Extraction automatique de réponses : implémentation du système ExtrAns[1].

Jawad Berri, Diego Mollá Aliod et Michael Hess

Département d'Informatique (IFI), Équipe de Linguistique Informatique
Université de Zurich
Winterthurerstrasse 190, CH-8057 Zurich Suisse
Tél. : (41 1) 63 54 308 - Fax : (41 1) 63 56 809
{berri,molla,hess}@ifi.unizh.ch
http://www.ifi.unizh.ch/groups/CL/

Résumé

Nous décrivons dans cet article un système d'extraction automatique de réponses. L'extraction automatique de réponses (EAR) a pour but de trouver les passages d'un document qui répondent directement à une question posée par un utilisateur. L'EAR est plus ambitieuse que la recherche d'informations et l'extraction d'informations dans le sens que les résultats de la recherche sont des phrases et non pas des documents en entier, et dans le sens que les questions peuvent être formulées de façon libre. Elle est par contre moins ambitieuse que les systèmes questions-réponses car les réponses ne sont pas générées à partir d'une base de connaissance, mais extraites des textes.

La version actuelle d'ExtrAns permet d'analyser la documentation en ligne (en Anglais) du système Unix (les "man pages"), et construit une représentation sémantique (sous forme d'expressions logiques) des phrases. Un programme de démonstration de théorèmes trouve ensuite les passages pertinents qui sont mis en évidence dans leur contexte.

1. Introduction et état de l'art

L'interrogation de bases de données textuelles est l'un des domaines où la technologie visant la compréhension du langage naturel n'a pas vraiment réussi à fournir des systèmes très convaincants. Les systèmes qui lisent des textes, en assimilent le contenu, et répondent à des questions formulées librement sur ces textes seraient d'une très grande utilité pour une large variété d'applications, notamment si aucune restriction de rédaction n'est imposée aux questions *et* aux textes. Ils seraient la solution parfaite au problème de la surabondance de l'information qui s'accentue de plus en plus de nos jours avec le réseau internet. Même si

[1] Ce travail est financé par le fond national suisse de la recherche scientifique, n° de contrat : 1214-45448-95.

actuellement (et dans les prochaines années à venir) il est possible d'implémenter pareils systèmes, ceux-ci restent limités à des domaines particuliers, à des corpus restreints, et avec des coûts de développement assez élevés. Ces systèmes (comme LILOG (Herzog et Rollinger 1991)) restent des cas prototypiques de systèmes de laboratoire.

Actuellement, il n'y a que deux sérieux prétendants pour traiter des textes en grandes quantités : La recherche d'informations et l'extraction d'informations. Malheureusement les deux techniques ont de sérieux inconvénients.

Les techniques standards de *recherche d'informations* ("Information Retrieval"), donnent la possibilité à l'utilisateur de poser des questions de tous types et permettent la recherche dans des collections de textes très importantes (plusieurs giga-octets). Des exemples de ces systèmes sont : SMART (Salton, 1989), SPIDER (Schäuble,1993), GURU (Maarek, 1992). Le système GURU par exemple, permet de générer des systèmes d'aide pour divers systèmes documentés. GURU se base sur la notion d'"Affinité Lexicale" (sorte de collocation) entre deux unités du langage. Pour tout texte de la documentation et toute question, GURU calcule les affinités lexicales (AL) et les trie selon la quantité d'information véhiculée par chaque mot composant les AL. Les textes qui sont présentés à l'utilisateur sont ceux qui contiennent en priorité les AL de la question. Les systèmes de ce type sont très avantageux puisqu'il ne nécessitent aucune connaissance préalable du domaine traité. Cependant, leur inconvénient est qu'ils sont conçus pour rechercher des textes en entier ce qui les rend peu utiles pour des questions très spécifiques et pour des documents dont la taille dépasse deux ou trois paragraphes. Par ailleurs, les techniques classiques de recherche d'informations possèdent des limitations inhérentes, elles traitent les mots comme des items syntaxiquement isolés (d'où la confusion entre "livre de l'année" et "année du livre").

Les techniques d'*extraction d'informations* quant à elles sont rapides, elles sont appropriées lorsqu'il s'agit de textes de tous types et de taille variable, mais elles sont limitées à la recherche d'informations dans des schémas d'informations prédéfinis à l'avance. Les systèmes développés dans le cadre de la série de conférences MUC ("Message Understanding Conference") (ARPA, 1996) en sont des exemples, ils permettent d'extraire des informations spécifiques depuis des textes de presse. Cependant certains de ces systèmes, comme celui développé par R. Grishman (Grishman, 1996), et le système FASTUS (Appelt et al., 1993) qui utilisaient des modules d'analyse syntaxique, ont été simplifiés, pour des raisons de rapidité, en supprimant du traitement ces modules d'analyse syntaxique. Les résultats ont été bien sûr affectés. Dernièrement, dans le cadre de MUC-6 (Grishman et Sundheim, 1996), des tentatives ont été initiées pour remettre l'accent sur des analyses plus profondes du langage.

2. Extraction automatique de réponses : vers une approche linguistique

Il y a actuellement un besoin croissant de systèmes, ayant une très grande précision et un fort taux de rappel, capables de localiser des informations dans des collections de textes dont la taille ne dépasse pas le giga-octet. Ces systèmes permettraient de traiter des *questions formulées librement* pour l'interrogation de collections de textes regroupant des *textes bruts*. On peut citer pour exemple les systèmes d'accès aux manuels techniques en ligne, les systèmes d'aide à l'utilisation de logiciels complexes, et les systèmes publics d'interrogation accessibles via

l'internet. Pour ces tâches, une très grande précision est requise (les questions peuvent être très spécifiques), la plupart du temps il est vital d'avoir un taux de rappel presque parfait (certaines explications recherchées dans les manuels techniques sont uniques), et dans certains cas le temps de réponse est un facteur important (par exemple pour des activités où il faut retrouver l'information exacte pour remédier à un imminent dysfonctionnement d'un système). Ce dont on a besoin, c'est un système qui trouve, avec une très grande précision, les *passages et les phrases* dans un texte (ou une collection de textes) dont le *sens* exprime la réponse exacte à une question spécifique. C'est l'idée principale de l'extraction automatique de réponses. Le fait d'opérer au niveau du sens des phrases (des questions et des textes) implique l'utilisation d'informations linguistiques (syntaxiques et sémantiques) qui peuvent s'avérer très coûteuses à implémenter. Cependant l'application qu'on envisage est destinée à la recherche dans des collections moyennes de textes (quelques centaines de kilo-octets, parfois quelques méga-octets), et à des domaines spécifiques. Ceci fait de l'extraction automatique de réponses un compromis raisonnable entre d'une part les systèmes questions-réponses visant à proposer des réponses à tous types de questions, et d'autre part la recherche d'informations et l'extraction d'informations qui peuvent traiter des corpus très importants.

Nous allons décrire dans ce qui suit ExtrAns, un système d'extraction automatique de réponses utilisant (un sous ensemble de) la documentation en ligne du système Unix (les "man pages"). Un premier prototype du système est opérationnel, il permet de traiter des questions formulées librement et les textes originaux de la documentation du système Unix, même si dans certains cas les données (les documents et les questions) ne peuvent pas être analysées complètement. Le système est extensible d'une façon incrémentale. En effet, les perfectionnements de la grammaire et/ou de la composante sémantique améliorent automatiquement la précision de la recherche sans que cela n'affecte les autres composants du système.

3. Les composants et les exigences du système

Si l'on considère qu'un système d'extraction automatique de réponses doit être capable de traiter des textes très techniques, il a besoin d'un module très fiable qui réalise toutes les segmentations en mots et en phrases d'un texte (qu'on appellera dans ce qui suit segmenteur, "tokeniser" en anglais), une grammaire ayant une large couverture, un analyseur syntaxique assez robuste, un traitement des ambiguïtés, un module qui permet de construire une représentation sémantique à partir des structures syntaxiques des phrases, et un moteur de recherche capable d'utiliser ces représentations sémantiques.

Dans ce qui suit, nous allons décrire en détail les trois composants les plus importants du système disposés de manière séquentielle. *Premièrement*, nous présenterons certaines des difficultés engendrées par le prétraitement de manuels techniques qui va au-delà de ce qu'en général un segmenteur devrait faire. Nous ne décrirons pas dans cet article l'analyseur syntaxique utilisé qui est un système existant, qu'on a adapté, appelé "Link Grammar" de Sleator et Temperley (Sleator et Temperley, 1991). "Link Grammar" est, à notre connaissance, le meilleur système disponible pour une utilisation en recherche pour l'anglais. Il est basé sur une grammaire de dépendances et utilise un dictionnaire des formes. L'information syntaxique

est fournie sous forme d'un ensemble de connections étiquetées entre les mots formant la phrase. Même si ce système est assez robuste, on l'a enrichi avec quelques stratégies pour transformer les constituants des phrases non traités en mots-clés. *Deuxièmement*, nous décrirons les principes permettant de construire des représentations sémantiques à partir des structures syntaxiques produites par "Link Grammar". Nous ne présenterons pas en détails le module qui permet de résoudre les ambiguïtés pour lequel on a adopté l'approche de Brill et Resnik (Brill et Resnik, 1994). *Troisièmement*, nous montrerons comment nous traitons les ambiguïtés syntaxiques qui persistent après le précédent processus de désambiguïsation.

4. Prétraitement du langage technique

L'analyse du langage technique est en général plus simple que l'analyse de textes portant sur divers domaines (articles de presse, etc.), cependant elle requiert un important prétraitement. Cela concerne les segmentations, la normalisation et l'analyse de la structure des documents.

4.1. Segmentations et normalisation

En général, la segmentation d'un texte revient essentiellement à identifier les mots et les phrases. Cependant, dans les textes très techniques comme les pages du manuel d'Unix, cette tâche requiert une attention particulière. Hormis les formes régulières des mots, le segmenteur d'ExtrAns doit reconnaître une foule de symboles spécifiques au domaine traité, il doit ensuite les représenter sous la forme d'expressions normalisées. Les cas suivants en sont des exemples :

Les noms de commandes : eject, nice (problème : identifier ces mots lorsqu'ils sont utilisés comme des noms de commandes dans des phrases comme "**eject** is used for ...", et leur utilisation standard comme dans "It is not recommended to physically eject media").

Les noms de fichiers et les chemins : /usr/bin/X11, usr/5bin/ls, /etc/hostname.le (problèmes : la barre oblique ("/") en tête, à l'intérieur et en queue, les nombres et les points).

Les options de commandes : -C, -ww, -dFinUv (problème : les séquences précédées d'un tiret peuvent être utilisées comme options de commandes ou bien dans des exemples du type "... whose name ends with .gz, -gz, .z, -z, _z or .Z and which begins ...").

Les noms de variables : *filename1*, *device*, *nickname* (problèmes : reconnaître les mots utilisés comme noms de variables, la plupart du temps comme arguments de commandes, par exemple "... the first *mm* is the hour number; *dd* is the day ...").

Caractères spéciaux faisant partie des mots : AF_UNIX, sun_path, ^S(CTRL-S), K&R C, ANSI C, C++, %, %% (exemple : "A single % is encoded by %%."), marques de ponctuation (exemples : "... corresponding to cat? or fmt?", "/usr/man/man?", "name@domain", [host!...host!]host!username", et "<signal.h>".

La normalisation de ce type de mots spéciaux consiste, entre autres, à les codifier d'une façon appropriée, comme c'est le cas, par exemple, des noms de commandes (autrement l'analyseur syntaxique échoue). Heureusement, les pages du manuel d'Unix contiennent des informations, matérialisées par les *commandes de formatage*, qui vont au delà du simple niveau textuel.

Ainsi, les noms de commandes sont, en règle générale, en gras, et les expressions utilisées comme variables, en italique, par exemple :

compress [-cvf] [-b *bits*] [*filename* ...]

Ce type d'informations est extrait des commandes de formatage et codé avec les mots correspondants pour pouvoir être reconnus par les modules suivants du système. Par exemple, le mot "eject" est codé "eject.com" quand il est utilisé comme nom de commande, et le mot "filename" est codé "filename.arg" quand celui-ci est utilisé comme argument.

4.2. L'analyse de la structure du document

Les commandes de formatage sont utilisées quelquefois dans les pages du manuel d'Unix de façon non systématique (ces pages ont été écrites par différentes personnes). Pour extraire les informations liées aux commandes de formatage (voir le paragraphe précédent), le segmenteur doit palier à ces inconsistances dans les textes sources. Il doit effectuer un important travail d'analyse de la structure des documents en collectant, par exemple, les noms de commandes et les arguments (de chaque page du manuel) des sections "SYNOPSIS" et "NAME". Une fois ce travail fait, il est alors possible d'analyser la section "DESCRIPTION" où on retrouve la description de chaque commande. Le traitement des pages du manuel d'Unix devient ainsi un traitement de chacune des sections d'une façon particulière.

Finalement, dans le but de sélectionner et présenter les passages pertinents dans les textes (et les mettre en évidence; voir section 6), le segmenteur associe à chaque mot un certain nombre d'informations en construisant une représentation. Ainsi, il calcule la position de chaque mot dans le texte et associe à chaque mot la phrase et la section où il se trouve.

5. Construction des expressions logiques

La principale caractéristique du système ExtrAns consiste dans le fait que les phrases du texte sont converties en **formules logiques existentiellement fermées** qui encodent les relations sémantiques essentielles entre les mots des phrases. Toutes les formules sont quantifiées existentiellement car, pour les besoins de la recherche d'informations, celles-ci existent dans l'univers du discours. Les verbes (représentés par "evt", pour 'eventualité'[2]), les noms ("object", pour 'objet'), les adjectifs et les adverbes ("prop", pour 'propriété') introduisent des concepts dans le discours auxquels on peut se référer par la suite. En particulier, un prédicat introduit par le verbe "copy" (copier) dans la phrase "cp copies files" (cp copie des fichiers) serait : `evt(copy,e1,[c1,f1])`, où e1 représente le concept, c1 représente la commande cp et f1 représente les fichiers. Un prédicat introduit par le nom "cp" serait : `object(cp,o1,c1)`, où o1 représente le concept "c1 est cp".

Les clauses de Horn, qui sont le sous-ensemble du calcul des prédicats utilisé par Prolog, sont utilisées pour implémenter les expressions logiques. Elles sont construites sur la base d'une ontologie utilisant différents niveaux de représentation, (pour plus de détails voir (Hess,

---

[2] Dans cet article, le mot "éventualité" désigne "eventuality" au même sens que T. Parsons (Parsons, 1990).

1997)). Ainsi, pour chaque verbe le système crée un prédicat d'arité fixe, d'autres prédicats sont créés pour les compléments (obligatoires) et les modifieurs (non obligatoires). Les liens entre, d'une part les expressions construites, et d'autre part les phrases et les mots d'où les expressions ont été déduites sont ajoutées à la représentation.

Les deux exemples suivants illustrent une autre distinction effectuée par le système :

(1) "**cp** copies the contents of *filename1* onto *filename2*"
(**cp** copie le contenu de *filename1* dans *filename2*)

(2) "If the unmount operation fails, **eject** prints a warning message"
(Si l'opération de désinstallation échoue, **eject** imprime un message d'avertissement)

Dans le premier exemple l'événement copier a réellement lieu (dans le monde générique des pages du manuel d'Unix). Cela n'est pas le cas pour les deux actions dans le second exemple (échouer et imprimer) où toutes les deux sont introduites sous la portée de la condition. On voudrait que ces passages soient trouvés pour certaines questions (voir l'exemple de la figure Fig. 1), mais on n'aimerait surtout pas qu'ils le soient lorsqu'il est question des différentes façons dont un utilisateur dispose pour imprimer *intentionnellement* un message ("How do *I* print ... "; "Comment *j*'imprime ..."), par opposition à l'impression des messages d'erreur, qui est simplement un effet de bord. Pour cette raison, nous avons dû modifier les bases de notre ontologie concernant les éventualités de manière à représenter différemment les éventualités qui ont lieu réellement. Toutes les autres sont présumées exister seulement dans l'univers du discours[3]. Par conséquent, pour les deux exemples précédents, on a les deux représentations suivantes[4] :

```
holds(e1)/s1. object(cp,o1,x1)/s1. object(command,o2,x1)/s1.
evt(copy,e1,[x1,x2])/s1. object(content,o3,x2)/s1.
object(filename1,o4,x3)/s1. object(file,o5,x3)/s1.
of(x2,x3)/s1. object(filename2,o6,x4)/s1.
object(file,o7,x4)/s1. onto(e1,x4)/s1.
```

qui peut être paraphrasée de la manière suivante : "x1 se réfère à un objet qui est une commande et qui est cp, x3 et x4 sont des objets qui sont des fichiers et qui sont respectivement filename1 et filename2, x2 est un objet qui est le contenu de x3, e1 consiste en l'événement x1 qui copie x2 dans x4, et e1 a réellement eu lieu (`holds(e1)`). Pour la deuxième phrase on a :

```
object(eject,o8,x7)/s2. object(command,o9,x7)/s2.
evt(print,e8,[x7,x11])/s2. object(message,o10,x11)/s2.
object(warning,o12,x11)/s2. if(e8,e5)/s2.
object(operation,o13,x5)/s2.
object(unmount,o14,x5)/s2. evt(fail,e5,[x5])/s2.
```

---

[3] Notons toutefois, que cette distinction n'est pas encore faite pour les objets et les propriétés.

[4] Les informations relatives aux positions des mots ne sont pas présentées ici.

où ni l'événement "failing" (échouer) ni l'événement "printing" (imprimer) (e5 et e8) ne sont représentés comme ayant eu lieu réellement. L'existence réelle des éventualités peut ainsi être inférée, bloquée, ou bien non spécifiée selon le contexte (Hobbs, 1985).

On peut constater que les expressions logiques générées sont simplifiées. En effet, les pluriels, les modalités et les quantifications sont ignorées; les conditionnels et les négations sont repésentés comme des prédicats normaux et non comme opérateurs logiques. Cela est suffisant et nous paraît mieux adapté à l'extraction automatique de réponses. Par exemple, si l'utilisateur pose la question suivante : "Can cp copy a file onto itself?" (Est-ce que cp peut copier un fichier dans lui-même?), le système peut facilement trouver "cp refuses to copy a file onto itself" (cp refuse de copier un fichier dans lui-même), qui est une information utile pour l'utilisateur. Si on utilisait une représentation sémantique plus complexe, le système devrait recourir à des inférences beaucoup plus coûteuses pour trouver le même résultat.

Notons aussi que certaines informations *typographiques* (et indirectement provenant de la structure du document) extraites du texte par le segmenteur ont été utilisées dans cette représentation. Le fait que "eject" est le nom d'une commande (au lieu du verbe "eject" (éjecter) en anglais) résulte en la création d'un prédicat supplémentaire `object(command,x7).` De manière similaire, "cp" est reconnu comme référant à une commande. Sans le recours à ce type d'informations, la première phrase ne serait pas accessible par des questions du type "What commands copy files? " (Quelles sont les commandes qui copient des fichiers?), la deuxième serait considérée non grammaticale.

Dans le but de répondre aux questions, les procédures standards de réfutation sont utilisées pour trouver *toutes* les preuves possibles de la question par rapport à la base de connaissance. Ainsi, la question

(3) "Which command prints something?"
(Quelle commande imprime quelque chose?)

devient la requête Prolog suivante, après construction de l'expression logique et recherche des synonymes[5]

```
?- findall(S, (object(command,O1,X)/S,
    (evt(display,E,[X,Y])/S; evt(visualize,E,[X,Y])/S;
    evt(print,E,[X,Y])/S), object(N,O2,Y)/S), R).
```

et le moteur d'inférences de Prolog retourne, dans `R=[S1, S2,...]`, les références aux phrases pertinentes, dont la phrase (2).

6. Présentation des résultats (éventuellement ambigus)

La résolution des ambiguïtés est l'un des problèmes auquel notre système a été confronté. L'analyseur syntaxique de Sleator et Temperley (Sleator et Temperley, 1991) ne traite pas les

[5] Les synonymes sont ajoutés en ayant recours à un thésaurus du style WordNet (Miller et al. 1990) que nous sommes en train de développer pour notre application. Notons que "printing" (imprimer) pour le système Unix veut aussi dire "printing to the screen" ou bien "displaying", c'est-à-dire (afficher à l'écran).

ambiguïtés syntaxiques et sémantiques. Une phrase longue pourrait avoir des centaines, voire des milliers d'analyses syntaxiques différentes. ExtrAns essaie de résoudre les ambiguïtés (syntaxiques) en deux étapes. Dans la première étape quelques règles heuristiques éliminent les cas les plus invraisemblables. Un exemple d'une telle règle est : "Une phrase prépositionnelle introduite par ("of") ne peut être attachée qu'au nom ou à la coordination nominale qui la précède immédiatement". D'autres règles, ajoutées au système, éliminent certaines analyses incorrectes non détectées par l'analyseur syntaxique, par exemple l'analyse suivante "(The ((files are deleted) and (directories are copied)))" est incorrecte. Dans la deuxième étape, nous utilisons l'approche de Brill et Resnik (Brill et Resnik, 1994) pour désambiguïser les phrases prépositionnelles. Le module proposé par les deux auteurs a été entraîné sur des données extraites de notre corpus. Avec l'aide de ce module nous pouvons utiliser des données statistiques pour résoudre les attachements des phrases prépositionnelles, la plus importante source d'ambiguïté.

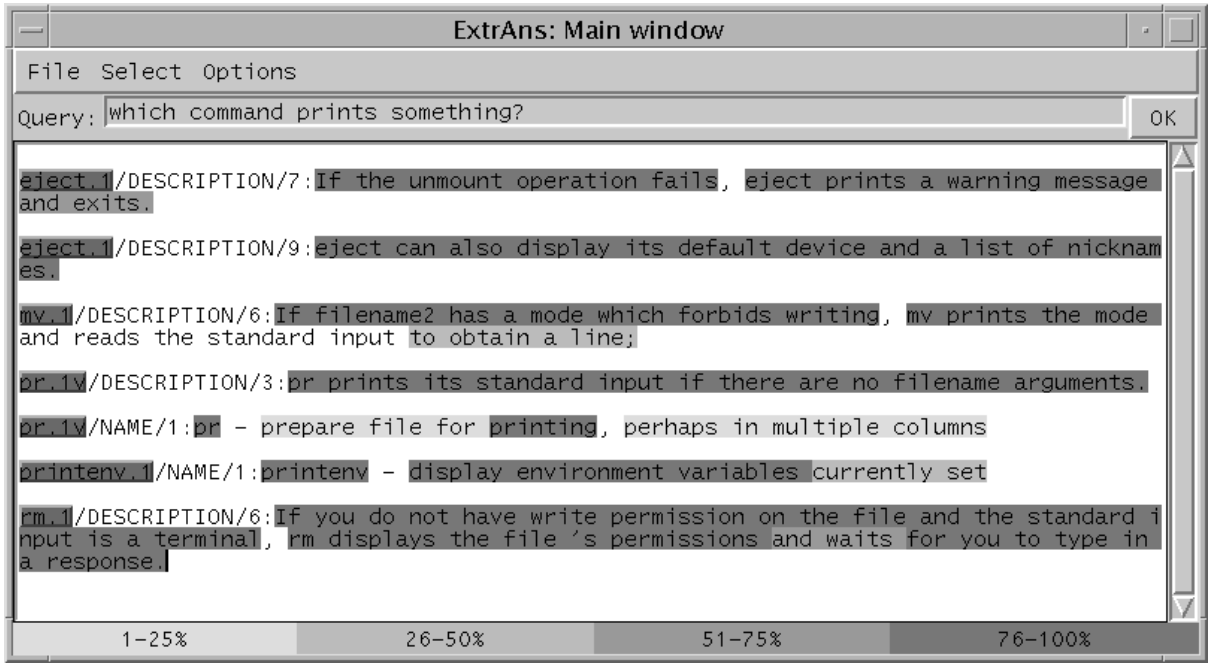

Figure 1 : Le résultat de la question "Which command prints something?" (Quelle commande imprime quelque chose?). (La figure originale est en couleur).

Après ces deux étapes, le nombre des ambiguïtés est réduit même si, pour quelques phrases, certaines persistent car elles ne sont pas traitées par l'algorithme de Brill et Resnik. Si une phrase a plusieurs interprétations irréductibles, ExtrAns les enregistre toutes dans sa base de connaissances. Lorsque l'utilisateur pose une question, une phrase peut être trouvée plusieurs fois (puisque celle-ci peut avoir plusieurs expressions logiques qui sont compatibles avec la question). Ces différentes expressions logiques peuvent mettre en évidence divers mots, comme conséquence aux différentes interprétations de la phrase. ExtrAns traite cela en superposant toutes les mises en évidence des phrases de manière à assigner une couleur plus significative, selon un *schéma gradué de couleurs*, aux parties ayant été trouvées le plus

fréquemment par le moteur d'inférences. Par exemple, si l'on considère la chaîne "If filename2 has a mode which forbids writing, mv prints the mode" (Si filename2 a un mode qui interdit l'écriture, mv imprime le mode), dans la figure (Fig. 1). Celle-ci fait partie de toutes les interprétations de la phrase désignée par `mv.1/DESCRIPTION/6` qui ont été trouvées en réponse a la question de l'utilisateur. Elle est par conséquent mise en évidence avec une couleur plus intense. En effet, c'est cette partie de la phrase qui donne une réponse plus directe à la question.

En outre, il est possible d'accéder à la page complète du manuel qui contient la phrase (simplement en cliquant sur le nom du fichier correspondant se trouvant à gauche de chaque phrase). La page du manuel montre les mêmes mises en évidence de passages permettant ainsi à l'utilisateur de noter rapidement la phrase la plus pertinente et de déterminer, par inspection du contexte, si la phrase contient en effet une réponse à la question. Avec cette façon de présenter les résultats, les possibles ambiguïtés passent inaperçues.

7. Conclusion et recherches futures

Conclusions. Le système tel qu'il est actuellement montre que le *concept* d'extraction automatique de réponses est très utile et qu'il requiert un *effort relativement limité de traitement du langage*. Il montre aussi qu'il est très facile pour les utilisateurs de faire avec les ambiguïtés lorsque celles-ci sont fusionnées, classées, et présentées dans leur contexte.

Caractéristiques à améliorer. Pour que le système soit plus utile, on doit clairement agrandir notre corpus de textes du manuel technique du système Unix (le système utilise actuellement 30 descriptions de commandes). On doit par conséquent améliorer le traitement des phrases non grammaticales par le système. A l'état actuel, le système convertit les mots qu'il ne peut pas analyser en mots-clés. Cette méthode peut être améliorée de manière à traiter *les parties de phrases* lorsque la phrase ne peut pas être analysée entièrement. En plus, pour certaines phrases, le nombre d'analyses syntaxiques reste très important, il est donc indispensable d'améliorer les méthodes utilisées pour désambiguïser et éliminer les analyses les moins plausibles en utilisant éventuellement des informations sémantiques. Nous sommes aussi en train d'étendre de façon plus systématique notre thésaurus des synonymes et hyponymes en utilisant des outils d'acquisition automatique de collocations. On envisage aussi d'ajouter au système la possibilité de résoudre les pronoms et les anaphores nominales. Finalement, nous sommes en train d'améliorer les techniques d'inférence pour augmenter le taux de rappel en intégrant au système des modules qui permettent de prendre en compte, entre autres, le problème de distributivité de la coordination.

Références

ARPA (1996), *Proceedings of the sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, parrainée par l'ARPA, November 1995, Morgan Kaufmann.

Appelt D. E., Hobbs J. R., Bear J., Israel D., Kameyama M., Tyson M. (1993), "SRI: Description of the JV-FASTUS System Used for MUC-5", *Proceedings of the fifth Message Understanding Conf. (MUC-5)*, Baltimore, MD, August 1993, Morgan Kaufmann.

Brill E., Resnik P. (1994), "A rule-based approach to prepositional phrase attachment disambiguation", *in proceedings of the 15th International Conference on Computational Linguistics COLING'94*, Vol. 2, pp. 1998-1204.

Davidson D. (1967), "The logical form of action sentences", *The Logic of Decision and Action*, Rescher N. (ed.), Univ. of Pittsburg Press. Aussi dans *Essays on Actions and Events*, Clarendon Press, 3ème edition, pp. 105-122, 1980.

Grishman R. (1996), "The NYU System for MUC-6 or Where's the Syntax?", in *ARPA 1996.*

Grishman R., Sundheim B. (1996), "Message Understanding Conference - 6: A Brief History", in *ARPA 1996.*

Herzog O., Rollinger C.-R. (1991), *Text Understanding in LILOG*, Herzog O., C.-R. Rollinger C.-R. (eds.), Springer-Verlag, Berlin.

Hess M. (1997), "Mixed-level knowledge representations and variable-depth inference in natural language processing", *International Journal on Artificial Intelligence Tools*, Vol. 6, No. 4, pp. 481-509.

Hobbs J. R. (1985), "Ontological promiscuity", *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 61-69.

Maarek Y. S. (1992), "Automatically Constructing Simple Help Systems from Natural Language Documentation", *Text-Based Intelligent Systems - Current research and Practice in Information Extraction and Retrieval* , ed. Jacobs Paul S., pp. 243-255, Lawrence Erlbaum, Hilsdale.

Miller G. A., Beckwith R., Fellbaum C., Gross D., Miller K. J. (1990), "Introduction to Wordnet: an On-Line Lexical Database", *International journal of Lexicography*, 3 (4), pp. 235-244.

Parsons T. (1990), *Events in the Semantics of English: A study in subatomic semantics*, MIT Press.

Salton G. (1989), *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley Publishing, New York.

Schäuble P. (1993), "SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data", *ACM SIGIR conference on R&D in Information Retrieval*, pp. 318-327.

Sleator D. D.; Temperley D. (1991), "Parsing English with a link grammar", Rapport technique CMU-CS-91-196, Carnegie Mellon University.