TALN 2001, Tours, 2-5 juillet 2001

Aides à l'analyse pour la construction de banque d'arbres : étude de l'effort

Nicolas Auclerc & Yves Lepage
{nicolas.auclerc, yves.lepage}@slt.atr.co.jp
ATR 音声言語通信研究所

Résumé - Abstract

La construction de banque d'arbres est une entreprise lourde qui prend du temps. Pour faciliter cette construction, nous voyons la construction de banques d'arbres comme une série d'opérations d'édition et de recherche. Le but de cet article est d'estimer l'effort, en nombre d'opérations d'éditions, nécessaire pour ajouter une nouvelle phrase dans la banque d'arbres. Nous avons proposé un outil, `Boardedit`, qui inclut un éditeur d'arbres et des aides à l'analyse. Comme l'effort nécessaire dépend bien sûr de la qualité des réponses fournies par les aides à l'analyse, il peut être vue comme une mesure de la qualité de ces aides. L'éditeur d'arbres restant indispensable à notre outil pendant l'expérience, les aides à l'analyse seront donc toujours associées à l'éditeur d'arbres. Dans l'expérience proposée, nous augmentons une banque d'arbres de 5 000 phrases par 1 553 nouvelles phrases. La réduction obtenue est supérieure aux 4/5 de l'effort.

The construction of a treebank is a very cumbersome and time-comsuming process. To speed up this process, we see the process of building a treebank as a sequence of edition and search operations. Our purpose is to assess the effort, measured by operations (cliks and keystrokes), needed to augment a treebank. We have proposed a tool, `Boardedit`, which incorporates a tree editor and parsing aids. However, the effort needed will depend on the quality of the answer of the parsing aids, this can be seen as a measure of the quality of the parsing aids. Our tree editor is essential for our tools during this experiment, parsing aids will be always used with the tree editor. In the experiment, we augment a tree-bank of 5 000 sentences with 1 553 new sentences. We show that the reduction in the number of operations is more than 4/5 of the effort.

Mot Clés - Keywords

Banque d'arbres, analogie, filtrage tolérant, éditeur d'arbres, mesure de l'effort.

Treebank, analogy, approximate matching, tree editor, tree-banking assessment.

1 Introduction

Une banque d'arbres est un corpus dans lequel chaque phrase se voit associer une description linguistique, c'est-à-dire un arbre. L'intérêt de ces banques est indéniable comme ressources linguistiques, et les approches d'analyse statistique en ont un besoin crucial. Une célèbre banque d'arbres pour l'anglais est celle de l'université de Pennsylvanie (http://cis.upenn.edu). Notre laboratoire construit aussi des banques d'arbres pour l'anglais, en collaboration avec l'université de Lancaster (Black et al. 96), pour le français et pour le Japonais. Nous voyons la construction de banque d'arbres comme une série d'opérations d'édition et de recherche. Nous avons proposé un outil, `Boardedit` (voir figure 1), qui inclut un éditeur d'arbres (Lepage & Auclerc 00) et des aides à l'analyse. Cet article a pour but d'estimer l'effort en nombre d'opérations d'édition, et non en temps, nécessaire pour augmenter une banque d'arbres avec cet outil. Comme cet effort dépend bien sûr de la qualité des réponses fournies par les aides à l'analyse, il peut être vu comme une mesure de la qualité de ces aides. Bien que l'éditeur d'arbres soit historiquement présent, dans notre outil, avant les aides à l'analyse, il en reste indispensable. Dans un premier temps, nous allons présenter ces deux outils, puis nous passerons à l'expérience.

2 Les aides à l'analyse

La construction de banques d'arbres est une entreprise lourde qui prend du temps. Différentes techniques ont déjà été proposées pour la simplifier. (Brants & Crocker 00) proposent d'utiliser des mécanismes probalilistes dans les modèles de résolution de l'ambiguïté chez l'homme pour créer des analyseurs stochastiques afin de transporter les performances d'analyse linguistique, de l'homme, à un analyseur. (Black et al. 96) montrent que pendant la construction de la banque d'arbres ATR-Lancaster, l'utilisation d'analyseurs pour augmenter cette même banque d'arbres, provoque un « un effet boule de neige » : plus la banque d'arbres est importante, plus son extension est rapide et fiable. Dans notre méthode, nous utilisons aussi des techniques basées sur l'exemple. Toutefois, nous utilisons des aides à l'analyse et non des analyseurs. Dans un précédent article présentant notre outil (Lepage & Ando 96), nous donnions, par ordre de complexité, trois méthodes de recherche associées à un éditeur d'arbres. Les étapes suggérées, c'est-à-dire l'utilisation des ces méthodes de recherche pour la construction de banques d'arbres, se retrouvent énumérées de façon différente ici, plus en adéquation avec le souci de minimiser l'effort pour augmenter la banque d'arbres. Nous proposons donc les étapes suivantes:

1. Rechercher si la nouvelle phrase existe déjà dans la banque d'arbres. Pour cela, utiliser la méthode de recherche par filtrage exacte. S'il n'y a pas de résultat, continuer.

2. Utiliser la complétion par analogie (Lepage 99): cette méthode est plus qu'une simple méthode de recherche (elle repose sur l'analyse par l'analogie et construit un candidat pour la nouvelle phrase à partir de la banque d'arbres.) Adapter l'arbre si nécessaire. S'il n'y a pas de résultat, continuer.

3. Rechercher une phrase similaire. Pour cela, utiliser la méthode de recherche par filtrage tolérant. Si des résultats sont obtenus, adapter la structure associée pour obtenir la nouvelle structure linguistique. S'il n'y a pas de résultat satisfaisant, continuer.

4. Construire entièrement l'arbre à la main à partir de zéro .

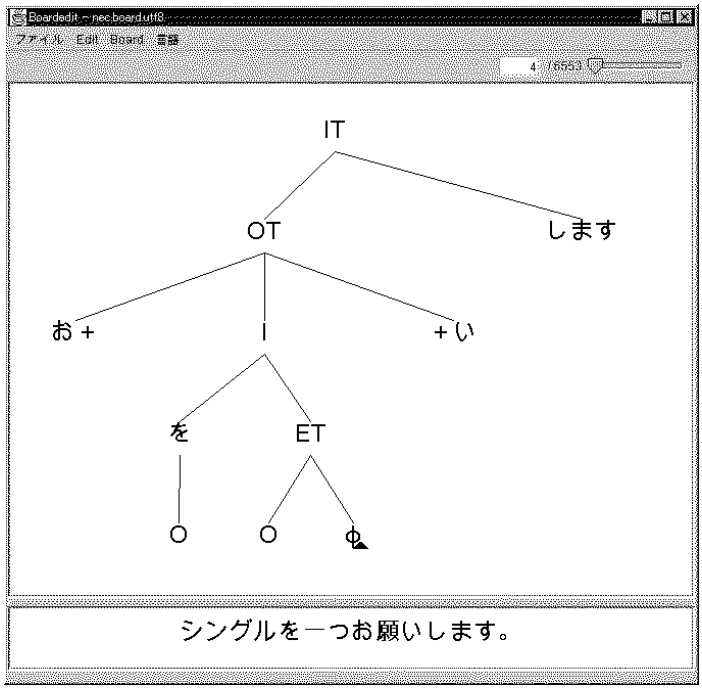

Figure 1: une phrase en japonais et sa structure affichée sous Boardedit

3 L'éditeur d'arbres

Toujours dans le souci de minimiser l'effort pour augmenter la banque d'arbres, nous avons dû réaliser un outil d'édition d'arbres dont le but est de faciliter la saisie d'arbres linguistiques. Grâce à un parallèle (en terme d'opérations et de fonctions) avec l'édition de texte, l'utilisateur n'a pas besoin de se familiariser avec les fonctions d'éditions, ce qui rend l'utilisation de notre éditeur d'arbres intuitive. De plus, cet éditeur est économique dans le sens où dessiner un arbre requiert moins d'opérations que le saisir sous sa forme parenthésée. Les fonctionalités de base de notre éditeur d'arbres ont déjà été présentées dans un précédent article (Lepage & Ando 96). Nous nous limiterons à citer les fonctionnalités démontrant le parallèle entre notre modèle d'édition d'arbres et la forme parenthésée.

3.1 Le modèle d'édition d'arbres

Dans notre modèle d'édition, un nœud ne porte que son étiquette et aucune autre information. De ceci découle un parallèle entre d'un côté les nœuds et les sous-arbres et de l'autre les mots et les lignes (voir tableau 1). De cette façon, toutes les fonctions d'édition (clic de souris, sélection, insertion, copier, coller, etc.) pour les arbres auront le même comportement que sous un éditeur de textes.

Bien que le parallèle montre clairement qu'un nœud est différent d'une étiquette, cette distinction n'est pas intuitive pour un utilisateur novice. Les gens assimilent généralement une étiquette à un nœud. Pour que notre éditeur d'arbres reste intuitif, nous nous efforçons de contredire le moins possible cette façon de penser.

Tableau 1: parallèle entre édition de textes et d'arbres

arbre	**texte**
étiquette	mot
nœud	–
sous-arbre complet	ligne

Nous distinguons trois moyens pour manipuler un arbre:

- le clavier: le tableau 2 montre que chaque frappe de touche compte pour une opération;
- la souris: elle permet à l'utilisateur de créer de nouveaux nœuds en cliquant sur les zones sensibles[1] et de sélectionner des noeuds ou des sous-arbres complets en vue d'opérations comme le copier/coller ou le glisser/déplacer;
- le presser-papier: il permet à l'utilisateur de dupliquer ou de transférer des sous-arbres en deux frappes de touche seulement (une pour le copier, une pour le coller.)

Tableau 2: équivalence entre fonctions d'édition dans un arbre et dans un texte

clic	**effet**	**emplacement**		**coût**
		texte	**arbre**	
simple (zone sensible)	curseur sur...	–	nouveau nœud	1
simple (une selection)	glisser/déplacer	texte	arbre	1
simple (un nœud)	curseur sur...	mot	nœud	1
double (un nœud)	sélectionne...	le mot	le nœud	2
triple (un nœud)	sélectionne...	la ligne	le sous-arbre complet	3
touche	**effet**	**emplacement**		**coût**
		texte	**arbre**	
<esp>	crée ...	un mot	un nœud frère droit	1
<ret>	crée ...	une ligne	un nœud fils	1
<↑>	curseur sur...	ligne au dessus	nœud père	1
<↓>	curseur sur...	ligne au dessous	premier nœud fils	1

3.2 Le gain en utilisant l'éditeur d'arbres

La partie supérieure du tableau 3 montre les correspondances entre les touches listées dans le tableau 2 et les séparateurs utilisés dans la forme parenthésée. Ceci montre clairement que notre modèle d'édition suit le modèle courant d'édition de textes.

[1]Ces zones sensibles permettent à l'utilisateur d'insérer un nouveau nœud en cliquant avec la souris. Elles sont attachées à un nœud et divisées en quatre catégories: zone père, zone frère droit, zone frère gauche et zone fils.

Tableau 3: les opérations en correspondance avec les séparateurs

séparateur	**touche**
(	<ret>
,	<esp>
)	<↑>
séparateur	**clic**
[ )]*,	<simple clic>

Il en résulte qu'un arbre tel que A(B(C,D)),E sera saisi dans notre éditeur par un simple remplacement des séparateurs par les touches associées comme le montre le tableau 4. Ainsi, l'effort pour saisir un arbre sous notre éditeur est le même que pour saisir ce même arbre sous forme parenthésée sous un éditeur de textes. Toutefois, il ne nous est pas nécessaire d'entrer au clavier les parenthèses fermantes pour terminer l'arbre. En utilisant un simple clic de souris, nous pouvons réduire le nombre de frappes de touche: n'importe quelle séquence de parenthèses fermantes suivie par une virgule peut être remplacée par un simple clic de souris qui créera ainsi un nouveau nœud au bon endroit (voir la partie inférieure du tableau 3). L'arbre précédent pourra donc être saisi de différentes façons comme le montre le tableau 4. De cette manière, le nombre d'opérations, c'est-à-dire l'effort, est simplement le nombre de nœuds pour créer la structure, plus le nombre total de caractères dans les étiquettes de nœuds.

Tableau 4: comparaison entre la forme parenthésée et la saisie sous l'éditeur d'arbres.

étape	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
forme parenthésée	A	(	B	,	C	(	D	)	)	,	E
structure d'arbre	A	<ret>	B	<esp>	C	<ret>	D	<↑>	<↑>	<esp>	E
coût unitaire	1	1	1	1	1	1	1	1	1	1	1
coût total	11										
structure d'arbre	A	<ret>	B	<esp>	C	<ret>	D	<simple clic>			E
coût unitaire	1	1	1	1	1	1	1	1			1
coût total	9										

Dans tous les cas, saisir ou bien même modifier un arbre sous notre éditeur nécessite moins d'effort, mesuré par le nombre d'opérations (clics et frappes de touche), que de saisir ou modifier sa forme parenthésée sous un éditeur de textes.

Tableau 5: caractéristiques des données de base et de l'ensemble de test

	tailles		moyenne ± écart-type	nombre de classes ou d'étiquettes ≠
	min	max		
phrases de la base	1	33	7.89 ± 3.61	201
arbres de la base	1	50	10.71 ± 5.77	208
phrases test	1	25	8.58 ± 3.85	133
arbres test	1	39	11.81 ± 6.25	141

4 Expériences

4.1 Les données

Dans cette expérience, nous augmentons une banque d'arbres de 5 000 phrases (données de base) par 1 553 nouvelles phrases (ensemble de test). Nos données proviennent de la banque d'arbres ATR-NEC (Lepage & al. 98). Les phrases sont des dialogues en japonais de réservation de chambres d'hôtel. Les structures des arbres utilisent une représentation en dépendance. Les caractéristiques de ces deux ensembles de données figurent dans le tableau 5.

Nous utilisons les étapes présentées dans la section 2 pour incorporer les 1 553 phrases à la banque d'arbres. En réalité, pour chacune de ces 1 553 phrases, nous connaissons déjà la stucture linguistique voulue. En comparant ces structures avec les résultats obtenus par la recherche par filtrage exacte, la complétion par analogie et la recherche par filtrage tolérant, nous pouvons mesurer l'effort nécessaire pour obtenir la structure exacte quantifiée par le nombre d'opérations (clics et frappes de touche).

L'éditeur d'arbres étant considéré comme indispensable à notre outil, la méthode proposée ne sera pas étudiée sans son utilisation. L'expérience ne comportera donc pas le cas où les aides à l'analyse sont associées à un simple éditeur de textes pour estimer l'effort.

4.2 Lignes de références

La première ligne de référence est obtenue en faisant comme si nous devions saisir, sous un éditeur de textes, la structure linguistique sous forme parenthésée de chacune des 1 553 phrases. Ceci nous coûterait 88 394 frappes de touche (nombre total de caractères des étiquettes et des séparateurs.) La deuxième ligne de référence est obtenue en utilisant notre éditeur d'arbres. Si nous devions construire les structures linguistiques des 1 553 phrases à la main, ceci nous coûterait 13 006 clics de souris pour dessiner la structure et 41 789 frappes de touche pour saisir les étiquettes. Au total, cela représente 54 795 opérations.

On voit donc comme nous l'avons dit ci-dessus, que l'utilisation de notre étiteur d'arbres facilite la saisie. L'effort est réduit de $(88\ 394 - 54\ 795)/88\ 394 = 38\%$.

4.3 La méthode proposée avec les aides à l'analyse

4.3.1 Recherche par filtrage exact

Dans un premier temps, nous utilisons la recherche par filtrage exact. Sur les 1 553 phrases (suites de classes morpho-syntaxiques), 493 étaient déjà présentes dans la banque d'arbres. Dans 465 cas, la méthode nous donne une réponse exacte, ce qui signifie que nous avons, dès la première étape, sans la moindre opération à effectuer, obtenue la structure linguistique (0 opération). Dans les 28 cas restants, une structure différente nous est retournée, due à des choix différents pour la représentation des structures. Par exemple, la représentation de « すぐにできますよ。 » /sugu ni dekimasu yo/ (je peux le faire tout de suite) et celle de « 三号車にありますよ。 » /san go sya ni arimasu yo/ (il se trouve dans la voiture 3) ne sont pas les mêmes, alors que les suites de classes morpho-syntaxiques sont bien les mêmes: mais すぐに /sugu ni/ (tout de suite) est en fonction adverbiale alors que 三号車に /san go sya ni/ (dans la voiture 3) est un complément de lieu. Nous avons calculé l'effort nécessaire si nous avions édité les 28 structures erronées. Cet effort serait l'effort nécessaire pour modifier à la main ces structures erronées, en vue de les rendre identiques aux structures exactes. Nous avons compté un total de 64 opérations en utilisant notre éditeur d'arbres.

4.3.2 Complétion par analogie

Dans les cas où les phrases ne sont pas exactement analysées par la recherche par filtrage exacte ($1553 - 465 = 1088$ phrases), nous passons à la technique de complétion par analogie.

Tableau 6: résultats de la complétion par analogie

	nombre de phrases	**pourcentage**	
total	1 088		100%
analysées	701	100%	64%
exactement analysées	337	48%	30%

Les résultats de la complétion par analogie[2] sont comparés avec les structures liguistiques exactes. Dans 64% des cas (701 phrases), nous obtenons au moins un candidat. Dans un peu moins de la moitié des cas (337 phrases), une des structures linguistiques proposées est exacte, et il n'y a donc aucun effort d'édition à faire. Toutefois, un parcours des résultats est nécessaire: dans le pire des cas, il coûte 5 clics de souris. Au total, cela représente $337 \times 5 = 1\ 685$ opérations.

Si nous désirons éditer les structures proposées dans les $701 - 337 = 364$ autres cas, l'effort nécessaire est mesuré par les distances d'éditions entre arbres. En moyenne, nous obtenons 3,95 arbres proposés, ce qui implique que, dans le pire des cas, nous avons besoin de $3,95 \times 364 = 1438$ clics pour parcourir les candidats afin de trouver la structure linguistique la plus proche de la reponse exacte. Nous comptons 1891 clics pour transformer le plus

[2]Nous ne prenons que les dix premiers résultats sortis. Nous aurions pu en prendre plus. Toutefois, des expériences ont montré que cela n'aurait pas d'effets significatifs (seulement 1% d'augmentation du nombre de phrases exactement analysées, soit 9 phrases, quand on passe 10 à 90 résultats sortis).

proche candidat en la structure linguistique exacte. Au total, ceci nous amène à $1891 \times (1 + 3,21) = 7\ 961$ opérations (clics et frappes de touche) pour effectuer les transformations[3].

4.3.3 Recherche par filtrage tolérant

Dans les cas où les phrases ne sont pas exactement analysées par la complétion par analogie ($1\ 088 - 337 = 751$ phrases), nous utilisons la recherche par filtrage tolérant. Cette méthode consiste à trouver les phrases les plus proches dans la banque d'arbres (rappelons que nous recherchons des suites de classes morpho-syntaxiques, et non pas des séquences de mots), et ensuite proposer les structures linguistiques des phrases trouvées. En moyenne, les arbres proposés sont à une distance de 5,75 nœuds de l'arbre desiré. Au total, la transformation des ces arbres nous coûterait 4 319 opérations (clics et frappes de touche.)

À comparer avec le chiffre précédent de 7 961 opérations obtenu dans la section précédente, ceci montre que, lorsque la complétion par analogie ne délivre pas d'analyse exacte, il est préférable d'utiliser la recherche par filtrage tolérant plutôt que d'adapter à la main un arbre proposé par complétion par analogie.

4.3.4 Construction de l'arbre à partir de zéro

Dans cette expérience d'incorporation de 1 553 phrases nouvelles à une banque de 5000 arbres, nous n'avons jamais eu besoin de construire un arbre à partir de zéro car éditer un arbre obtenu par filtrage tolérant est toujours plus rapide que recontruire tout l'arbre à la main.

5 Synthèse de l'expérience

Nous avons synthétisé les résultats de cette expérience en comparant le nombre d'opérations (clics et frappes de touche) en trois groupes : en utilisant la méthode proposée (les aides à l'analyse plus l'éditeur d'arbres), en utilisant un éditeur de textes pour saisir les arbres sous forme parenthésée (première ligne de référence) et en utilisant l'éditeur d'arbres seul (deuxième ligne référence). Ces résultats sont rassemblés dans le tableau 7. Nous avons ajouté le nombre de fois ou l'utilisateur doit cliquer sur les menus pour activer les aides à l'analyse.
Il apparaît clairement que la méthode proposée dans la section 2 est extrêmement bénéfique. Elle réduit l'effort, en nombre d'opérations, par $(54\ 795 - 10\ 663)/54\ 795 = 81\%$ par rapport à l'utilisation de l'éditeur d'arbres seul. Comparée avec l'utilisation d'un éditeur de textes pour saisir les formes parenthésées, cette méthode réduit l'effort de $(88\ 394 - 10\ 663)/88\ 394 = 88\%$. Rappelons que l'utilisation de notre éditeur d'arbres seul réduit l'effort de 38% (voir la section 4.2) par rapport à l'utilisation d'un éditeur de textes pour saisir les formes parenthésées.

6 Travaux futurs

Les précédent calculs prennent en compte le pire des cas à chaque utilisation des aides à l'analyse. Il serait plus juste d'affiner les opération en terme de suppressions, d'insertions et

[3] Un clic de souris pour positionner le curseur plus la taille moyenne des nœuds en caractères (3,21).

Tableau 7: synthèse des résultats

méthode	nombre d'opérations	nombre de phrases
boutons du menu	$3 \times 1\ 553$	
filtrage exact	0	465
complétion par analogie	1 685	337
filtrage tolérant	4 319	751
méthode proposée	(total)	(total)
(aides à l'analyse et éditeur d'arbres)	10 663	1 553
2ème ligne de référence		
(éditeur d'arbres)	54 795	1 553
1ère ligne de référence		
(éditeur de textes)	88 394	1 553

de remplacements pour calculer l'effort plus précisément. L'insertion est un clic de souris, plus le nombre moyen de caractères par nœud. Une supresion est un double-clic de souris pour sélectionner un nœud (ou un triple-clic pour sélectionner un sous-arbre complet) et la touche `<del>`. Le remplacement compte pour un double-clic pour sélectionner un nœud (ou un triple-clic pour sélectionner un sous-arbre complet), suivi par le nombre de caractères du nœud. Disons aussi que nous n'avons pas utilisé le presse-papier qui devrait aussi réduire le nombre d'opérations.

Dans cette expérience, nous n'avons pas pris en compte le temps que prend chaque opération. On ne peut nier que le changement entre le clavier et la souris (et réciproquement) prend du temps. Ainsi, bien que nous prévoyions que la mesure du temps serait en faveur de l'utilisation de la méthode proposée, le fossé entre les différentes méthodes pourait être réduit.

Dans le futur, nous voulons intégrer plus entre elles les aides à l'analyse. Un premier pas, déjà effectué, a consisté à incorporer la recherche par filtrage exacte dans la complétion par analogie. De cette façon, l'utilisateur n'a plus besoin de séparer les deux premixeres étapes. Le deuxième pas sera d'intégrer la recherche par filtrage tolérant après l'échec de la complétion par analogie.

7 Conclusion

Pour accélérer la construction de banques d'arbres, nous avons proposé un outil, un éditeur d'arbres avec des aides à l'analyse. Le souci de minimiser l'effort pour augmenter la banque d'arbres, nous a amené à proposer une méthode comportant l'utilisation d'aides à l'analyse et des étapes d'éditions. Les aides à l'analyse ont pour but d'obtenir la structure linguistique exacte ou celle la plus proche. L'éditeur d'arbres a pour but de réduire le nombre d'opérations (clics et frappes de touche) pour transformer le plus proche candidat en la structure linguistique exacte. Nous avons montré l'importance des réductions du nombre d'opérations nécessaires à l'augmentation d'une banque d'arbres de 5 000 phrases par 1 553 nouvelles phrases. Cette réduction est supérieure aux 4/5 de l'effort total.

Références

Ezra Black, Stephen Eubank, Kashioka Hideki, David Magerman, Roger Garside and Geoffrey Leech
Beyond Skeleton Parsing: Producing a Comprehensive Large–Scale General–English Treebank with Full Grammatical Analysis
Proceedings of COLING-96, Copenhagen, August 1996, pp. 107-112.

Yves Lepage & Ando Shin-Ichi
Un éditeur pour la construction de banques d'arbres
Actes de TALN-96, Marseille, mai 1996, pp. 104-111.

Goh Chooi Ling
Penyunting Papan (Board Editor)
Projek tahun akhir, Pusat Pengajian Sains Komputer, Universiti Sains Malaysia, 1996.

Yves Lepage, Ando Shin-Ichi, Akamine Susumu, Iida Hitoshi
An annotated corpus in Japanese using Tesnière's structural syntax
ACL-COLING Workshop on Processing of Dependency-Based Grammars, Montréal, August 1998, pp. 109–115.

Yves Lepage
Open Set Experiments with Direct Analysis by Analogy
Proceedings of NLPRS-99, Beijing, November 1999, pp 363-368.

Thorsten Brants & Matthew Crocker
Probabilistic Parsing and Psychological Plausibility
Proceedings of COLING 2000, vol 1, Saarbrücken, July-August 2000, pp. 111–117.

Yves Lepage & Nicolas Auclerc
A tool to build a tree bank for conversational Chinese
Proceedings of ICSLP 2000, vol IV, Beijing, October 2000, pp. 985–988.

Thorsten Brants & Oliver Plaehn
Interactive Corpus Annotation
Second International Conference on Language and Resources (LREC-2000), Athens, pp. 453-459.