, le produit et son numéro de version ne forme qu'un seul chunk. Là encore la coordination de produits en coordonnant seulement le numéro de la version (p.e. *Windows 2000 et XP*, *Xerox Document Centre 420/425/432/440* ou *Xerox DC220/230/332/340*) sera très vraisemblablement problématique pour le traitement. Il reste donc à discuter si on fait un seul chunk dans ce cas, peut-être avec un trait spécial pour marquer l'énumération.

Conclusion et Perspectives

En conclusion je voudrais attirer l'attention sur le fait qu'une nouvelle grammaire pour XIP a été créée. Elle contient des règles pour tous les modules nécessaires au traitement d'un texte allemand pour une analyse syntaxique robuste.

Ceci est fait par l'analyseur XIP qui fait un traitement incrémental du texte. Il contient trois modules principaux et des modules de prétraitement optionels.

Le module de désambiguïsation attribut les catégories syntaxiques aux mots du texte. L'ambiguïté entre les cas (un problème spécifique à l'allemand), souvent présente pour les noms et les adjectifs, est lévée grâce aux règles de double réduction qui permettent une désambiguïsation basée sur les contextes.

L'étape suivante, le chunking, crée un arbre de chunk avec les nœuds désambiguïsés et assemble les mots qui forment un groupe syntaxique. À part les chunks basiques comme les NP, VFIN, etc. il y a des chunks pour les subordonnées, et d'autres sont prévus au futur.

Les dépendances sont extraites en dernier, le traitement donne en sortie une liste de dépendances (des relations) que les mots de la phrase entretiennent entre eux. Il y a d'un côté les dépendances simples qui modifient des mots comme NMOD, VMOD, ..., mais aussi des dépendances pour des cas plus complexes, p.e. pour exprimer la relation du sujet, de l'objet, etc.

ParSem dispose maintenant de trois nouvelles grammaires, utilisables non seulement par Pocket Engineer, mais aussi pour des applications plus générales, comme par exemple la coréférence ou l'extraction d'entités nommées. Le fait que nous étions beta-testeurs de l'interface a beaucoup amélioré celle-ci par nos expériences et commentaires.

J'ai appris à concevoir des règles pour une grammaire de ma langue maternelle, l'allemand, en travaillant et discutant en français et anglais. J'ai surtout appris qu'une langue maternelle est aussi « compliquée » qu'une langue apprise plus tard, même si on a tendance à croire le contraire ! Créer des règles pour des phénomène que l'on rencontre tous les jours et que l'on croit simples n'a pas été si trivial que ça.

Ce qui reste à faire pour la deuxième partie de mon stage c'est l'accomplissement de la grammaire spécifique pour Pocket Engineer, que j'ai commencé il y a peu de temps. Il me faudra développer des règles plus fines, notamment pour pouvoir distinguer les questions des réponses dans la base de données, et reconnaître les synonymes dans les questions pour trouver les bonnes réponses, même si les mots de la requête ne correspondent pas aux mots-clés de l'indexation.

Un point qui reste à examiner sont les mots composés, très fréquents en allemand, surtout dans les textes de PE d'après ce que j'ai pu voir jusqu'à présent. Souvent un seul mot englobe un sens complexe qui est réalisé par une suite de mots (relié par la préposition *de*) dans les textes des langues latines. Un grand défi pour le PE allemand sera donc de décomposer correctement ces mots pour obtenir le sens global à partir des segments.

Car en fait, la décomposition étant gérée par l'analyse morphologique il y a souvent beaucoup trop de résultats (l'analyse retourne des multiples segmentations qu'il faudra désambiguïser). Pour le mot « Papierverbrauch » (*consommation de papier*) par exemple, on aura non seulement la bonne analyse en « Papier-verbrauch », mais aussi « Papier-verb-rauch ».

Et pour conclure mon stage, une évaluation des grammaires (la générale et la spécifique de PE) devra être réalisée. La première étape sera de déterminer la méthodologie pour ces évaluations. Ce sera une évaluation d'échantillons ou de problème précis (p.e. la dépendance sujet) pour la grammaire générale. Pour la grammaire de Pocket Engineer une évaluation thématique (trouver les bonnes informations dans le corpus) sera plus intéressante.

Mon projet de développement d'une grammaire de l'allemand pour XIP s'est situé au sein de l'équipe ParSem, un groupe de recherche dynamique et ouvert à de nouveaux projets. Le travail était très bien organisé, la communication entre les stagiaires et les responsables était toujours fructueuse et constructive.

Beaucoup de réunions et discussions avec les autres stagiaires et ma responsable m'ont permis d'avancer dans la bonne direction. L'expérience de participer à un travail de groupe fut une expérience intéressante et enrichissante pour moi.

Remerciements

Anne Schiller

Je voudrais remercier en premier ma responsable Anne Schiller qui a toujours été là pour répondre à mes questions. Chaque fois que j'avais un problème je savais que je pouvais aller la voir et en discuter. C'est elle qui a pris le temps toutes les semaines pour réfléchir avec nous sur nos grammaires et à leur avancement. J'ai beaucoup appris d'elle, surtout sur la langue allemande !

Aude, Elena et Giovanni

J'ai partagé le bureau avec les autres développeurs de grammaire, le travail en groupe était une belle expérience. Si quelqu'un avait un problème on en a discuté et chacun a dit ce qu'il en pense. Du coup tous en ont profité.

ParSem

Cette équipe m'a accueillie chaleureusement dans son sein et m'a offert la chance de faire un travail sur ma langue dans un domaine qui m'intéresse le plus dans mes études : le TALN. Le travail dans ce groupe de chercheurs ouvert à tout était très agréable, j'ai passé trois mois excellents et je suis contente de pouvoir continuer encore trois mois.

Mes parents et Stefan

Je suis très reconnaissante envers mes parents qui m'ont permis de faire des études intéressantes et qui m'ont soutenue tout le long. Leurs corrections et réflexions sur ce rapport m'ont beaucoup avancée.

Et last but not least je voudrais remercier l'homme de ma vie pour sa présence !

Bibliographie

Livres, articles et manuels

- *Grammatik der deutschen Gegenwartssprache. Duden Band 4.* Herausgegeben und bearbeitet von Günter Drosdowski in Zusammenarbeit mit Gerhard Augst, Hermann Gelhaus, Helmut Gipper, Max Mangold, Horst Sitta, Hans Wellmann und Christian Winkler. Dudenverlag, Mannheim, Wien, Zürich, 4., völlig neu bearbeitete und erweiterte Auflage, 1984.
- *Neue deutsche Grammatik.* Heinz Griesbach. Langenscheidt KG, Berlin, München, 1986.
- *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht.* Gerhard Helbig, Joachim Buscha. Langenscheidt, Verlag Enzykolpädie Leipzig, 15., durchgesehene Auflage, 1993.

- *German compound analysis with wfsc.* Anne Schiller. In : Finite State Methods and Natural Language Processing 2005, Helsinki, 1-2 September 2005.
- *Parsing By Chunks.* Steven Abney. In : Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing.* Kluwer Academic Publishers, Dordrecht, 1991.
- *Chunk Stylebook. Working draft.* Steven Abney, 1996.

- *XIP Reference Guide.* Xerox Corporation, 2001.
- *XIP User's Guide.* Xerox Corporation, 2001.
- *XIP Tutorial.* Salah Aït-Mokhtar, Jean-Pierre Chanod, Claude Roux, 2001.
- *XIP User's Guide (New Features).* Claude Roux, 2003.
- *Graphical User Interface for XIP.* Vianney Grassaud, 2005.

Sites internet Xerox :

- `http://www.xrce.xerox.com`
- `http://www.xerox.fr`
- `http://www.xerox.com`

Grammaire :

- `http://www.canoo.net`
- `http://www.sfs.nphil.uni-tuebingen.de/~abney/Papers.html`

A Quelques règles

Cette section contient une sélection de règles de la grammaire XIP pour l'allemand. Avant chaque groupe de règles se trouve le mot-clé qui informe XIP de quel type de règle il s'agit, p.e. `Tagging`, `Sequence`, etc.

A.1 Règles de désambiguïsation

```
Tagging:
// upper case words (other than at the beginning of a sentence)
 must be
// * noun or proper name
// * other defined upper case words (e.g. polite "Sie"), which
 are marked [upp:+]
// * special case: adjectives which are part of proper names
 ("Vereinigte Staaten")
1> ?<maj:+, start:~> = adj<proper=+>, noun, ?<upp:+> .

// adjectives at the beginning of a sentence
// "ALTE und Kranke"
7> ?<maj:+, start:+, noun:+, conv_adj:+, pl:+> = noun
  | conj[coord], noun |.
// "ALTE sind weise"
7> ?<maj:+, start:+, noun:+, conv_adj:+, pl:+> = noun
  | verb[pl:+] |.


DoubleReduction:
// (1) agreement of attributive adjectives and following noun
// "die GRÜNEN BÄUME"
22> | adj#1, ?*[noun:~, det:~], noun#2 | =>
        (  #1[agr] :: #2[agr] ).

// (2) agreement of determiner with following noun
// "EIN grüner BAUM"
23> | det#1, ?*[noun:~, det:~], noun#2 | =>
        (  #1[agr] :: #2[agr] ).


// (3) agreement of determiner with following adjective
// "EIN GRÜNER Baum"
```

```
24> | det#1, ?*[noun:~, det:~], adj#2 | =>
        (  #1[agr] :: #2[agr] ).

// for compounds:
// choose reading with minimal segment number
// "Donau-dampf-schiff-fahrt"
40> ? = ?[rank:~].
40> ? = ?[rank:1].
40> ? = ?[rank:2].
```

A.2 Règles de chunking

```
Sequence:
// simple NPs
// personal pronoun
// "ICH, DU, ER, ..."
115> np = pron[pers, !agr:!, !person:!].

// noun with adjective
// "der GRÜNE BAUM"
115> np[p3=+] = det#1[ !wh:+!, !agr:! ],
        ( num[ card, !agr:! ]; num[ ord, dig:+] ),
        ( ap[ !agr:! ] ),
        noun#2[ !agr:!, symbol:~ ].

// proper names
// "Marie Schmidt"
120> np[human=+, p3=+] = noun[firstname], noun[lastname,
 !agr:! ].

// finite verbs
// "er LÄUFT"
200> vfin =  verb#2[finite, !modal:!, !subjc:!, !indic:!,
 !nursubjc:!].
```

A.3 Règles de dépendance

```
DependencyRules:
// modifier of the noun
// "SCHÖNE, GRÜNE Bäume"
| ap{?*, adj#1}, #2[noun,last] |
//      if ( #1[case]:#2[case] & #1[number]:#2[number] & #1[pred:~] )
        NMOD(#2, #1).

// coordinated nouns
// "Kinder UND Jugendliche"
| np{?*, #1[last]}, (pp; adv)*, conj[coord:+, subord:~], np{?*,
 #2[last, rel:~]} |
        if ( #1[case]:#2[case] )
        COORDITEMS(#1, #2).

// main verb
// "sie TRINKT Sprudel"
| vfin{verb#1[modal:~, aux:~]} |
        VMAIN(#1).

// auxiliary verbs
// "sie HAT Sprudel GETRUNKEN"
| vfin{verb#1[aux]}, ?*[verb:~, finite:~, aux:~, punct:~,
 subord:~], verb#2[verb, finite:~] |
        VMAIN(#2), AUXIL(#1, #2).

// determiner rules
// "DER Baum"
| det#1[first, !def:!, !dem:!, !neg:!, !poss:!, !quant:!,
 !wh:!], ?*, noun#2[last]; nmod{?*, #2[last]}; np{?*, #2[last]} |
        if ( #1[agr]:#2[agr] )
        DETERM(#2, #1).

// Rules for SUBJ :
// "MARIE kauft einen Drucker."
| np#3[nom, rel:~]{?*, #1[last]}, (np[gen]), ((adj), adv; pp)*,
 vfin{?*, #2[last]} |
        if ( #2[number]:#3[number] & ~MOD_GEN(#10, #1) )
        SUBJ(#2, #1).
```

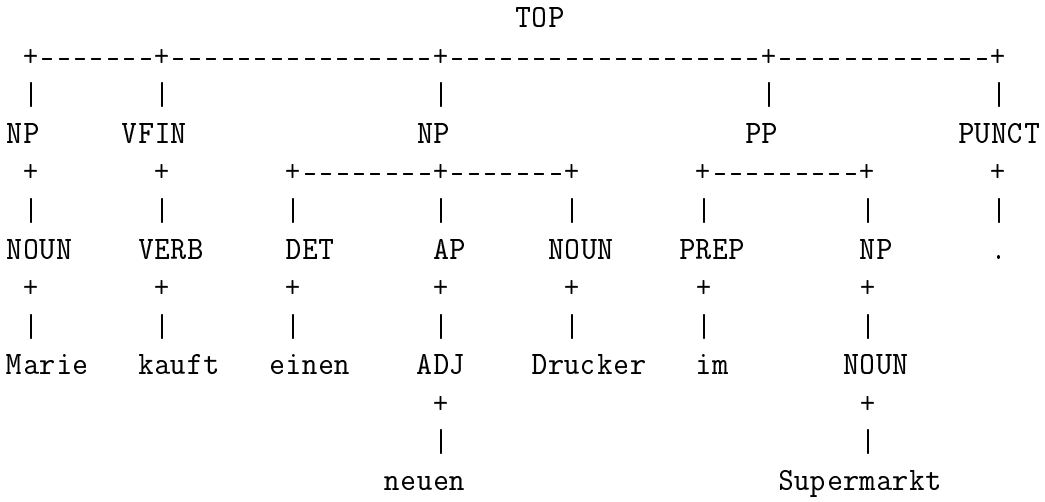
```
// Rules for OBJ_ACC :
// the object comes after the verb
// "Marie kauft einen DRUCKER."
| np{?*, #10[last]}, (pp; adv; np[gen:+])*, vfin{?*, #2[last]},
 ?*[subord:~, punct:~, verb:~], np#3[acc:+]{?*, #1[last]} |
        if ( SUBJ(#2, #10) & ~PRED(#10, #1) & ~SUBJ(#5, #1) )
        OBJ_ACC(#2,#1).

// the object comes before the verbe
// "Einen Drucker kauft Marie."
| np#3[acc:+]{?*, #1[last]}, (pp;adv)*, vfin{?*, #2[last]},
 np{?*, #10[last]} |
        if ( SUBJ(#2, #10) )
        OBJ_ACC(#2, #1) .

// Rules for the subject of a subordinated clause
// "Peter weiß, dass MARIE einen Drucker gekauft hat."
| np[nom]{?*,#1[last]}, vfin{?*, #2[last]}, ?*[finite:~],
 (punct[comma:+]), conj#3[subord:+]; adv#3[wh:+],
 np[nom]{?*,#4[last]}, *?[finite:~], vfin{?*, #5[last]} |
        if ( SUBJ(#2, #1) & OBJ_SENT(#2, #3) & ^SUBJ(#5, #4) )
        SUBJ_BG(#5, #4) .
```

B Une sortie de XIP

XIP livre la sortie suivante pour la phrase « Marie kauft einen neuen Drucker im Supermarkt. » (*Marie achète une nouvelle imprimante au supermarché.*).




```
0> Marie kauft einen neuen Drucker im Supermarkt.
NMOD_[74](Drucker,neu)
VMAIN_[94](kaufen)
DETERM_[116](Drucker,ein)
SUBJ_[119](kaufen,Marie)
OBJ_ACC_[146](kaufen,Drucker)
0>TOP{NP{Marie} VFIN{kaufen} NP{ein AP{neu} Drucker}
 PP{in NP{Super^#Markt}} .}
```