Correction automatique en temps réel
Contraintes, méthodes et voies de recherche

Roger Rainero

Société Diagonal
1, Traverse des Brucs – Valbonne — Sophia Antipolis
roger.r@prolexis.com

Mots-clés :

Correction automatique, temps réel, analyse syntaxique, grammaire de contraintes.

Résumé

Cet article expose un cas concret d'utilisation d'une grammaire de contraintes. Le produit qui les applique a été commercialisé en 2003 pour corriger automatiquement et en temps réel les fautes d'accord présentes dans les sous-titres des retransmissions en direct des débats du Sénat du Canada. Avant la mise en place du système, le taux moyen de fautes était de l'ordre de 7 pour 100 mots. Depuis la mise en service, le taux d'erreurs a chuté à 1,7 %.

Nous expliquons dans ce qui suit les principaux atouts des grammaires de contraintes dans le cas particulier des traitements temps réel, et plus généralement pour toutes les applications qui nécessitent une analyse au fur et à mesure du discours (c.-à-d. sans attendre la fin des phrases).

Keywords:

Automatic correction, real-time, syntactic analysis, grammar of constraints.

Abstract

This article sets out a concrete use case of a grammar of constraints. The product which applies them was commercialised in 2003 to automatically correct in real time the errors of agreement present in the sub-titles of live televised debates from the Senate of Canada.

Before the introduction of this system, the average rate of mistakes was in the order of 7 per 100 words. With the introduction of this system, the rate of errors has fallen to 1.7%.

In the following section, we explain the main advantages of a grammar of constraints in the specific case of real-time processing, and more generally for all applications which require an analysis during the speech (that is, without waiting until the end of sentences).

1. Exposé du problème

Le Sénat du Canada diffuse certains de ses débats en direct sur une chaîne de télévision spécialisée. Chaque sénateur pouvant s'exprimer dans sa langue maternelle (français ou anglais), les interventions se succèdent indifféremment dans ces deux langues. Les téléspectateurs ont la possibilité d'afficher des sous-titres, soit en français, soit en anglais, mais lorsqu'une langue est choisie, la totalité des débats est transcrite dans cette langue (fonction légale pour les malentendants).

Les sous-titrages français sont obtenus par retranscription sténotypée soit directe (locuteur français) soit indirecte (locuteur anglais traduit simultanément en français, la sténotypiste enregistrant alors la traduction). Les sténotypistes francophones et anglophones utilisent la même méthode de saisie mise au point en Amérique du Nord, très performante pour les langues globalement phonétiques (où la majorité des lettres se prononcent). Cette méthode donne ainsi d'excellents résultats en anglais. Mais pour le français qui comporte de nombreuses syllabes finales muettes, les ajustements ont été longs et fastidieux, et la mise en ondes a été maintes fois repoussée, à la recherche d'un taux acceptable de transcription exacte. Les résultats ont régulièrement progressé jusqu'en 2002, où ce taux a plafonné aux alentours de 93 %. (sur 100 mots, seuls 93 étaient corrects).

Bien que ce taux paraisse très élevé, il génère un nombre d'incidents de lecture très au-delà de ce qui est acceptable. Il suffit, pour s'en convaincre, de constater qu'il correspond à 8 fautes par minute de lecture. Le Sénat du Canada a alors fait un appel d'offres international dans le but de trouver une solution automatique susceptible d'améliorer cette situation. La solution proposée devait permettre de corriger automatiquement le plus grand nombre de fautes résiduelles possibles, sans ajouter de fautes là où il n'y en a pas. Par ailleurs, l'automate devait s'intercaler dans le processus d'acquisition du texte sténotypé (logiciel Eclipse déjà installé) sans le ralentir de façon notable.

La société Diagonal a soumissionné en proposant une adaptation spécifique de ses moteurs d'analyse déjà utilisés dans les logiciels de correction ProLexis et Myriade. Cette solution retenue par le Sénat a été livrée en octobre 2003.

2. Exigences dynamiques de la correction automatique des sous-titrages en temps réel

Le système demandé par le Sénat imposait de faire les corrections au fur et à mesure de la saisie, c'est-à-dire sans que l'on puisse attendre la fin des phrases. Cette obligation vient essentiellement du direct : les sous-titres suivent à peu près les paroles des orateurs. En théorie donc, les corrections doivent être faites quasi immédiatement après les fautes.

En pratique, nous disposions des souplesses suivantes :

- les diffusions sont en léger différé d'une à deux secondes,
- les sous-titres sont découpés en lignes qui ne partent à l'antenne que lorsqu'elles sont pleines.

Exemple avec la phrase : *« Les filles jouent aux billes, les garçons jouent au ballon. »*

Voici ce que saisit la sténotypiste par tranche de 0,5 seconde (avec les fautes) :

0,5 s	*Les*
1,0 s	*Les fille*
1,5 s	*Les fille joue*
2,0 s	*Les fille joue au*
2,5 s	*Les fille joue au bille,*

3,0 s *Les fille joue au bille, les*
3,5 s *Les fille joue au bille, les garçon*
4,0 s *Les fille joue au bille, les garçon joue*
4,5 s *Les fille joue au bille, les garçon joue au*
5,0 s *Les fille joue au bille, les garçon joue au ballon.*

Et voici ce que doit voir le téléspectateur (entre parenthèses, les corrections à faire) :

0 s (rien)
3 s LES FILLE(S) JOUE(NT) AU(X) BILLE(S),
6 s LES GARÇON(S) JOUE(NT) AU BALLON.

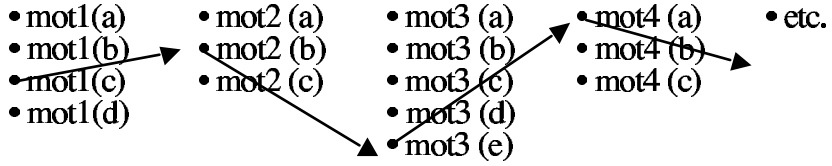
L'automate doit corriger les fautes de la 1re ligne au plus tard au temps 5,0 (temps réel 3,0 + 2 secondes de différé), c'est-à-dire lorsque le 4e mot de la ligne suivante vient d'arriver. Pour la première faute (fille), il dispose d'un retard de 7 mots, mais pour la dernière faute (bille), il ne bénéficie plus que d'un retard de 4 mots. L'automate ignorant totalement à quel moment les lignes sont déclarées « pleines », il est obligé de s'astreindre à faire toutes ses corrections avec un maximum de quatre mots de retard !

C'était bien là la plus grande difficulté à laquelle nous allions être confrontés.

3. Les atouts des systèmes d'analyse basés sur les contraintes

Ce qui a rendu la chose possible avec ProLexis tient dans le fait essentiel que notre moteur exploite un principe de propagation de contraintes.

Comme dans les principaux systèmes basés sur la satisfaction de contraintes (cf. (Blache00)), notre approche ne vise pas à construire un arbre syntaxique de la phrase, mais plutôt à optimiser, dans un réseau de contraintes, le chemin menant du premier au dernier mot de la phrase :



Sur ce schéma, les contraintes ne sont pas figurées. Elles ne se manifestent qu'au travers du choix du chemin affiché qui est censé satisfaire le maximum d'entre elles.

Chaque nouveau mot apporte son lot de variantes possibles, mais aussi son lot de contraintes potentielles pour toutes les variantes établies depuis le début de la phrase :

→ Appliquer une contrainte revient à calculer son coefficient de satisfaction (son « poids ») dans toutes les variantes établies.

→ Propager une contrainte revient à reconsidérer l'application des contraintes déjà appliquées, comme conséquence de l'arrivée du nouveau jeu de contraintes.

Notre système ne construit donc pas un arbre, mais il pondère un réseau. À la fin de la phrase, un algorithme simple peut restituer l'arborescence de la structure syntaxique, si le besoin s'en fait sentir, mais cela n'est pas nécessaire pour diagnostiquer les erreurs et les corriger.

La pondération est souple, dans la mesure où une contrainte mal satisfaite n'est qu'une indication d'un écart par rapport à la norme formalisée par cette contrainte. En ce sens, elle produit des analyses robustes qui s'accommodent de déviances parfois fortes par rapport aux usages ou même de l'absence de certains mots.

Tout cela est bien connu et caractérise les grammaires basées sur les contraintes, mais n'est pas déterminant dans le cas qui nous intéresse.

Le plus gros avantage de notre grammaire, dans le cadre de la correction automatique en temps réel, provient de sa capacité à délivrer une analyse pondérée des variantes après chaque mot. Bien sûr, l'analyse est réputée optimale lorsque tous les mots de la phrase sont connus, mais cette analyse est néanmoins disponible en phase intermédiaire après chaque mot.

Enfin, le mécanisme de propagation après chaque mot présente un autre avantage de taille dans le cas de notre application « temps réel » : il permet, par un choix judicieux des coefficients de pondération, de marginaliser assez vite certaines variantes isolées et de limiter à un niveau raisonnable le nombre de variantes concurrentes que le logiciel évalue en parallèle à chaque itération.

Ce mécanisme peut même s'autoréguler en détruisant systématiquement les variantes les moins probables à chaque passe, de telle sorte que leur nombre total reste en dessous d'un seuil critique pour le temps d'exécution.

Évidemment, toutes ces actions aveugles sont préjudiciables à la qualité de l'analyse *in fine*. Toute la question était de quantifier leur influence réelle sur la fiabilité des corrections attendues.

4. Tests de fiabilité des résultats intermédiaires à « mot + 4 »

Jusqu'à présent, nous n'avions jamais testé la fiabilité des résultats intermédiaires avant la fin de la phrase. Il nous fallait donc vérifier que dans le contexte du Sénat, nous disposions effectivement de suffisamment d'indices pour décider des corrections au maximum à mot + 4.

En théorie, en effet, tout nouveau mot dans une phrase peut changer totalement son analyse. C'est un exercice bien connu auquel se livrent volontiers les professionnels de l'analyse syntaxique. Et c'est aussi avec de tels exemples que l'on peut démontrer que ce que nous avons fait est impossible. En voici quelques-uns :

Début : *Le chien regarde le chat et la souris…*
Suite 1 : *Le chien regarde le chat et la souris mais…*
Suite 2 : *Le chien regarde le chat et la souris prise…*
Suite 2a : *Le chien regarde le chat et la souris prise… (au piège ?)*
Suite 2b : *Le chien regarde le chat et la souris prise… (du tabac ?)*

Mais quelle est la portée statistique réelle d'un tel phénomène et quelle est son influence sur la fiabilité d'un système de correction automatique après quatre mots ?

Pour l'estimer, nous avons fabriqué un prototype du produit fini simulant le comportement de l'outil d'acquisition Eclipse. Ce logiciel lisait un extrait des débats du Sénat obtenu par sténotypie et l'envoyait signe à signe à l'automate qui gardait trace dans un fichier de sortie de toutes les corrections faites et du moment où elles pouvaient être faites.

Le tableau suivant montre un extrait de ce fichier de sortie, concernant un début de phrase telle qu'elle est délivrée par Eclipse, fautes comprises (colonne de gauche). À droite, les fautes corrigées sont intercalées après le mot qui les rend possibles :

Après la saisie de…	Les corrections suivantes sont faites…
Il faut également des solution pratique qui…	solution ➔ solutions pratique ➔ pratiques
soit sensé…	soit ➔ soient
pour…	sensé ➔ sensées
ceux qui travaille sur…	travaille ➔ travaillent

On constate que les deux premières erreurs « solution » et « pratique » sont corrigées dès la saisie de « qui », donc respectivement à mot + 2 et mot + 1.

Regardons de plus près les analyses qui sont faites à ce stade : le mot « pratique » est ambigu : ce peut être un nom, un adjectif ou un verbe. Toutes ces formes génèrent potentiellement autant de variantes. Les variantes verbales paraissent improbables, mais, comme toujours en pareil cas, une petite réflexion permet d'en découvrir certaines formes légitimes :

« Il faut également des solutions, pratique ce sport et tu verras ! »
« Il faut également des solutions(,) explique César… »

Bien sûr, la virgule semble cruciale, mais la pondération de son absence n'est pas suffisante ici. En revanche, l'arrivée du mot suivant « qui » est déterminante, elle rend la flexion verbale pour le mot « pratique » quasiment impossible. En théorie, la flexion verbale ne peut être totalement exclue, car l'hypothèse d'oubli d'un mot peut toujours la justifier. Mais les réglages actuels de nos seuils de probabilités pour des textes de provenance sténotypée font qu'elle est rejetée ici.

La suite est compréhensible : déterminé nominal, le groupe précédent est fautif sur l'accord GN. Deux formes correctes sont possibles : « une solution pratique » et « des solutions pratiques ». C'est là qu'interviennent des automates spécialisés dans la correction automatique spécifique du Sénat du Canada : ces automates choisissent de façon probabiliste la correction au pluriel.

Appliqué au texte de référence de 2 000 mots fourni par le Sénat, contenant 149 fautes et correspondant à un débat réel de 20 minutes, le prototype a donné les résultats suivants :

Nombre de fautes corrigées :	…avec un retard de :
89	1 mot
17	2 mots
3	3 mots
1	4 mots

La faible incidence des situations ambiguës sur les corrections automatiques envisagées pour les débats du Sénat du Canada paraissait donc confirmée, au moins sur le texte étudié. Et la propagation de contraintes montrait là une capacité tout à fait étonnante à résoudre le problème posé. Restaient à démontrer son efficacité et sa stabilité à grande échelle.

5. Méthode d'évaluation à grande échelle.

L'inconvénient des systèmes probabilistes est que leur comportement ne peut être totalement déduit de tests à petite échelle. Typiquement, dans le cas du Sénat, la nature même des débats influe grandement sur le vocabulaire, les intervenants et donc sur les types de phrases prononcées. Nul doute qu'un simple échantillon de 2 000 mots ne pouvait représenter correctement la totalité des situations auxquelles devrait faire face l'automate après sa mise en service.

Après avoir été choisis par le Sénat du Canada pour exécuter le marché, nous avons donc lancé en parallèle les deux réalisations suivantes : d'une part, le logiciel lui-même (bien entendu), et d'autre part, l'étalonnage d'un corpus de 50 000 mots destiné à valider les tests d'usine du logiciel, avant sa livraison chez le client.

Ce corpus a été extrait des transcriptions sténotypées de débats récents représentant un peu plus de 10 heures d'antenne réparties sur une période de deux mois. On ne s'est limité à cette taille que pour des contraintes de temps. Deux personnes ont travaillé pendant un mois pour sélectionner les textes, éliminer les passages en double, détecter les fautes et les baliser dans le texte. Quelque 3 500 fautes y ont été repérées.

L'application de l'automate sur ce corpus a corrigé plus de 2 500 fautes sur 3 500, établissant un taux de reconnaissance à grande échelle stable à 98,31 %. Sur ce même corpus, l'automate n'a introduit que 23 fautes, soit un taux moyen de surcorrection de 1/2100.

6. Perspectives et voies de recherche

Il faut se méfier de l'idée fausse qui consiste à penser que le temps d'exécution n'est pas un problème majeur pour les algorithmes d'analyse automatique. On entend souvent dire : « de toute manière, les machines iront de plus en plus vite et un jour viendra où les algorithmes lents s'exécuteront vite ! ».

Tout cela est vrai, sauf pour les applications « temps réel ». La correction automatique des sous-titrages est un exemple, mais il y en a bien d'autres. La reconnaissance vocale multilocuteur et la traduction simultanée sont deux domaines qui pointent déjà à l'horizon et qui n'attendent que l'émergence de nouvelles technologies linguistiques pour se déployer à grande échelle.

Le découpage du mécanisme d'analyse en strates successives ou le contrôle intégré du nombre de variantes concurrentes que permettent aujourd'hui les grammaires guidées par la satisfaction de contraintes semble être un atout de poids dans les applications temps réels.

Références

Blache P. (2000) Le rôle des contraintes dans les théories linguistiques et leur intérêt pour l'analyse automatique, Actes de *TALN 2000*.

Blache P. (2000) Constraints, Linguistic Theories and Natural Language Processing, *Lecture Notes in Artificial Intelligence,* Springer-Verlag.

Harper M. P., Helzerman R. A. (1995). Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, Vol. 9 (3), pp 187-234.

Maruyama, H. (1990). Constraint dependency grammar and its weak generative capacity. *Computer Software*.