

Une stratégie de contrôle pour l'analyse syntaxique

Philippe Blache

LPL - CNRS
29, Avenue Robert Schuman
F-13621 Aix-en-Provence
pb@lpl.univ-aix.fr

Résumé

Les techniques de pré-analyse ne permettent généralement pas de fournir des informations extrêmement précises. Or, un contrôle efficace de l'analyse syntaxique, en particulier pour traiter la question de l'ambiguïté, doit nécessairement faire appel à des contraintes suffisamment évoluées, par exemple pour contrôler efficacement le retardement d'évaluation. Nous proposons dans cet article une méthode construisant à partir d'une phrase des contraintes syntaxiques de haut niveau sans faire appel à la grammaire ni, par voie de conséquence, à des mécanismes d'analyse proprement dits. Elle est donc efficace et générale puisqu'indépendante des formalismes syntaxiques.

1. Introduction

L'analyse syntaxique pose un certain nombre de problèmes, et notamment celui de l'ambiguïté, pouvant nécessiter une phase de pré-traitement. Le problème vient du fait que les techniques utilisées ne permettent pas de fournir des informations très détaillées sauf à anticiper l'analyse syntaxique. Nous proposons dans cet article une approche fournissant des contraintes syntaxiques de haut niveau, permettant à la fois de contrôler le processus de désambiguïsation et de servir de base à l'analyse syntaxique. Cette technique repose exclusivement sur des informations contenues au niveau lexical, et permet d'élaborer ces contraintes sans faire appel à une analyse syntaxique. Il s'agit d'un procédé de contrôle efficace tant par la qualité des informations fournies que par son faible coût d'implantation.

Nous présentons dans la première partie les données de base disponibles au niveau lexical et que nous appelons *quasi-arbres unaires*. Ces structures permettent de représenter directement des contraintes de dépendance au niveau syntaxique en s'appuyant sur les

projections possibles des catégories analysées. La seconde partie présente la construction du réseau de preuve servant de base au calcul des contraintes de dépendance. Dans la dernière partie, nous décrivons la technique de simplification permettant d'extraire le sous-ensemble de contraintes minimal.

2. Les Quasi-Arbres Unaires

Les informations de sous-catégorisation d'une catégorie sont toujours accessibles dès le niveau lexical. Les objets sous-catégorisés peuvent être de niveau lexical ou syntagmatique et dans ce cas, une analyse classique doit attendre de construire la structure syntagmatique avant de pouvoir établir une quelconque relation.

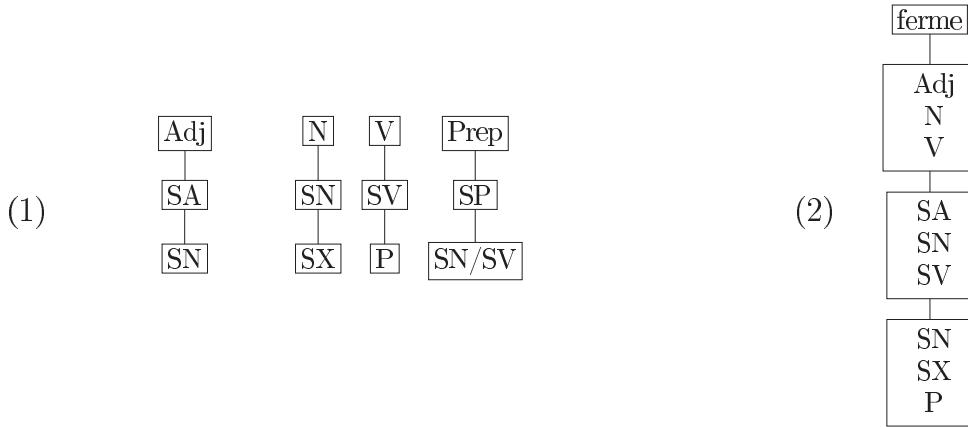
Nous proposons de régler ce problème en représentant dès le niveau lexical les projections possibles. Nous utilisons pour cela des *quasi-arbres unaires* (notés dorénavant QAU) décrits dans (Blache98). L'idée de base consiste, comme (Vijay-Shanker92) le propose dans le cadre des TAGs, à construire le quasi-arbre qui sera utilisée comme contrainte sur l'arbre syntaxique lui-même. Dans notre approche, nous réduisons la généralité des quasi-arbres comme suit:

- Une relation entre deux noeuds α et β (α dominant β dans le QAU) indique qu'il existe une dérivation de la forme $\alpha \Rightarrow^* B$ telle que $\beta \in B$. (en d'autres termes, le constituant β appartient à un ensemble de constituants B dérivés à partir de α)
- Chaque noeud a un seul descendant (l'arbre est réduit à une liste).
- Chaque noeud peut être remplacé par un sous-arbre de même type.
- Les noeuds peuvent contenir plusieurs objets. On représente ainsi l'ambiguïté.
- Chaque noeud est une formule disjonctive.
- Un QAU est limité à trois niveaux : lexical, syntagmatique et propositionnel.

Bien entendu, seules les catégories ayant une projection (les catégories majeures) seront associées à un QAU. Nous obtenons ainsi un ensemble de quatre QAU élémentaires (pour des raisons de cohérence avec les exemples de la section suivante, ces arbres sont représentés la tête en bas) :

Dans l'exemple (1), le QAU associé à l'adjectif indique que celui-ci appartiendra dans la structure finale à un sous-arbre dont la racine est un syntagme adjectival qui lui-même appartiendra à un sous-arbre dont la racine est un syntagme nominal.

On construit les QAU par concaténation en fonction des catégorisations possibles d'un mot : si un mot possède trois catégorisations, chaque noeud du QAU sera composé d'un ensemble de trois valeurs. C'est le cas de l'exemple (2) dans lequel *ferme* peut être adjectif, nom ou verbe.



L'ensemble des valeurs possibles est bien entendu de même cardinalité à chaque niveau : il y a *covariation* de ces valeurs ou, en d'autres termes, si une valeur est sélectionnée à un niveau, les valeurs qui lui sont reliées aux autres niveaux doivent également être sélectionnées. Cette relation peut être implantée à l'aide de *disjonctions contrôlées* (cf. (Blache97)).

3. Les Réseaux de Preuve

La spécification de contraintes repose sur un mécanisme en deux étapes. La première phase consiste à établir l'ensemble des relations possibles sur la base des informations de sous-catégorisation. Ces relations forment un graphe également appelé *réseau*. La seconde étape consiste à simplifier ce graphe pour en extraire les relations pertinentes.

Nous décrivons dans cette section les données de base utilisées pour la construction du réseau. Il s'agit en particulier de représenter l'ensemble des liens possibles et d'identifier les noeuds de ce graphe qui pourront être connectés.

3.1. *Les relations potentielles*

La notion de réseau utilisée pour représenter les relations de dépendances est notamment présente dans les approches s'appuyant sur la logique linéaire (voir par exemple (Lecomte92), (Retoré96), (Johnson97a) ou (Johnson97b)). Dans notre cas, nous utiliserons le même terme de réseau pour désigner l'ensemble des *relations potentielles* ainsi que le résultat obtenu par simplification de cette structure préliminaire. Cette notion se rapproche de celle de pré-réseau (cf. (Retoré96)) dans le sens où il s'agit d'une étape préliminaire à l'analyse.

La première étape de construction du réseau consiste à spécifier les relations unissant les objets linguistiques. Nous représentons ainsi une phrase comme une liste de noeuds identifiés par un entier correspondant à la position du mot dans la phrase (cf. figure (1)). Nous nous limiterons ici aux sous-catégorisations mettant en relation des catégories de

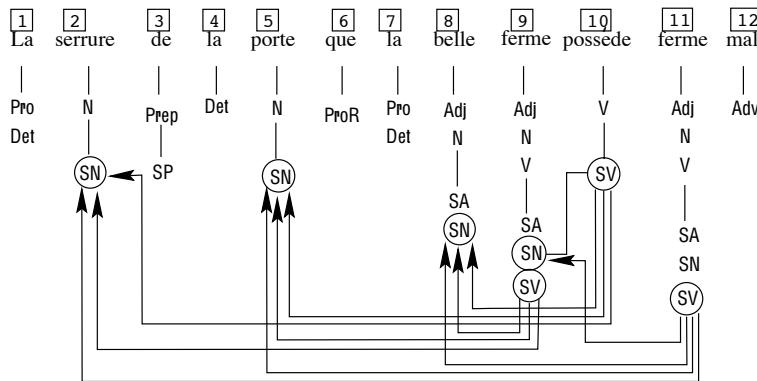


FIG. 1 – Réseau de contraintes

niveau syntagmatique (i.e. le second niveau des QAU). Nous représentons séparément (pour des raisons de clarté) les relations orientées vers la droite de celles orientées vers la gauche. L'exemple de la figure (1) représente ainsi l'ensemble des relations potentielles à gauche.

3.2. Tableau de ressources

Les noeuds des QAU peuvent être, dans la terminologie des réseaux de preuve, des objets *consommateurs de ressources* ou à l'opposé constituer eux-même des *ressources* à consommer. Dans ce dernier cas, il est intéressant de stocker cette information de façon à faciliter la détection de liens.

Un QAU est référencé par la position du mot dans la phrase (i.e. un entier). De plus, en cas d'ambiguïté, chaque valeur d'un noeud est à son tour référencée par une position. Dans l'exemple (2), la valeur *V* occupe la troisième position du noeud. On fera par la suite référence au *nœud* pour indiquer l'indice du mot dans la phrase et à la *valeur* pour indiquer l'indice de son interprétation dans la disjonction de valeurs possibles.

D'une façon générale, les valeurs sont référencées par un doublet $\langle i, j \rangle$ dans lequel i représente la position du mot correspondant dans la phrase et j le rang de la valeur dans le noeud du QAU.

L'ensemble des ressources syntagmatiques de ce réseau est stocké dans un *tableau de ressources* (noté dorénavant T_{Res}). Celui-ci indique toutes les valeurs pouvant constituer une ressource consommable par type de niveau syntagmatique. L'exemple (3) présente le tableau de ressources de la phrase (1).

| | |
|------|---|
| SA | $\langle 8, 1 \rangle, \langle 9, 1 \rangle, \langle 11, 1 \rangle$ |
| SN | $\langle 2, 1 \rangle, \langle 5, 1 \rangle, \langle 8, 2 \rangle, \langle 9, 2 \rangle, \langle 11, 2 \rangle$ |
| SV | $\langle 9, 3 \rangle, \langle 10, 3 \rangle, \langle 11, 3 \rangle$ |
| SP | $\langle 3, 1 \rangle$ |

3.3. Sous-ensembles de valence

La construction du réseau de preuve repose sur l'exploitation des informations contenues dans le tableau de ressources ainsi que sur la connaissance de la position relative de l'objet sous-catégorisé par rapport à la tête (i.e. avant ou après la tête). Cette information peut être accessible plus ou moins directement en fonction des formalismes utilisés. Les informations de sous-catégorisation seront donc divisées en deux sous-ensembles:

- V_D = le sous-ensemble de valence à droite de la tête
- V_G = le sous-ensemble de valence à gauche.
- On note $V_D(\alpha)$ (resp. $V_G(\alpha)$) la fonction qui retourne le sous-ensemble de valence à droite (resp. à gauche) de α .

4. Représentation par tableau

Nous distinguerons dans la représentation l'ensemble des nœuds sources (comportant au moins un arc sortant) de l'ensemble des nœuds cibles (comportant au moins un arc entrant). Chaque valeur de nœud est représentée par un doublet $\langle i, j \rangle$ (cf. section 3.2). Une relation devra pouvoir être lue au niveau des nœuds (niveau général), mais également au niveau des valeurs elles-mêmes, offrant ainsi la possibilité de points de vues différents.

Nous avons choisi de représenter les relations du réseau à l'aide d'un tableau, noté T_{Preuve} , dont les lignes représentent les indices des nœuds sources tandis que les colonnes représentent les indices des nœuds cibles.

Soient deux nœuds α et β référencés respectivement par $\langle i, s \rangle$ et $\langle j, t \rangle$ où i et j représentent les positions, s et t représentent les valeurs.

- Une relation entre α et β est représentée par la cellule $T_{Preuve}[i, j]$,
- Le contenu de $T_{Preuve}[i, j] = [s, t]$

En d'autres termes, une relation entre deux nœuds se représente en indiquant dans la cellule correspondante les valeurs de ces nœuds. On peut se contenter de savoir s'il existe ou non une relation entre des nœuds (point de vue général) ou au contraire *zoomer* sur cette case et voir quelles sont les valeurs des nœuds utilisées.

Le tableau de la figure (2) décrit le réseau de l'exemple (1). La cellule 3/9 contient le doublet [1, 2] qui indique une relation entre la première valeur de 3 (le SP) et la deuxième valeur de 9 (le SN).

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|----|---|-------|---|-------|---|---|-------|-------|-------|-------|----|----|
| 1 | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |
| 3 | | | | [1,1] | | | [1,2] | [1,2] | | [1,2] | | |
| 4 | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | |
| 6 | | | | | | | | [1,3] | [1,1] | [1,3] | | |
| 7 | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | |
| 9 | | [3,1] | | [3,1] | | | [3,2] | | | | | |
| 10 | | [1,1] | | [1,1] | | | [1,2] | [1,2] | | | | |
| 11 | | [3,1] | | [3,1] | | | [3,2] | [3,2] | | | | |
| 12 | | | | | | | | | | | | |

FIG. 2 – Réseau de contraintes sous forme matricielle

5. Construction du réseau

Le mécanisme de construction du réseau des relations potentielles repose sur l'exploitation des informations de valence et d'un tableau de ressource construit comme indiqué dans la section 3.2. Plus précisément, pour chaque valeur des nœuds de second niveau des QAU formant l'entrée, on vérifie les valences à gauche (resp. à droite). Dans le cas d'une valence non vide, on connecte la valeur analysée avec toutes les ressources possibles situées avant ce nœud (resp. après). Les ressources possibles sont toutes les valeurs spécifiées dans le tableau des ressources et dont la ligne correspond au type spécifié par la valence.

Prenons l'exemple (1) dont le tableau de ressources est décrit en (3). La valence à gauche de la valeur 3 du QAU 9 décrivant le mot *ferme* (i.e. la valeur *SV*) indique une valeur *SN* (i.e. le sujet du verbe). La ligne *SN* du tableau T_{Res} contient les ressources possibles de ce type. Les ressources valides sont donc toutes les valeurs de cette ligne correspondant à un nœud d'indice inférieur à 9¹. Ces ressources déterminées, on ajoutera alors à toutes les cellules correspondantes de la ligne 9 du tableau T_{Preuve} les doublets $\langle 3, i \rangle$ où i représente le rang de la valeur du nœud cible.

1. Réciproquement, pour les valences à droite, les indices valides sont ceux supérieurs à la position analysée

6. Filtrage du réseau

Il convient de limiter au maximum les *relations potentielles* exprimées dans le réseau tout en essayant de tirer parti d'éventuelles relations de dépendances pouvant exister entre les valeurs. Nous examinerons dans cette section une méthode permettant d'éliminer un grand nombre de liens considérés comme non valides, en d'autres termes de filtrer le réseau.

6.1. Règles de filtrage

Deux règles de filtrage sont proposées. Elles reposent sur des propriétés généralement satisfaites dans la plupart des langues. La première règle indique que deux unités syntaxiques ne peuvent se croiser. La seconde stipule que toutes les ressources doivent être utilisées une fois et une seule².

Règle R1 : Il n'existe pas de relations croisées

Règle R2 : Une ressource est consommée une fois et une seule

L'application de ces règles repose donc sur des propriétés purement topologiques et ne fait pas appel à une grammaire. Sans être des principes universels, ces règles doivent cependant avoir une motivation linguistique. On peut imaginer compléter par des principes plus spécifiques cet outillage de filtrage du réseau. Dans tous les cas, il est important d'insister sur le fait que cette démarche ne s'appuie pas sur une grammaire, ce qui explique notamment sa facilité d'adaptation à différents formalismes linguistiques.

6.2. Processus de filtrage

On peut récapituler le mécanisme de filtrage décrit empiriquement dans la section précédente comme suit :

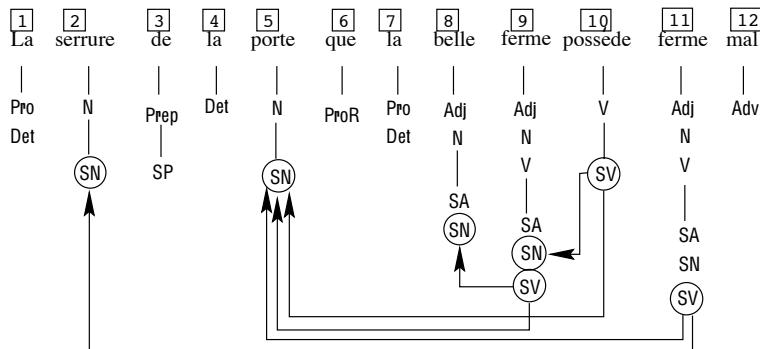
Mécanisme : soit un lien i/j , soit un noeud α .

- si il existe un lien i/k avec $k \neq j$ et
- si tous les liens reliant α entraînent une violation de R1 ou R2 par rapport à i/j ,
- alors on supprime le lien i/j .

On appelle α un *noeud bloquant* pour la relation i/j .

L'application de ce mécanisme concerne les prédictions à gauche, les prédictions à droite, mais également la propagation des informations obtenues par ces filtrages. En d'autres

2. Il existe des langues comme le suisse allemand ou des constructions comme les infinitives qui ne respectent pas l'une ou l'autre de ces règles. Mais elles sont satisfaites dans le cas général et dans tous les cas n'empêchent pas une analyse correcte pendant la phase de parsing elle-même.

FIG. 3 – *Rélations à gauche simplifiées*

termes, la simplification des prédictions à gauche peut par exemple permettre de désambiguïser une catégorie. Dans ce cas, il faut propager cette information sur les prédictions à droite de façon à restreindre encore les solutions possibles.

Prenons l'exemple (1) et analysons les effets de l'application de ce mécanisme de filtrage³. Le tableau suivant récapitule l'ensemble des prédictions à droite et à gauche constituant le réseau de relations potentielles. Nous indiquons dans la dernière colonne les relations supprimées et entre parenthèses l'indice du nœud bloquant entraînant la suppression de la relation.

| Type | Predictions | Suppressions |
|----------------------|---|---|
| Prédictions à droite | 3/5, 3/8, 3/9, 3/11, 6/9, 6/10, 6/11 | 3/8 (6), 3/9 (6) |
| Prédictions à gauche | 11/9, 11/8, 11/5, 11/2, 10/9, 10/8, 10/5, 10/2, 9/8, 9/5, 9/2 | 11/9 (10), 11/8 (9) 10/8 (9), 10/2(11) 9/2 (11) |

L'application de ce mécanisme de filtrage du réseau de relations potentielles permet ainsi d'obtenir le réseau simplifié (pour les relations à gauche) décrits dans l'exemple (3).

La propagation de ces informations sur l'ensemble du réseau permet d'obtenir un autre ensemble de simplifications. En effet, ces exemples nous indiquent qu'en tenant compte du type de la relation, nous pouvons en déduire d'autres informations. Il s'agit en d'autres termes de modifier le point de vue des relations en *zoomant* sur leurs types.

Dans l'exemple que nous étudions, nous pouvons en effet constater que la seule façon de consommer la ressource constituée par le nœud 2 est le lien 11/2. On peut donc dorénavant

3. Nous parlerons également de simplification pour faire référence au filtrage.

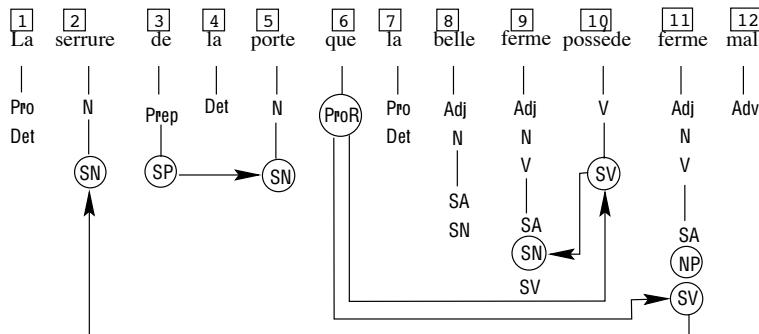


FIG. 4 – Réseau simplifié

considérer ce lien comme impératif et supprimer par voie de conséquence le lien 11/5. Mais un zoom sur le nœud 11 indique qu'il doit alors être de type *SV*. Cette nouvelle information nous permet donc de supprimer le lien 3/11 car celui-ci n'est valide que si 11 est de type *SN*. Seul reste alors le nœud 3/5 qui, comme précédemment, devient impératif. En vertu de R2, une même ressource ne pouvant être consommée qu'une fois, toutes les autres relations entrantes sur le nœud 5 sont donc supprimées à savoir les liens 9/5, 10/5 et 11/5. Par voie de conséquence, la relation 10/9 devient impérative ce qui permet de désambiguïser 9 au type *SN* et donc de supprimer les relations 6/9 et 9/8.

On peut récapituler cette ensemble de suppressions dans le tableau suivant. Toutes les actions y sont numérotées de façon, comme dans un processus de résolution, à pouvoir inférer de nouvelles conséquences.

| <i>Numéro</i> | <i>Action</i> |
|---------------|--|
| (1) | R2 \Rightarrow 11/2 est impérative |
| (2) | (1) \Rightarrow suppression de 3/11 |
| (3) | (2) + R2 \Rightarrow 3/5 est impérative |
| (4) | (3) \Rightarrow suppression de 9/5, 10/5 et 11/5 |
| (5) | (4) \Rightarrow 10/9 est impérative |
| (6) | (5) \Rightarrow 9 est de type <i>SN</i> |
| (7) | (6) \Rightarrow suppression de 6/9 et 9/8 |

Nous obtenons finalement le réseau simplifié décrit dans (4). On constate ainsi aisément que ce réseau a éliminé un très grand nombre de solutions considérées comme non valides.

7. Conclusion

L'efficacité de la technique décrite ici vient du fait qu'elle repose uniquement sur des informations de niveau lexical (la sous-catégorisation). Les autres niveaux d'information (les QAU) sont reconstitués en fonction de la catégorie analysée et sont dans tous les cas

généraux et systématiques.

Le second intérêt de cette approche réside dans le fait que le mécanisme de filtrage repose sur des propriétés géométriques et ne fait pas appel à de connaissances relevant de la grammaire. Les règles de filtrage ont bien entendu une motivation linguistique mais restent à un niveau très général.

Finalement, cette méthode de prédiction de contraintes syntaxiques est utilisable quelle que soit le formalisme linguistique utilisé dans la mesure où l'entrée repose sur des données simples et toujours disponibles.

La méthode de prédiction et de filtrage décrite ici a été originellement conçue en tant que technique de pré-analyse permettant de guider un analyseur en cas d'ambiguïté. Il est possible d'imaginer un raffinement de la technique de filtrage elle-même notamment en ajoutant des contraintes ou des règles de filtrage plus spécifiques. Ces règles peuvent être également de niveau syntaxique, adaptées par exemple à des tournures particulières, mais également d'un autre niveau comme par exemple la prosodie. Dans ce cas, il est en effet possible de fournir des indications notamment en termes de frontières d'unités.

Références

- Philippe Blache. 1997. "Disambiguating with Controlled Disjunctions", in *Proceedings of the International Workshop on Parsing Technologies*, Boston, USA.
- Philippe Blache. 1998. "Parsing Ambiguous Structures using Controlled Disjunctions and Unary Quasi-Trees", in proceedings of *ACL-COLING'98*.
- Mark Johnson. 1997. "Proof-Nets and the Complexity of Processing Center-Embedded Constructions", in proceedings of *LACL'95*.
- Mark Johnson. 1997. "Feature as Resources in R-LFG", in proceedings of *LFG'97*.
- Alain Lecomte. 1992. "Proof-Nets and Dependencies", in proceedings of *COLING'92*.
- Christian Retoré. 1996. "Calcul de Lambek et Logique Linéaire", in revue *T.A.L.*, 37:2.
- Christian Retoré (ed). 1997. *Logical Aspects of Computational Linguistics (Selected papers from LACL'96)*, Springer.
- Klaas Sikkel. 1997. *Parsing Schemata*, Springer.
- Gertjan van Noord. 1997 "An Efficient Implementation of the Head-Corner Parser", in *Computational Linguistics*, 23:3.
- K. Vijay-Shanker. 1992 "Using Descriptions of Trees in a Tree Adjoining Grammar", in *Computational Linguistics*, 18:4.