Robustesse et analyse syntaxique

Philippe Blache, Stéphane Rauzy

Vers une recherche d'information robuste et personnalisée

Patrice BELLOT

22 juin 2011

PREMIÈRE PARTIE

Robustesse

Chapitre 3

Robustesse et analyse syntaxique

3.1. Introduction

Pour le traitement automatique des langues, la robustesse d'une application se mesure à sa capacité à résister aux erreurs. Celles-ci peuvent provenir soit d'une défaillance du système, soit d'une difficulté linguistique inhérente au texte ou à l'énoncé traité. Dans un cas comme dans l'autre, un système robuste devra être capable de poursuivre son traitement malgré l'erreur.

La question de la robustesse est donc un problème d'ingénierie, dans sa dimension de résistance à un environnement non homogène. En particulier, ce type de comportement constitue un pré-requis dans certaines situations comme le traitement de corpus volumineux ou de données non standards. Cependant, la robustesse n'est pas nécessairement liée à la capacité du système à identifier, décrire, voire rattraper des erreurs. Il s'agit également d'un problème théorique dans la mesure où les problèmes rencontrés concernent soit la grammaire utilisée, soit la capacité à traiter du matériel linguistique non canonique. Dans les deux cas, le problème se pose au niveau de la compétence linguistique du système, se traduisant par une grammaire ou, plus généralement, un mécanisme d'analyse suffisamment général. En TALN nous pouvons donc dire que la question de la robustesse se traduit à la fois par des techniques ou des stratégies spécifiques (voir [CHA 01, BRI 06]), mais également par le développement de ressources voire de formalismes linguistiques plus adaptés ([MEN 98]). La robustesse constitue donc une question essentielle, tout en restant finalement une notion peu précise, très dépendante du type de traitement à effectuer ainsi que du type de texte.

Chapitre rédigé par Philippe BLACHE et Stéphane RAUZY.

La question de la robustesse se pose de façon particulière dans le domaine de la recherche d'information ([LEW 96, STR 94]). En effet, bon nombre de techniques de RI n'exploitent finalement que peu d'informations linguistiques et ne nécessitent pas véritablement d'analyse linguistique détaillée. On remarque cependant que les progrès réalisés dans le domaine du traitement sémantique commencent à utiliser des analyses dépassant le niveau lexical, nécessitant des techniques plus sophistiquées permettant d'effectuer des traitements prenant en compte les unités syntaxiques ainsi que les relations les reliant. La RI est donc également concernée par cette évolution. De plus, ce domaine pose des problèmes spécifiques pouvant nécessiter des analyses plus fines (compréhension de questions, requêtes multimodales, comparaison de textes, etc.). Nous sommes finalement aujourd'hui confrontés, en RI comme dans les autres domaines du traitement des langues, à cette question de la robustesse, nécessitant le traitement de données disparates, non canoniques, partielles, etc.

Nous proposons dans cet article d'aborder cette question en commençant par décrire plus précisément les situations conduisant les systèmes à des erreurs. L'étude des besoins spécifiques à la RI nous permettra d'identifier plus clairement les points à traiter pour proposer un traitement robuste permettant une analyse linguistique fine. Nous nous concentrerons sur la question de l' analyse syntaxique, qui constitue une étape essentielle dans les traitements en profondeur. Ce domaine a longtemps été laissé de côté dans les systèmes, en partie à cause de son coût, mais également de son manque de robustesse. Nous présenterons ici quelques techniques permettant de répondre à ces besoins. Nous décrirons en particulier une approche basée sur les contraintes offrant l'avantage d'être à la fois robuste, cohérente d'un point de vue formel, et capable de répondre aux évolutions futures notamment en termes de traitement de la multimodalité.

3.2. Les situations

Il est intéressant de décrire plus précisément les situations confrontant potentiellement un système à une erreur. Nous l'avons vu plus haut, ces erreurs peuvent soit provenir du système lui-même, soit d'une situation inattendue provenant de l'*input*.

Dans le premier cas, il s'agit essentiellement de deux types de problèmes :

– *défaut de couverture* : le système (par exemple l'analyseur syntaxique) ne permet pas de traiter tous les phénomènes. S'il s'agit d'un système symbolique, la grammaire n'est donc pas assez couvrante et la solution à ce problème consiste à augmenter cette couverture. Il s'agit cependant d'un travail énorme, pour ne pas dire sans fin, aucune grammaire ne permettant de couvrir en totalité une langue. S'il s'agit de systèmes reposant sur des techniques numériques, c'est le corpus d'apprentissage qu'il faut étendre, de façon à prendre en compte ces nouveaux phénomènes. Là encore, et pour les mêmes raisons, il est toujours possible d'améliorer la couverture du système,

mais sans pouvoir espérer une couverture totale. Remarquons au passage que l'amélioration de la couverture d'un analyseur syntaxique est plus facile dans le cas des systèmes symboliques ;

– *erreur de traitement* : tout système peut commettre des erreurs à chaque niveau de traitement, pouvant entraîner des conséquences problématiques pour les étapes ultérieures. Typiquement, une erreur dans la phase d' étiquetage (qui consiste à associer aux mots leurs catégories morpho-syntaxiques) aura bien entendu des conséquences sur l' analyse syntaxique à proprement parler. La conséquence peut être plus ou moins grave selon l'erreur : elle peut conduire à une mauvaise structure syntaxique ou, plus grave, à l'impossibilité de construire une structure. Il faut remarquer que les étiqueteurs permettant de désambiguiser les catégories à affecter tiennent compte du contexte proche et donc effectuent une partie de l' analyse syntaxique. Une erreur d' étiquetage se propagera généralement aux catégories voisines, contaminant ainsi l'ensemble de l'énoncé tout en permettant généralement de parvenir à une structure syntaxique licite, mais ne constituant pas la bonne analyse de la phrase. Ce type d'erreur, en phase de traitement complètement automatique, n'est pas possible à récupérer, le résultat étant considéré comme bien formé. En revanche, une situation de blocage total conduit les systèmes robustes à déclencher des traitements spécifiques.

Le second type de problème pouvant conduire à des erreurs de traitement provient de la forme de l'input lui-même. Là encore, deux situations peuvent se produire :

– l'*input* est *corrompu* par un pré-traitement. Il s'agit typiquement d'inputs provenant de systèmes de reconnaissance de la parole pouvant conduire à des mots mal reconnus ou des mots inconnus. Le premier cas correspond en termes de traitement à celui de l'erreur d' étiquetage, aux conséquences décrites plus haut en termes de syntaxe. Le second cas est plus problématique car aucune possibilité de rattrapage de la reconnaissance n'est envisageable. Ce type de situation se retrouve de façon plus prégnante dans le cas où l'entrée est multimodale (voir par exemple [BAN 09]), pouvant conduire à une mauvaise reconnaissance d'une ou plusieurs modalités, ayant pour conséquence une impossibilité d'interprétation ;

– l'*input* contient des *phénomènes spécifiques*, difficiles ou impossibles à traiter. C'est typiquement le cas du traitement de la langue parlée, sujette en particulier à des phénomènes de disfluence (par exemple des répétitions, des mots ou des syntagmes incomplets, etc.). Nous nous retrouvons ici dans une situation comparable à celle du défaut de couverture : le système n'est pas capable de reconnaître une construction particulière et ne peut la traiter. Là encore, selon le comportement du système, ce type de problème peut conduire à un blocage.

Dans les deux cas, les problèmes ayant conduit à un échec auront des conséquences similaires et seront traitées, du point de vue de la robustesse, avec des techniques comparables voire identiques. En d'autres termes, la robustesse d'un système peut

être vue comme sa capacité à traiter des entrées mal formées, que cela provienne d'un défaut du pré-traitement ou d'une forme particulière de l'*input*.

3.3. Le cas de la recherche d'information

La recherche d'information a ceci de particulier qu'elle est amenée à traiter du matériel linguistique à la fois dans les documents susceptibles de contenir l'information, mais également dans les requêtes. Typiquement, une partie de l'efficacité des systèmes de question-réponse repose sur la qualité de l'interprétation de la question, nécessitant donc une analyse fine. En revanche, le traitement du matériel contenu dans les documents peut se contenter d'une analyse superficielle, voire se passer complètement de traitement de haut niveau.

Pour ce qui concerne la partie "recherche d'information" à proprement parler, les techniques sont variées et les avis diffèrent quant à l'utilisation de techniques d'analyse linguistiques sophistiquées. Sur le principe, il tombe sous le sens qu'une information plus riche devrait permettre une recherche plus précise. Cependant, ceci peut s'avérer inexact selon le type de traitement. Par exemple, les techniques de lemmatisation permettent d'élargir la recherche de termes au delà d'une forme unique. Une requête portant ainsi sur la forme "*recherche*" permettra de prendre en compte des termes voisins morphologiquement comme "*rechercher*", "*recherché*" ainsi que toutes les flexions du verbe. Il est vrai que dans certains cas, une analyse trop superficielle risque d'introduire des ambiguïtés voire de conduire à de fausses interprétations. Mais ce risque disparaît, ou du moins diminue lorsque l'analyse est suffisamment détaillée. De la même façon, une recherche portant sur les termes "*recherche d'information*" pourrait retourner des documents concernant "*informations sur les avis de recherche*". Ce type de problème disparaît dès lors que la requête prend en compte la notion de syntagme.

Sans entrer dans ce débat, nous prenons ici le parti de considérer qu'un apport d'informations linguistiques, notamment syntaxiques, est de nature à améliorer la précision de la recherche. Dans ce cas, il convient donc d'identifier le niveau d'analyse adéquat. Il n'est bien entendu pas envisageable d'effectuer une analyse détaillée des documents. Cependant, compte tenu des résultats des systèmes actuels, une analyse superficielle peut être envisagée. Il s'agit dans ce cas de repérer (indépendamment de leur structuration) les grandes unités syntaxiques et les principales relations. Ce type de traitement a été analysé à l'occasion de plusieurs campagnes d'évaluations (Easy, et Passage) qui ont permis de montrer l'efficacité de certains systèmes, capables de traiter 2 à 300 000 mots par minute en offrant une F-mesure de plus de 90 %.

3.4. Les techniques

Nous proposons dans cette section une rapide présentation de quelques techniques robustes. Nous donnons à titre illustratif des exemples plus précis de techniques que nous avons développées au LPL. Après une description d'une technique d' étiquetage robuste et efficace, nous abordons le problème de l' analyse syntaxique.

3.4.1. *Etiquetage et désambiguisation*

La procédure de désambiguisation consiste à associer à chacun des *tokens* de l'énoncé une catégorie morphosyntaxique unique. Dans notre approche, nous utilisons le modèle des patrons [BLA 06, BLA 07], un modèle de Markov caché (HMM) plus performant que les modèles de type *n-grammes*. Pour les n-grammes, les états de l'automate sont identifiés par des séquences de catégories de taille identique $n-1$. Le modèle des patrons relâche cette contrainte en acceptant des états identifiés par des séquences de longueur variable, voir par exemple [RON 96]. Cette caractéristique permet en pratique d'extraire du corpus d'apprentissage un ensemble d'états, les *patrons* du modèle, qui capture de façon optimale l'information contenue dans le corpus.

Chaque patron du modèle est caractérisé par :

– son identifiant, c'est-à-dire la séquence de catégories qui le constitue, par exemple la suite de catégories `Det Adj Noun` ;

– un vecteur donnant la probabilité de transition vers chaque catégorie du modèle, conditionnée par l'identifiant du patron, par exemple $\mathtt{p}(\mathtt{Verb} \mid \mathtt{Det\ Adj\ Noun})$[1] pour la probabilité de transition vers la catégorie `Verb` ;

– un vecteur donnant pour chaque transition possible, le nouvel état atteint par le système (le *patron cible*). Par exemple, si au système dans l'état `Det Adj Noun` est proposé la catégorie `Verb`, le système évolue vers l'état `Noun Verb`.

Par construction, le patron cible associé à une transition ne dépend pas du chemin suivi dans l'espace des états du système. Cette propriété permet en pratique d'utiliser des algorithmes de programmation dynamique très efficaces, comme l'algorithme de Viterbi, pour calculer la solution la plus probable. Ici, on ne suppose pas que les données vérifient effectivement cette hypothèse d'indépendance. Le modèle des patrons est vu comme une approximation du modèle statistique décrivant réellement les données, une approximation qui est d'autant plus fidèle que le corpus d'apprentissage possède une taille importante. Des phénomènes comme les dépendances non bornées par exemple, ne seront pas capturés par le modèle des patrons.

---

1. $\mathtt{p}(\mathtt{Verb} \mid \mathtt{Det\ Adj\ Noun})$ est la probabilité de `Verb` conditionnée par le contexte gauche `Det Adj Noun`.

Pour la tâche de désambiguisation, le problème consiste à trouver pour la séquence de *tokens*, chaque *token* proposant sa liste de catégories potentiellement réalisables, la séquence de catégories de probabilité maximale. Ceci équivaut à trouver le chemin optimal suivi dans le treillis des états du système compte tenu des choix proposés par la séquence de *tokens* en entrée. Cette solution est obtenue par application de l'algorithme de Viterbi.

Le modèle est entrainé sur le corpus Grace/Multitag [PAR 00], un échantillon d'environ 700 000 mots annoté morphosyntaxiquement selon le jeu de traits Multext [IDE 94]. L'information morphosyntaxique disponible a été groupée de manière *ad hoc* en 44 catégories distinctes (2 types de catégories pour les ponctuations, 1 pour les interjections, 2 pour les adjectifs, 2 pour les conjonctions, 1 pour les déterminants, 3 pour les noms, 8 pour les auxiliaires, 4 pour les verbes, 5 pour les prépositions, 3 pour les adverbes et 11 pour les pronoms). Les informations comme les traits d'accords en genre, nombre et personne ou le temps des verbes ne sont pas exploitées dans la version actuelle de l'étiqueteur.

Le modèle des patrons complet est composé de 3 053 patrons de taille variable (le plus grand contexte dans la liste des patrons est composé d'une séquence de 6 catégories). L'évaluation du modèle est réalisé en comparant l' étiquetage optimal obtenu à partir du modèle (par application de l'algorithme de Viterbi) et la référence. Pour le jeu de catégories sélectionnées (les 44 catégories mentionnées précédemment), la qualité de l'étiqueteur atteint un score de $0,94$ (F-mesure), voir [BLA 08] pour plus de détails.

3.4.2. *analyse syntaxique*

Pour ce qui concerne l' analyse syntaxique à proprement parler, il est possible de regrouper les approches permettant d'améliorer la robustesse des systèmes autour de trois familles de techniques :

– réduction de la complexité du processus d'analyse par ajout de contrôles ou heuristiques

– adaptation des systèmes de façon spécifique au type de matériel contenu dans l'*input*

– réduction de la complexité de l'analyse : systèmes proposant une analyse superficielle

Le premier ensemble recouvre un grand nombre de techniques. Le but est dans ce cas de contrôler une stratégie d'analyse donnée au moyen de mécanismes spécifiques proposant par exemple de déclencher des systèmes de récupération d'erreurs ([BOU 05]), ou encore de contrôler le processus d'analyse en s'appuyant sur des informations probabilistes ([JOH 98]).

Les techniques du second type consistent quant à elles à adapter les ressources au type de textes ou d'énoncés traités. On retrouve dans ces approches les techniques reposant sur des grammaires de fautes, des lexiques et des grammaires spécialisées, etc. Ces techniques visent à réduire l'espace de recherche des analyseurs en s'appuyant sur des règles, des informations ou des traitements ad hoc.

La dernière famille de techniques consiste à réduire la complexité de l'analyse en proposant d'effectuer une analyse superficielle (voir par exemple [HIN 83], [ABN 96]). L'idée consiste à construire des structures contenant peu d'information, ou éventuellement une information sous-spécifiée (ce qui nécessite pour les systèmes la possibilité de construire des structures partielles). Il existe plusieurs stratégies permettant d'effectuer une analyse superficielle robuste. Bien entendu, s'agissant d'une tâche d'identification de frontières, les approches probabilistes sont d'une grande efficacité tout en étant par principe robustes (voir [TJO 00] pour une comparaison entre différents types d'analyseurs superficiels). Mais des approches symboliques peuvent également être proposées pour ce type de tâche. Ces techniques sont très efficaces et largement utilisées. Elles peuvent également être elles-mêmes utilisées comme techniques de contrôle d'un processus d'analyse approfondi ([CRY 02], [USZ 02], [MAR 02]).

Nous allons dans le reste de cette section détailler ces techniques en les illustrant avec deux types d'approches que nous avons développées : une approche symbolique, l'autre stochastique, représentatives des deux grands courants dans ce domaine.

3.4.2.1. *analyse syntaxique superficielle symbolique*

Une stratégie symbolique robuste consiste à identifier les frontières gauches des unités syntaxiques (également appelées coins gauches, [ROS 70]). Une telle stratégie, utilisée habituellement pour contrôler une analyse syntaxique ascendante, peut à certaines conditions suffire à identifier les unités syntaxiques élémentaires (ou *chunks*), sans avoir recours à des règles. La technique consiste à créer des tables de constituante et de précédence indiquant pour chaque catégorie si elle peut être frontière gauche d'un syntagme d'un type donné. Le processus fonctionne ensuite en deux temps :

```
si la catégorie courante est coin gauche potentiel d'un groupe de type SX
   si (la catégorie précédente n'est pas coin gauche d'un groupe SX)
   ou si (elle n'appartient pas à un groupe de type SX)
      alors une frontière gauche de type SX est posée
```

Le processus est complété par un traitement plus fin, autorisant dans certains cas la juxtaposition de syntagmes de même type composés d'un seul mot ou encore empêchant en fonction du contexte l'ouverture d'un nouveau syntagme.

L'analyseur superficiel symbolique que nous avons développé est déterministe. Il repose sur les *Grammaires de Propriétés* ([BLA 01]) avec une stratégie de coin

gauche. La grammaire utilisée est complète en ce sens qu'elle peut être utilisée indifféremment pour une analyse profonde ou superficielle [BAL 05]. La particularité de cet analyseur est de s'appuyer sur un sous-ensemble de contraintes de la grammaire (en particulier les propriétés de linéarité et de constituance) pour identifier les coins gauches. La stratégie consiste à repérer à partir des coins gauches la frontière droite du *chunk* sur la base des autres propriétés. Cette heuristique est très efficace et permet à l'analyseur de bénéficier d'une grande rapidité (moins de 4 minutes pour traiter 1M de mots). Dans le cadre de la campagne Passage 2007, cet analyseur a obtenu une F-Mesure de $91,57\,\%$.

Cet analyseur a été adapté pour les besoin des campagnes d'évaluation Easy et Passage (voir [PAR 00], [CLE 08]) au format de sortie requis. Au delà des *chunks*, il produit cependant des arbres syntaxiques complets. Il a ainsi été utilisé pour produire l'étape initiale du *treebank* du LPL : 1 500 arbres, générés par cet analyseur, ont ensuite été corrigés manuellement ; formant ainsi une ressource syntaxique complémentaire à celles existantes.

3.4.3. *analyse syntaxique superficielle stochastique*

L'analyseur stochastique superficiel que nous avons développé est basé, comme l'étiqueteur décrit plus haut, sur le modèle des patrons. La grammaire de *chunks* utilisée est celle développée dans le cadre des campagnes Easy et Passage [GEN 03]. Elle compte six constituants non emboîtables : les groupes `GN` (groupe nominal), `GP` (groupe prépositionnel), `NV` (noyau verbal), `GA` (groupe adjectival), `PV` (groupe verbal introduit par une préposition) et `GR` (groupe adverbial). Dans notre représentation, les catégories non terminales sont annotées par leur catégorie ouvrante et fermante associées, par exemple `<GN>` et `</GN>` pour le groupe `GN`. Pour chaque catégorie terminale (c'est-à-dire les 44 catégories terminales introduites pour l'étiqueteur), on définit six catégories mentionnant le groupe dominant la catégorie et une catégorie supplémentaire lorsque la catégorie est isolée. Par exemple `GN/Noun` représente la catégorie `Noun` dans un groupe `GN`, `GP/Noun` représente la catégorie `Noun` dans un groupe `GP`, `./Pct` la catégorie `Pct` isolée. Au total, si 320 catégories sont définies, beaucoup d'entre elles ne sont pas présentes en pratique, par exemple `GN/Verb` (il n'y a pas de verbe dans un groupe nominal).

La phase d'apprentissage est effectuée sur le *gold standard* Easy, un corpus annoté en constituants d'environ 100 000 mots qui a servi de référence pour la campagne d'évaluation Easy [PAR 06]. Le corpus ne fournit pas les étiquettes des *tokens* composant les énoncés. Une phase préalable d' étiquetage (en utilisant l'étiqueteur présenté plus haut) a donc été nécessaire pour produire notre échantillon d'apprentissage. L'échantillon est ensuite encodé selon la définition des catégories décrites ci-dessus.

Le module d'apprentissage nous permet d'extraire de cet échantillon 1 340 patrons de taille variable identifiés par des séquences de catégories terminales ou de catégories

ouvrantes et fermantes associées aux groupes. Pour la tâche qui nous concerne ici, c'est-à-dire insérer au sein d'une séquence de catégories observées (les catégories terminales) les catégories non observées (les catégories ouvrantes et fermantes), une étape supplémentaire est requise.

Pour chaque patron du modèle dont l'identifiant se termine par une catégorie observée (*patron observé*), et pour chaque catégorie observée, est établie la liste des transitions possibles vers cette catégorie, avec ou sans insertion. Pour chaque item de cette liste est calculée la probabilité de transition, le patron cible et le vecteur de catégories non observées insérées pour réaliser cette transition. Par exemple, pour le patron observé `GN/Det GN/Noun` et pour la catégorie observée `NV/Verb`, il existe une transition vers le patron cible `<NV> NV/Verb` avec insertion des catégories `</GN> <NV>`. A la fin de la procédure, nous obtenons un modèle constitué de 1.080 patrons observés.

L'analyseur stochastique prend en entrée un texte étiqueté et désambiguisé ou une liste de catégories associée à chaque *token* de l'énoncé (la sortie de la phase segmentation plus accès au lexique). Pour chaque énoncé, l'algorithme de Viterbi permet d'insérer les groupes Easy maximisant la probabilité de l'énoncé. Dans le cadre de la campagne Passage 2007, cet analyseur a obtenu une F-Mesure de $93,03\,\%$.

3.4.4. *Bilan*

Ces différentes stratégies peuvent éventuellement être combinées entre elles. Leur intérêt réside dans le fait que, en plus du fait qu'elles proposent des solutions au problème de la robustesse, elles sont d'une grande efficacité, notamment en termes de temps de traitement. Elles sont donc adaptées au traitement de grandes masses de données. Cependant, elles ne constituent pas une solution générique dans le sens où une simplification doit être apportée au formalisme utilisé, au format de sortie ou au processus lui-même. En d'autres termes, il s'agit d'une adaptation du cadre initial.

La principale difficulté de l'analyse de phénomènes variés (voire non canoniques) vient du fait que les modèles disponibles sont trop stricts. C'est en particulier le cas des modèles génératives pour lesquels une phrase peut être analysée si et seulement si elle peut être est générée par la grammaire. Ces modèles sont basés sur des règles permettant de décrire le mécanisme de construction de la structure syntaxique. Chaque règle est une étape dans la construction de l'arbre, correspondant ainsi à une étape dans le processus de dérivation appliqué pour générer un *input* à partir d'une grammaire donnée. Un échec dans le processus de dérivation signifie qu'aucune règle ne peut être appliquée, l'arbre ne peut alors être construit et le processus ne peut se poursuivre. Dans une telle approche, le traitement de phénomènes non canoniques n'est donc pas possible en dehors de l'application d'heuristiques ou de traitements *ad hoc*.

Il existe cependant des techniques de haut niveau permettant à la fois une analyse approfondie et robuste, sans nécessiter de modification du formalisme ou du cadre de

travail. Il s'agit des approches basées sur les contraintes ([POL 96]), offrant une plus grande souplesse pour plusieurs raisons. En particulier, les contraintes, à la différence des règles, peuvent être relâchées si elles ne sont pas satisfaites. L'analyse qui se poursuit est alors la recherche d'une solution optimale compte tenu du contexte plutôt que la recherche de la bonne solution.

Nous allons consacrer la section suivante à une présentation détaillée de leur fonctionnement.

3.5. Une approche syntaxique intrinsèquement robuste : les *Grammaires de Propriétés*

La notion de contraintes est désormais d'un usage largement répandu en traitement automatique des langues. Elles peuvent être utilisées à un bas niveau comme mécanisme de filtrage, comme dans les *Grammaires de contraintes* ([KAR 90]) ou au contraire, être partie intégrante de la théorie comme pour *HPSG* (voir [SAG 03]), la *Théorie de l'Optimalité* ([PRI 93]) ou les *Grammaires de Dépendance par contraintes* ([MAR 90]).

Notre approche est cependant différente de celles évoquées dans le sens où toutes les informations sont représentées sous forme de contraintes, sans s'appuyer sur la construction préalable ou parallèle d'une structure syntaxique quelconque. Cet aspect est particulièrement important pour le problème qui nous intéresse ici : il s'agit d'une approche intrinsèquement robuste dans le sens où ce qui est construit n'est pas une structure correspondant à l'*input* analysé (par exemple un arbre), mais une description de ses propriétés. Le mécanisme d'analyse peut être alors vu comme un mécanisme de satisfaction de contraintes au lieu d'un mécanisme de dérivation. De plus, il devient possible, quelle que soit la forme de l'entrée (canonique ou pas), de lui donner une description. Cette technique exploite le relâchement de contraintes, elle peut être contrôlée par des heuristiques du type de celles présentées plus haut, comme la recherche de coins gauches. Un des avantages principaux de cette approche est que la même technique d'analyse est utilisée, quelle que soit la forme de l'*input*, sans avoir besoin de déclencher des traitements spécifiques consécutifs à l'identification d'une erreur. De plus, ce type d'approche permet de fournir si besoin est des analyses partielles, ce qui peut revêtir un grand intérêt pour la recherche d'information.

3.5.1. *Les caractéristiques des Grammaires de Propriétés*

Les *Grammaires de Propriétés* ([BLA 01]) permettent de représenter l'information syntaxique de façon décentralisée et locale. Là où les approches classiques manipulent des structures, les GP permettent de spécifier des propriétés sur des ensembles de catégories, voire de traits, indépendamment de toute structure. Cette caractéristique est

essentielle pour le traitement de données incomplètes, partielles ou non canoniques. Nous sommes ainsi en mesure d'exprimer des relations entre deux objets indépendamment de leur position dans la chaîne ou dans une structure. De plus, la description d'une unité syntaxique – nous parlons de *construction* – se fait en tenant compte aussi bien des propriétés satisfaites que de celles qui ne le sont pas. Nous sommes ainsi en mesure de proposer une description très fine de l'information, y compris dans le cas de données non canoniques ou non grammaticales.

Sans entrer dans les détails des *Grammaires de Propriétés*, (notées dorénavant GP), rappelons-en malgré tout les grandes lignes. La caractéristique essentielle des GP repose sur l'idée qu'il est possible de représenter toute l'information syntaxique à l'aide de propriétés ([BLA 01]). Celles-ci sont des relations entre deux ou plusieurs catégories. Les propriétés expriment différents types d'information : l'ordre linéaire, la cooccurrence impérative ou impossible, la dépendance, la répétition, etc. Cette liste est évolutive, il est toujours possible d'ajouter de nouveaux types de propriétés, à condition bien entendu d'en préciser la sémantique. Une catégorie syntaxique est ainsi décrite dans la grammaire par un ensemble de propriétés mettant en relation ses constituants. Voici par exemple quelques propriétés décrivant la construction *SV* :

Linéarité	V $\prec$ SN
Cooccurrence	Aux $\Rightarrow$ V[ppas]
Dépendance	SN $\rightsquigarrow$ V
Restriction de cooccurrence	V[intrans] $\not\Rightarrow$ SP

L'analyse d'un énoncé consiste donc à vérifier pour chaque construction de l'énoncé l'ensemble des propriétés qui lui correspondent dans la grammaire. Certaines de ces propriétés peuvent être satisfaites, tandis que d'autres peuvent être enfreintes. Nous obtenons ainsi un ensemble de propriété évaluées (satisfaites ou pas) que nous appelons "*caractérisation*". Une telle approche permet donc de décrire n'importe quel type de construction : partielle, discontinue, non canonique, etc., répondant ainsi à l'objectif exprimé plus haut.

Cette souplesse d'utilisation a cependant un coût : le problème de l'analyse en GP est en effet d'une complexité exponentielle (cf. [VAN 05]). Cette complexité, nous y reviendrons, vient tout d'abord de la possibilité offerte de prendre en considération toutes les unités, indépendamment de leur position. Ceci nous permet de prendre en compte des relations distantes ou non projectives entre deux unités (dépendance à distance, constituants discontinus, dépendances croisées). Par ailleurs, l'analyse d'entrées non canoniques repose sur la possibilité de construire des descriptions à l'aide de propriétés plus ou moins satisfaites. En termes d'implémentation, une propriété étant une contrainte, il faut proposer une possibilité de relâchement de contrainte. Ces deux facteurs sont essentiels dans la complexité du problème.

3.5.2. *Les mécanismes d'analyse en GP*

Nous décrivons dans cette section un mécanisme théorique – et naïf – d' analyse syntaxique en GP. Il ne s'agit pas d'une stratégie d'analyse, mais d'un schéma permettant de décrire les facteurs de complexité.

Le processus d'analyse consiste à instancier l'ensemble des constructions décrivant un énoncé. Ce mécanisme revient en fait à produire les caractérisations leur correspondant. Ce processus nécessite l'identification des constituants entrant en jeu dans la construction, afin d'en vérifier les propriétés pertinentes. Les constructions pouvant être discontinues, il convient donc d'effectuer cette vérification sur toutes les combinaisons de constituants, en d'autres termes l'ensemble des sous-ensembles possibles de catégories prises en considération. Nous appelons *affectation* chaque combinaison de constituants. L'ensemble des catégories de départ est ainsi formé par l'ensemble des catégories correspondant aux mots de l'énoncé à analyser avec leur position dans l'énoncé. Toutes les *affectations*, construites à partir de cet ensemble, sont évalués. Ceci revient pour chaque affectation à parcourir la grammaire, autrement dit, l'ensemble des propriétés en les évaluant lorsque c'est possible. Pour certaines affectations, aucune propriété n'est pertinente et la caractérisation construite est un ensemble vide et l'affectation est non productive. En revanche, pour d'autres, une caractérisation peut être construite. Au premier niveau de l'analyse, toutes les constructions ont pour constituants des catégories lexicales, comme dans les exemples suivants :

Construction	*Constituants*	*Caractérisation*
SA	{Adv, Adj}	{Adv $\prec$ Adj ; Adv $\rightsquigarrow$ Adj ; ...}
SN	{Det, N}	{Det $\prec$ N ; Det $\rightsquigarrow$ N ; N $\not\Rightarrow$ Pro ; ...}

Une construction instanciée a pour conséquence d'ajouter son étiquette à l'ensemble des catégories à prendre en considération. Dans les exemples précédents, les catégories *SA* et *SN* sont ainsi ajoutées à l'ensemble des catégories lexicales de départ. Un nouvel ensemble d'affectations peut alors être construit, incluant ces nouvelles catégories, permettant ainsi d'identifier de nouvelles constructions. Le mécanisme global d'analyse est décrit par le schéma d'algorithme suivant :

```
Initialisation
   Pour chaque mot à une position i de l'énoncé
      créer l'ensemble des c_i, ses catégories possibles
   K ← {c_i | 1<i<nombre de mots}
   S ← ensemble des sous-ensembles de K
Répéter
   Pour chaque S_i ∈ S
      si S_i a une caractérisation productive
         ajouter k_i l'étiquette de la caractérisation à K
   S ← ensemble des sous-ensembles de K
Tant que de nouvelles caractérisations sont produites
```

Ce processus d'analyse illustre la complexité liée au nombre d'affectations à prendre en compte, celles-ci devant être reconstruites à chaque étape, c'est à dire chaque fois qu'une nouvelle construction est identifiée. A chaque niveau, il faut donc régénérer l'ensemble des sous-ensembles de catégories.

Précisons maintenant l'opération de *caractérisation*. Le processus consiste, étant donné une affectation, à vérifier toutes les propriétés évaluables de la grammaire. Une propriété *p* est évaluable lorsque l'affectation $\mathcal{A}$ contient les catégories nécessaires au calcul de *p*. Dans le cas de propriétés unaires portant sur une catégorie c il suffit d'avoir $c \in \mathcal{A}$. Les propriétés binaires ont une évaluabilité différente selon qu'elles sont positives ou négatives. Les premières mettent en relation deux catégories réalisées (par exemple la nécessité de cooccurrence). Dans ce cas, si c_1 et c_2 sont ces catégories, on a $\{c_1, c_2\} \subset \mathcal{A}$. Dans le cas de propriétés binaires négatives, si c_1 et c_2 sont les catégories de p, on a soit $c_1 \in \mathcal{A}$, soit $c_2 \in \mathcal{A}$.

Lorsqu'une propriété est évaluable pour un $\mathcal{A}$ donné, sa satisfaisabilité est calculée, en fonction de sa propre sémantique. Chaque propriété est ainsi associée à un solveur, leur détail n'est pas exposé ici. Le processus global s'écrit :

```
Soit 𝒢 l'ensemble des propriétés, soit 𝒜 une affectation
  ∀p_i ∈ 𝒢
     Si p_i est évaluable
        Calculer la satisfaction de p_i pour 𝒜
        Ajouter p_i et le résultat de son évaluation à la caractérisation C de 𝒜
  Décider si C est productive
```

Ce processus signifie que pour toutes les affectations, toutes les propriétés de la grammaire doivent être parcourues pour en vérifier la satisfaction.

Le dernier volet du processus concerne l'évaluation de la caractérisation elle-même. Une caractérisation productive permet d'instancier la catégorie correspondante ou, en d'autres termes, de considérer que cette catégorie est réalisée. Il est alors possible de l'ajouter à l'ensemble des catégories $\mathcal{K}$ à prendre en considération. Une caractérisation est bien entendu productive si elle est entièrement composée de propriétés satisfaites. Mais il est également possible de la considérer comme étant productive même si elle contient des propriétés enfreintes. Ceci revient à relâcher les contraintes. Ce processus doit bien entendu être contrôlé : définition d'un seuil de contraintes à satisfaire, spécification d'une hiérarchisation des contraintes, etc. Ce processus de décision est effectué sur la base de caractérisations complètes.

La construction des affectations, le calcul de leur évaluabilité et de leur productivité constituent les trois processus fondamentaux en GP. Chacun représente un facteur de complexité que nous proposons de contrôler grâce à la corrélation.

3.6. Conclusion

La recherche d'information peut être améliorée par des techniques d'analyse linguistique fine à la condition que celles-ci soient robustes et efficaces. Pour le dernier aspect, les récentes évolutions montrent qu'il est désormais possible d'envisager des temps de traitement raisonnables. Pour ce qui concerne la robustesse, là encore, les différentes techniques aujourd'hui disponibles permettent d'envisager des solutions pour le traitement d'énoncés non canoniques, provenant par exemple de la langue parlée. Les problèmes posés dans ce cas à l' analyse syntaxique sont multiples : outre le fait qu'il s'agit dans ces cas d'analyser des phénomènes ne faisant pas partie de la grammaire initiale, il faut de plus proposer des traitements capable de construire des solutions partielles. Dans le même temps, l'erreur doit pouvoir être décrite (et non pas simplement ignorée) pour pouvoir éventuellement appliquer des stratégies de rattrapage voire de correction.

Les techniques basés sur des approches stochastiques sont d'une grande efficacité et raisonnablement robustes, y compris lorsqu'elles sont confrontées à des *inputs* non canoniques. En d'autres termes, elles sont résistantes à l'erreur, mais sans pouvoir les traiter. Il est toujours possible d'améliorer les performances et le comportement des analyseurs reposant sur ces approches, à la condition de disposer de corpus d'entraînement adaptés. Plusieurs outils de ce type sont en cours de développement pour le français. Pour autant, cette approche reste peu utilisable dans la perspective d'un traitement dynamique des erreurs. Les approches symboliques sont dans ce cas plus efficaces. L'exemple des techniques basées sur les contraintes est de ce point de vue éclairant : elles permettent, tout en proposant un cadre théorique précis, de fournir des analyses quelle que soit la forme de l'*input*. De plus, ce type d'approche permet d'intégrer directement des sources d'information hétérogènes sous la forme de contraintes décrivant des domaines ou des modalités différentes (voir [BLA 09]). Nous sommes ainsi en mesure de proposer un outil efficace et capable de prendre en compte les besoins futurs, notamment en termes de multimodalité.

3.7. Bibliographie

[ABN 96] ABNEY S., « Partial Parsing via Finite-State Calculus », *Robust Parsing Workshop, ESSLLI'96*, p. 257-278, 1996.

[BAL 05] BALFOURIER J.-M., BLACHE P., GUÉNOT M.-L., VANRULLEN T., « Comparaison de trois analyseurs symboliques pour une tâche d'annotation syntaxique », *Actes de Traitement Automatique des Langues Naturelles*, vol. 2, Dourdan, France, p. 41-48, 6-10 juin 2005.

[BAN 09] BANGALORE S., JOHNSTON M., « Robust Understanding in Multimodal Interfaces », *Computational Linguistics*, vol. 35, n°3, p. 345-397, 2009.

[BLA 01] BLACHE P., *Les Grammaires de Propriétés : Des contraintes pour le traitement automatique des langues naturelles*, Hermès, 2001.

[BLA 06] BLACHE P., RAUZY S., « Mécanismes de contrôle pour l'analyse en Grammaires de Propriétés », *Actes de Traitement Automatique des Langues Naturelles*, Leuven, Belgique, p. 415-424, 10-13 avril 2006.

[BLA 07] BLACHE P., RAUZY S., « Le moteur de prédiction de mots de la Plateforme de Communication Alternative », *Traitement Automatique des Langues*, vol. 48, n°2, p. 47-70, 2007.

[BLA 08] BLACHE P., RAUZY S., « Influence de la qualité de l'étiquetage sur le chunking : une corrélation dépendant de la taille des chunks », *Actes de Traitement Automatique des Langues Naturelles*, Avignon, France, p. 290-299, 9-13 juin 2008.

[BLA 09] BLACHE P., « Des relations d'alignement pour décrire l'interaction des domaines linguistiques : vers des Grammaires Multimodales », *Actes de la conférence TALN*, 2009.

[BOU 05] BOULLIER P., SAGOT B., « Efficient and robust LFG parsing : SxLfg », *IWPT'05*, 2005.

[BRI 06] BRISCOE E. J. C., WATSON R., « The Second Release of the RASP System », *COLING/ACL 2006 Interactive Presentation Sessions*, Sydney, Australia, 2006.

[CHA 01] CHANOD J. P., « Robust Parsing and Beyond », JUNQUA J. C., VAN NOORD G., Eds., *Robustness in Language and Speech Technology*, Kluwer, p. 187-204, 2001.

[CLE 08] DE LA CLERGERIE E., AYACHE C., DE CHALANDAR G., FRANCOPOULO G., GARDENT C., PAROUBEK P., « Large Scale Production of Syntactic Annotations for French », *Proceedings of the international workshop on Automated Syntactic Annotations for Interoperable Language Resources*, Hong-Kong, 2008.

[CRY 02] CRYSMANN B., FRANK A., KIEFER B., MÜLLER S., NEUMANN G., PISKORSKI J., SCHÄFER U., SIEGEL M., USZKOREIT H., XU F., BECKER M., KRIEGER H., « An Integrated Architecture for Shallow and Deep Processing », *Proceedings of ACL-02*, 2002.

[GEN 03] GENDNER V., ILLOUZ G., JARDINO M., MONCEAUX L., PAROUBEK P., ROBBA I., VILNAT A., « PEAS, the first instantiation of a comparative framework for evaluating parsers of French », *Research Notes of EACL 2003*, Budapest, Hongrie, Avril 2003.

[HIN 83] HINDLE D., User manual for Fidditch, a deterministic parser, Technical memorandum n°7590-142, Naval Research Laboratory, 1983.

[IDE 94] IDE N., VÉRONIS J., « MULTEXT : Multilingual Text Tools and Corpora », *Proceedings of the 15th. International Conference on Computational Linguistics (*COLING 94*)*, vol. I, Kyoto, Japan, p. 588–592, 1994.

[JOH 98] JOHNSON M., « PCFG Models of Linguistic Tree Representations », *Computational Linguistics*, vol. 24, n°4, 1998.

[KAR 90] KARLSSON F., « Constraint grammar as a framework for parsing running texts », *proceedings of ACL-90*, 1990.

[LEW 96] LEWIS D. D., JONES K. S., « Natural language processing for information retrieval », *Commun. ACM*, vol. 39, n°1, p. 92–101, 1996.

[MAR 90] MARUYAMA H., « Structural Disambiguation with Constraint Propagation », *proceedings of ACL-90*, 1990.

[MAR 02] MARIMON M., « Integrating Shallow Linguistic Processing into a Unification-Based Spanish Grammar », *proceedings of COLING-02*, 2002.

[MEN 98] MENZEL W., « Constraint satisfaction for robust parsing of spoken language », *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 10, n°1, p. 77 - 89, 1998.

[PAR 00] PAROUBEK P., RAJMAN M., « MULTITAG, une ressource linguistique produit du paradigme d'évaluation », *Actes de Traitement Automatique des Langues Naturelles*, Lausanne, Suisse, p. 297-306, 16-18 octobre 2000.

[PAR 06] PAROUBEK P., ROBBA I., VILNAT A., AYACHE C., « Data Annotations and Measures in EASY the Evaluation Campaign for Parsers in French », *Proceedings of the 5th international Conference on Language Resources and Evaluation*, Genoa, Italy, p. 314-320, May 2006.

[POL 96] POLLARD C., « The Nature of Constraint-Based Grammar », *Proceedings of Pacific Asia Conference on Language, Information, and Computation*, 1996.

[PRI 93] PRINCE A., SMOLENSKY P., *Optimality Theory : Constraint Interaction in Generative Grammars*, Technical Report RUCCS TR-2, Rutgers Center for Cognitive Science, 1993.

[RON 96] RON D., SINGER Y., TISHBY N., « The power of amnesia : Learning probabilistic automata with variable memory length », *Machine Learning*, vol. 25, p. 117-149, 1996.

[ROS 70] ROSENKRANTZ S. J., LEWIS P., « Deterministic left corner parsing », *Proceedings of the 11th Annual Symposium on Switching and Automata*, p. 139-152, 1970.

[SAG 03] SAG I., WASOW T., BENDER E., *Syntactic Theory. A Formal Introduction*, CSLI, 2003.

[STR 94] STRZALKOWSKI T., « Robust text processing in automated information retrieval », *Proceedings of the fourth conference on Applied natural language processing*, p. 168-173, 1994.

[TJO 00] TJONGKIMSANG E., BUCHHOLZ S., « Introduction do the CoNLL-2000 Shared Task : Chunking », *proceedings of CoNLL-2000*, 2000.

[USZ 02] USZKOREIT H., « New Chances for Deep Linguistic Processing », *proceedings of COLING-02*, 2002.

[VAN 05] VANRULLEN T., BLACHE P., PORTES C., RAUZY S., MAEYHIEUX J.-F., GUÉNOT M.-L., BALFOURIER J.-M., BELLENGIER E., « Une plateforme pour l'acquisition, la maintenance et la validation de ressources lexicales », *Actes de Traitement Automatique des Langues Naturelles*, vol. 1, Dourdan, France, p. 511-516, 6-10 juin 2005.