Perspectives en TAFD pour auteur monolingue après une première expérience : la maquette LIDIA-1

Hervé BLANCHON

GETA, Institut IMAG (UJF & CNRS)
BP 53, 38041 Grenoble Cedex 9, France
e-mail : Herve.Blanchon@imag.fr

Résumé

La Traduction Automatisée Fondée sur le Dialogue est envisagée comme une solution pratique pour de futurs systèmes permettant à des auteurs monolingues de traduire les documents qu'ils rédigent. Nous avons conçu et réalisé une première maquette, LIDIA-1, qui montre comment une pile HyperCard™ française peut être traduite en allemand, en anglais et en russe. Nous abordons les aspects informatiques, linguistiques et ergonomiques de la maquette, puis nous les discutons dans la perspective d'un futur prototype opérationnel.

Mots clés

TAO interactive, TA Fondée sur le Dialogue, TAFD pour auteur monolingue, désambiguïsation interactive, production de dialogues de désambiguïsation, architecture distribuée, architecture tableau blanc

Introduction

Le projet LIDIA est un cadre d'étude pour la traduction assistée par ordinateur personnelle [1], ou plus précisément la TAFD pour auteur monolingue.

Nous venons de terminer l'implémentation d'une première maquette, LIDIA-1. Nous avons choisi de travailler sur une maquette plutôt que sur un prototype pour pouvoir aborder rapidement tous les aspects — informatiques, liguistiques et ergonomiques — de tels systèmes. Ainsi, même si nous sommes loin d'avoir résolu tous les problèmes, ils sont mis en perspective.

Avant de présenter une démonstration de la maquette, nous introduisons le contexte dans lequel nous l'avons conçue et développée. Nous donnons ensuite quelques détails supplémentaires sur la maquette elle-même, les techniques d'implémentation utilisées, ainsi que sur le processus de désambiguïsation interactive. Finalement, nous discutons quelques points importants dans la perspective de la construction d'un futur prototype opérationnel.

1. Cadre de l'étude

1.1. La TAFD

La TAO interactive a été proposée pour la première fois par M. Kay dans la cadre du projet MIND [2]. D'autres projets ont ensuite expérimenté différentes approches, notamment le projet ITS [3] à Provo (1975 - 1981), le projet Alvey N-trans [4] à Manchester (1985 - 1987), le projet DLT [5] à Utrecht (1982 - 1988), le projet LMT [6] depuis 1989 dans plusieurs centres de recherche IBM, ainsi que le projet JETS [7] depuis 1989 dans les laboratoires IBM à Tokyo.

Avec KBMT-89 [8] à CMU-CCL, des questions de désambiguïsation étaient aussi posées si l'ontologie ne permettait pas à l'*augmentor* de résoudre automatiquement toutes les ambiguïtés.

De ces précédentes expériences, nous avons retenu :

- le type d'interface proposé par KBMT-89,
- le processus de reconnaissance de patrons utilisé dans LMT pour reconnaître certaines ambiguïtés,
- l'architecture distribuée de JETS.

1.2. La maquette LIDIA-1

Nous avons choisi une situation traductionnelle bien précise tant en ce qui concerne le profil de la tâche que le profil de l'utilisateur. Nous avons aussi intégré l'usage d'un processus de désambiguïsation au tout début de la conception de la maquette. Ainsi, les interactions entre les contraintes informatiques, linguistiques et ergonomiques ont été prises en compte très tôt dans nos travaux. L'organisation du processus de traduction est décrit dans [9].

Dans le scénario que nous proposons, un ingénieur français monolingue crée des documents techniques sous forme de piles HyperCard, sur un Macintosh de milieu de gamme, et aide le système à traduire ses piles en anglais, allemand et russe. Nous avons choisi une architecture distribuée : la station de travail de l'auteur communique avec un serveur de traduction hébergé par un mini IBM-4361.

Nous avons construit une pile de démonstration à propos des ambiguïtés du français que nous avons choisi de résoudre dans la maquette.

1.3. La pile de démonstration

Avec HyperCard, l'utilisateur peut construire une pile de façon interactive et incrémentale. Une pile est constituée de cartes. Sur une carte, l'utilisateur peut placer deux types d'objets, des champs et des boutons. Les champs contiennnent du texte. Les boutons sont des éléments actifs sur lesquels on clique pour déclencher des actions.

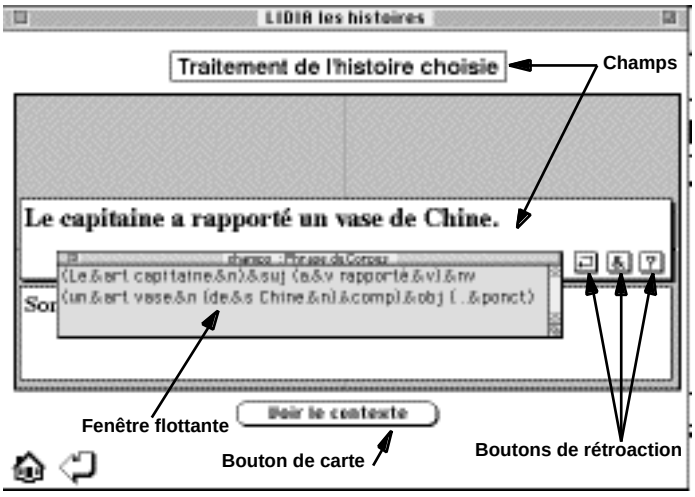

Figure 1: une carte et les objets HyperCard

La pile de démonstration qui se nomme "LIDIA les histoires" est constituée de deux types de cartes, des cartes d'histoires et des cartes de traitement.

Une carte d'histoires rassemble plusieurs histoires qui partagent une phrase ambiguë hors contexte. L'utilisateur est supposé résoudre les ambiguïtés grâce à sa compréhension des histoires. Voici un exemple de carte d'histoires :

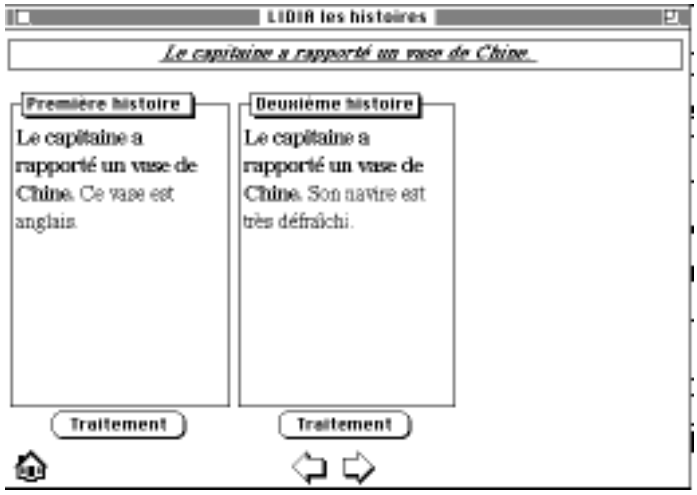

Figure 2 : une carte d'histoires

Pour les besoins de la démonstration, chaque histoire est présentée dans une carte de traitement sur laquelle le contexte de la phrase ambiguë peut être caché ou non.

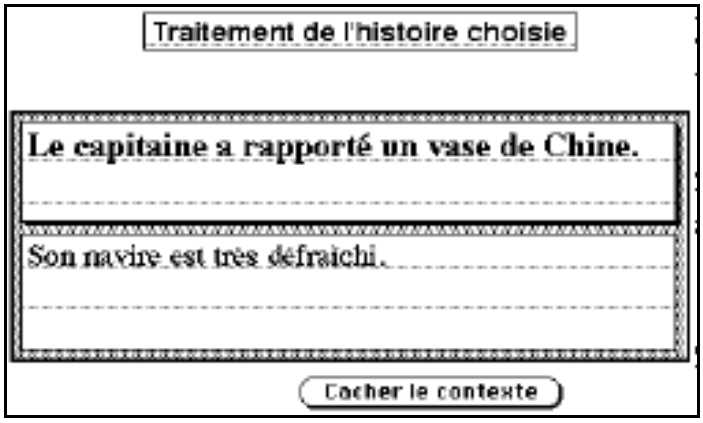

Figure 3: une carte de traitement

Pour que les histoires soient traduites, l'utilisateur va demander la traduction des champs des cartes de traitement. Il faut noter que l'utilisateur n'est jamais interrompu par une question. Les objets montrent qu'ils attendent l'intervention de l'utilisateur afin que la traduction se poursuive, l'utilisateur décide quand intervenir et à quelle question répondre.

2. Une démonstration

L'utilisateur choisit l'outil de sélection LIDIA (✓) et sélectionne un objet à traduire (fig. 4).

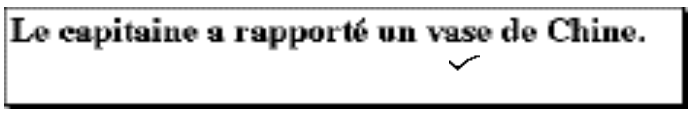

Figure 4 : selection d'un objet

Un bouton de suivi de l'avancement des traitements apparaît sur l'objet sélectionné. Lorsque l'on clique sur ce bouton (fig 5), une fenètre flottante apparaît (fig. 6).

Figure 5 : l'utilisateur demande l'état d'avancement des traitements

Le traitement en cours est distingué en gras, les traitements déjà effectués en style normal et les traitements à venir en italiques. Ainsi, figure 6, le système est en train d'effectuer l'analyse du fragment textuel.

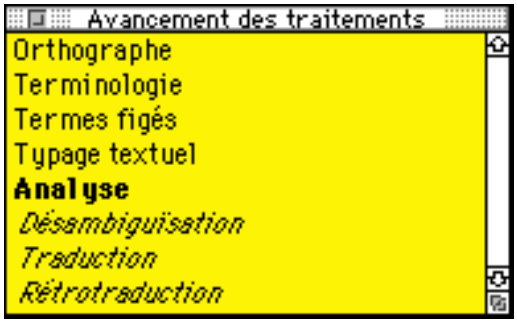

Figure 6 : fenêtre flottante pour l'état des traitements

Si le texte doit être désambiguïsé, le système demande à l'utilisateur d'intervenir. L'utilisateur est averti qu'une nouvelle question est en suspens par un nouvel item dans le menu **Message** ainsi que par un nouveau bouton qui apparait sur l'objet concerné comme dans la figure 7. L'utilisateur peut choisir d'intervenir immédiatement ou plus tard.

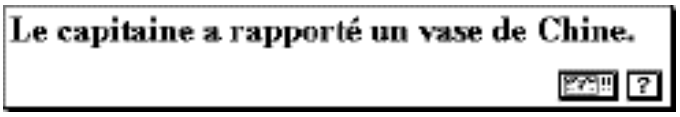

Figure 7 : l'objet a une ou plusieurs questions à poser

Lorque l'utilisateur clique sur le bouton  , une session de désambiguïsation interactive commence. Une première question apparaît (fig. 8).

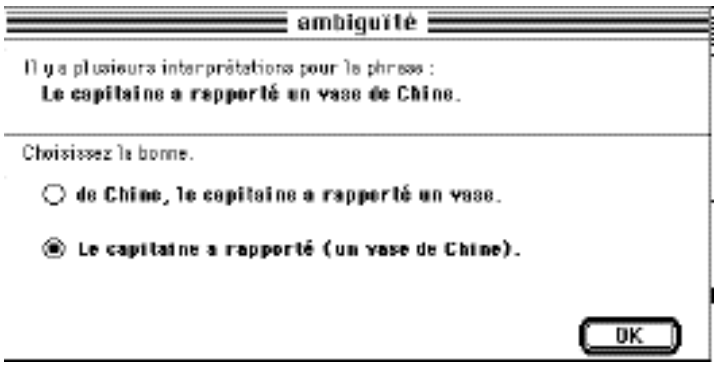

Figure 8 : problème d'attachement (histoire 2)

Avec le dialogue de la figure 8, l'utilisateur choisit l'attachement du groupe nominale "de Chine". Un second dialogue apparaît alors (fig. 9), qui permet à l'utilisateur de sélectionner le sens du mot "capitaine". Les sens des mots proviennent de Parax, une maquette de base de données lexicales multilingue [10].

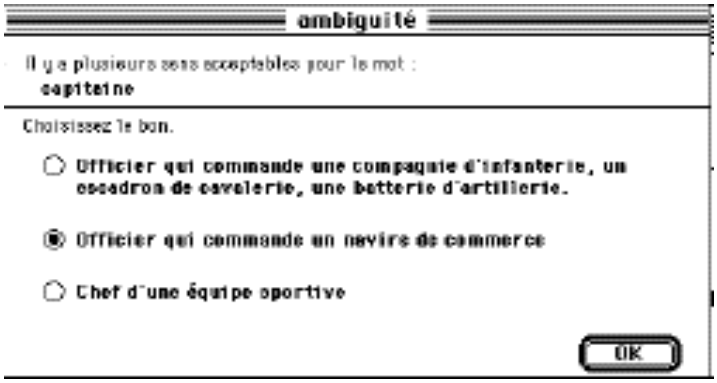

Figure 9 : sélection du sens de capitaine (histoire 2)

Lorsque le processus de désambiguïsation prend fin, l'utilisateur peut voir la forme annotée du texte. Cette forme montre, pour l'analyse finalement retenue, les groupes syntagmatiques avec leur fonction syntaxique, ainsi que la classe syntaxique de chaque occurrence.

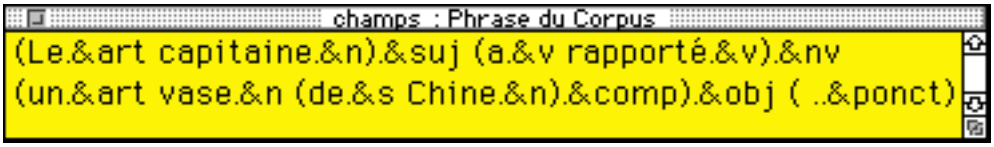

Figure 10 : la forme annotée de l'analyse retenue par l'utilisateur

Ces annotations doivent permettre à l'utilisateur de comprendre la structure produite par l'ananlyseur. Nous pensons que les utilisateurs expérimentés voudront diminuer le nombre des dialogues de désambiguïsation en insérant eux-mêmes de telles marques.

Pour vérifier les traductions produites par le système dans chacune des langues cibles, l'utilisateur peut demander une rétrotraduction. Pour la seconde interprétation de la phrase exemple et pour l'allemand, il va obtenir :

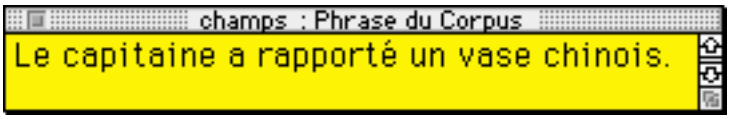

Figure 11 : une rétrotraduction

Finalement, le système va produire une carte d'histoires traduites.

Figure 12 : traduction des deux histoires en allemand

3. Quelques autres aspects

Nous donnons maintenant quelques détails supplémentaires à propos de l'interface, des techniques d'implémentation et de la méthode de désambiguïsation.

3.1. Les autres traitements

Lorqu'il a choisi d'utiliser LIDIA dans les préférences HyperCard, l'utilisateur peut définir les préférences LIDIA pour la pile courante.

a. Les préférences LIDIA

Il y a quatre profils de préférences, profil de la tâche, des serveurs, de l'utilisateur, des ressources lexicales (fig 13) Le panneau du profil de la tâche (fig. 13) permet à l'utilisateur de choisir les langues cibles, l'unité de traduction (sélection, carte, pile), ainsi que les traitements à effectuer : vérification orthographique et stylistique, standardisation terminologique, usage des termes spéciaux figés, typage textuel et traduction.

Le profil de l'utilisateur concerne le type de rétroaction et le niveau de dialogue. Le profil lexical permet de sélectionner le correcteur orthographique ainsi que les dictionnaires et thesaurus personnels.

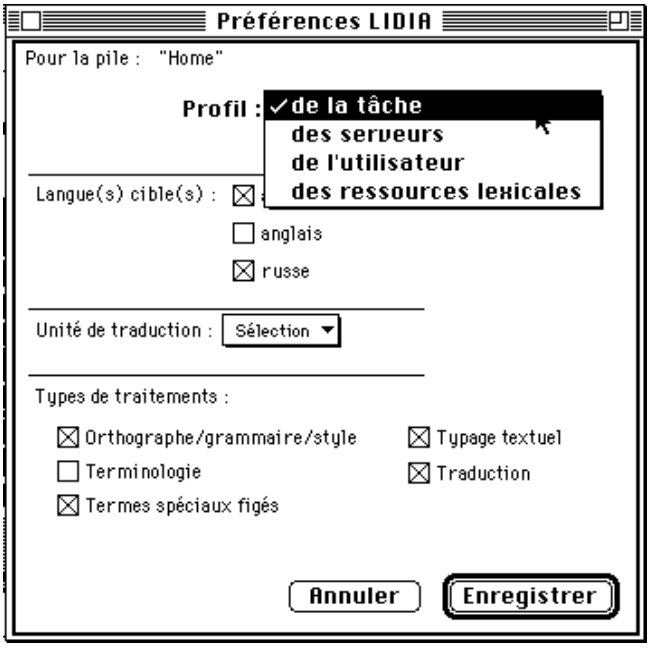

Figure 13 : les préférences LIDIA, profil de la tâche

b. Les outils d'interaction

Lorsque les préférences ont été définies, l'utilisateur dispose de menus et d'une palette pour interagir avec LIDIA.

L'interaction se fait au moyen du menu **LIDIA** (fig. 14), du menu **Messages**, d'une palette (fig. 15), de boutons de rétro-action (fig. 1) et de fenêtres flottantes (fig. 1).

Le menu ci-dessous montre les huit actions possibles.

Figure 14 : le menu LIDIA

L'utilisateur peut aussi accéder aux traitements les plus fréquents avec une palette flottante (fig. 15). Les outils de la première ligne sont les outils LIDIA (sélection d'un objet à traiter, voir l'état d'avancement des traitements, voir les annotations, voir les rétrotraductions). Les outils de la seconde ligne permettent de se déplacer dans la pile courante.

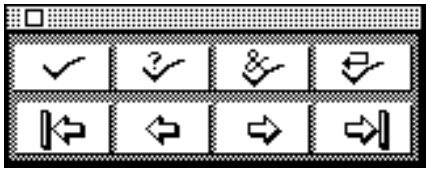

Figure 15 : la palette LIDIA

Le processus de traduction se compose de deux grandes étapes, la standardisation et la clarification. Nous avons vu l'étape de clarification pendant la démonstration, voyons le rôle de la phase de standardisation.

c. La standardisation

Le processus de standardisation doit permettre de réduire le degré d'ambiguïté des textes à analyser, et ainsi, diminuer le nombre des questions de clarification. Après l'étape de clarification,

- le texte doit être correctement orthographié, dans cette optique, nous sommes en contact avec la société Machina Sapiens afin d'intégrer le moteur de leur correcteur orthographique 101 ;
- la terminologie utilisée doit respecter des normes. Celles-ci peuvent être imposées par des habitudes locales d'écriture, par le domaine lui-même, ou par un effort de standardisation — dans une entreprise par exemple ;
- l'utilisation des tournures et syntagmes figés spéciaux ("Activer les bulles d'aides" à traduire en anglais par "Show balloons") doit être certifiée afin que la traduction soit produite correctement ;
- les champs textuels et les boutons doivent avoir reçu un étiquetage en type de texte. L'utilisateur devrait affecter un *style d'énoncé* pour les phrases ou d'autres énoncés traduisibles individuellement (titres, éléments homogènes d'une énumération…), ou un *genre de texte* pour les segments textuels plus longs. Avec ces étiquettes, l'analyseur choisira seulement les règles pertinentes pour le type d'énoncé à analyser et ainsi, le nombre d'interprétations de chaque unité de traduction sera réduit.

3.2. L'implémentation

L'implémentation est caractérisée par la mise en œuvre d'une architecture distribuée, l'utilisation d'une approche tableau blanc pour le partage des données et l'utilisation de langages à objets.

Trois machines physiques (fig. 16) sont impliquées dans le processus de traduction d'une pile.

a. Architecture distribuée

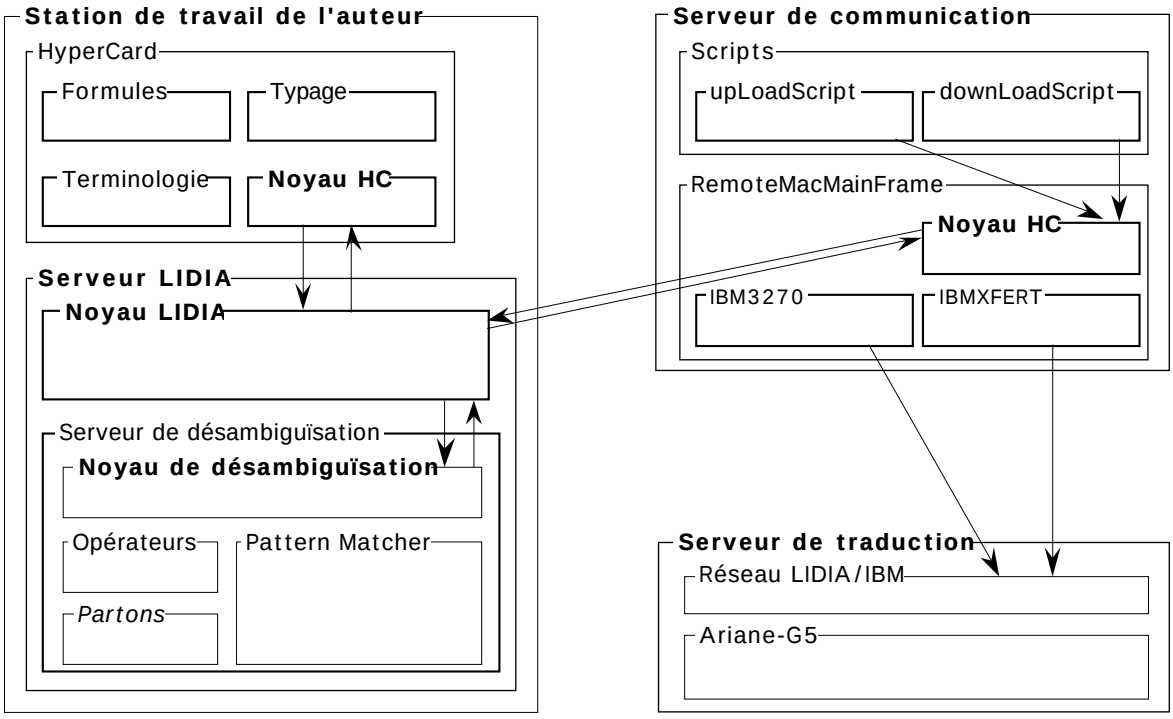

Figure 16: l'architecture de LIDIA-1

Sur la station de travail du rédacteur, le noyau HyperCard échange des messages avec le noyau LIDIA qui organise le processus de traduction de chaque objet. Le noyau LIDIA envoie des requêtes de traitement au serveur de traduction via un serveur de communication. Le noyau LIDIA communique aussi avec le serveur de désambiguïsation.

b. Approche “tableau blanc”

Pour chaque objet à traduire, le noyau LIDIA crée un objet miroir (un fichier texte) dans lequel sont stockées toutes les informations requises par le processus de traduction et nécessaires à la construction de la pile cible. Nous distinguons des informations statiques et des informations dynamiques. Les informations statiques sont les informations attachées à chaque objet par HyperCard, elles sont nécessaires pour construire la pile cible. Les informations dynamiques sont utilisées par LIDIA pour traduire le contenu d’un objet.

Ces fichiers miroirs peuvent être considérés commes des “tableaux blancs” définis en [11]. Au contraire d’un tableau noir, on accède au tableau blanc à travers un coordinateur (le noyau LIDIA), et pas par les composants (noyau de désambiguïsation et RemoteMacMainFrame). Le plus grand avantage de cette architecture est de permettre une intégration aisée de nouveaux composants déjà existants sans avoir à les modifier. Le coordinateur offre une vision des données du tableau blanc adaptées à chaque composant.

c. Programmation avec des objets

À l’exception du linguiciel, tous les autres composants sont implémentés en utilisant des langages à objets. Les modules de standardisation terminologique, de reconnaissance des tournures et syntagmes figés spéciaux, ainsi que le module de typage textuel sont écrits en HyperTalk, le langage de scripts de HyperCard.

Le serveur LIDIA est écrit en CLOS (MCL). Bien qu’ils soient encapsulés par le même environnement, le noyau LIDIA et le noyau de désambiguïsation communiquent par messages et peuvent ainsi être distribués.

L’utilisation de messages et de techniques de programmation par objets nous rapproche des modèles avec acteurs utilisés dans le contexte de systèmes coopérants distribués.

3.3. La désambiguïsation

Le processus de désambiguïsation est organisé autour d’un moteur de reconnaissance de patrons [12]. Pour cinq classes d’ambiguïtés parmis les huit que nous avons définies, nous utilisons des schémas de patrons. Un schéma de patrons est un ensemble de patrons qui partagent un ensemble de variables.

Pour une phrase ambiguë, on considère qu’un schéma de patrons est reconnu lorsque chacun des patrons de ce schéma se superpose sur au moins une des analyses de la phrase, que toutes les analyses sont couvertes et que les variables partagées par les patrons du schéma ont chacune la même couverture temporelle de la phrase. La figure 18 illustre le processus de filtrage.

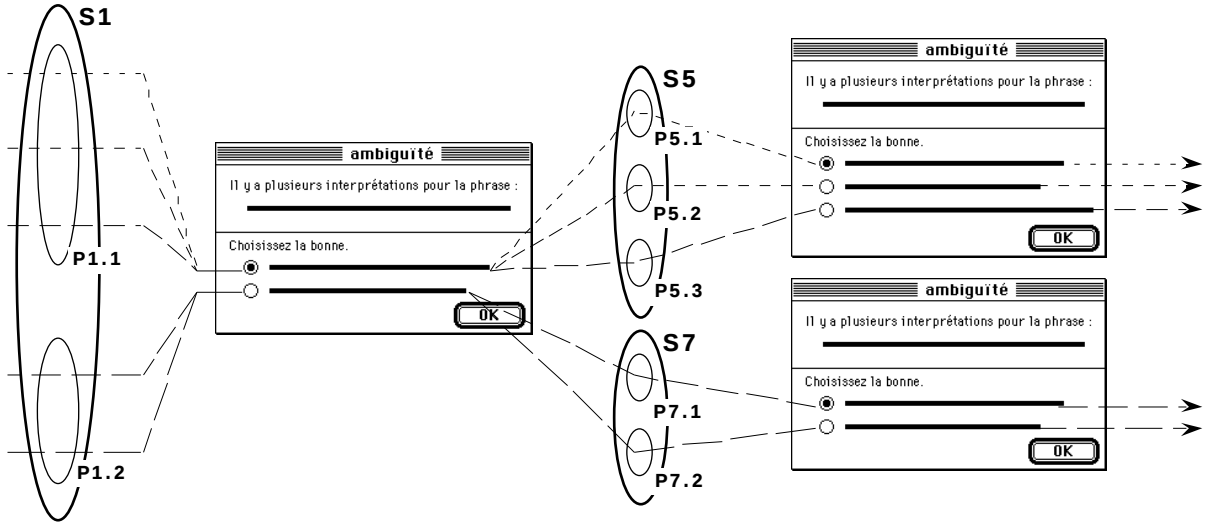

Figure 17 : processus de filtrage avec des schémas de patrons

Trois analyses sont filtrées par le patron P1.1 du schéma S1 alors que les deux autres sont filtrées par le patron P1.2. On peut alors construire un dialogue avec deux items, le premier permettant de choisir les trois premières analyses, le second, les deux autres. Pour chacune des deux familles ainsi

créées, il faut trouver un schéma de patrons qui permette de choisir une analyse. C'est le cas avec les schéma S5 et S7.

Une méthode de construction de paraphrase est associée à chaque patron. Ces méthodes sont construites au moyen d'un ensemble de treize opérateurs de base.

Par exemple, la figure 18 montre les deux arbre produits lors de l'ananlyse de la phrase "Le capitaine a rapporté un vase de Chine.".

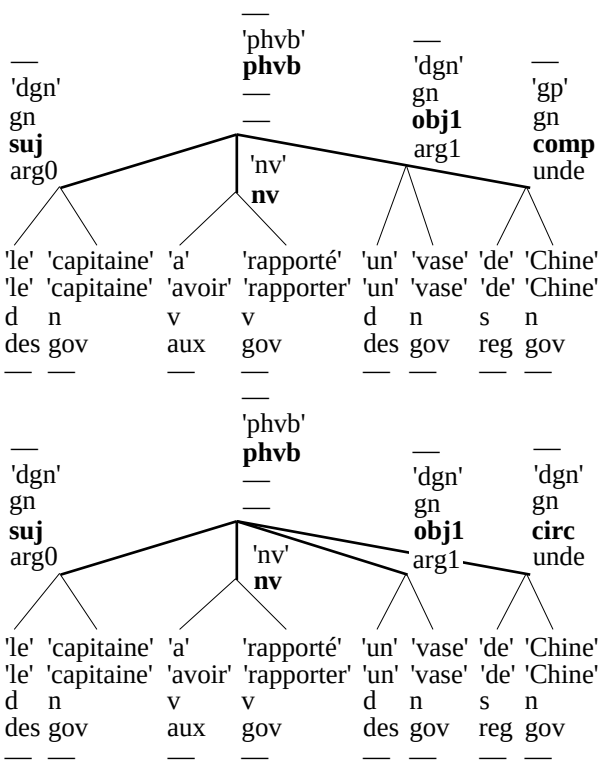

Figure 18 : une structure multisolution, multiniveau et concrète

Les deux patrons (`Patron 12` & `Patron 13`) utilisés pour reconnaître l'ambiguïté sont montrés dans la figure 199.

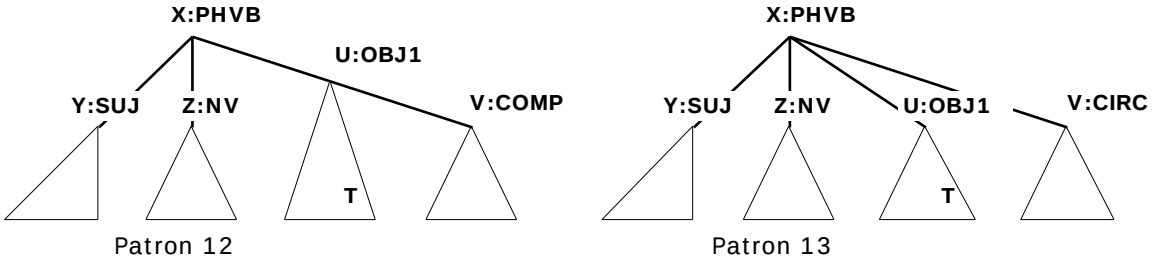

Figure 19 : 2 patrons

La méthode associée au patron 12 est :

`Texte(Y) Texte(Z) Parenthèse(Texte(T), Texte(V))`

qui permet de produire le texte suivant :

Le capitaine a rapporté (un vase de Chine.)

La méthode associée au patron 13 est :

`Texte(V) , Texte(Y) Texte(Z) Texte(T)`

qui permet de produire le texte suivant :

de Chine, le capitaine a rapporté un vase.

4. Vers un prototype opérationnel

4.1. L'interface

Pour l'instant, le point le moins avancé de la maquette est l'intégration du correcteur orthographique 101 de Machina Sapiens. Pour un prototype opérationnel, il nous faudra réussir cette intégration.

Les modules de standardisation terminologique et de reconnaissance des tournures et syntagmes figés devront utiliser un lemmatiseur pour reconnaître les usages fléchis. Il faudra aussi que les modules de standardisation soient autonomes et non plus réalisés avec HyperCard (fig. 16).

Si nous avons proposé et implémenté un module de typage textuel, nous n'avons pas pu mener assez loin l'étude des *types d'énoncés* et des *genres de textes* pour que ce module présente des informations pertinentes pour l'analyseur.

L'interface des modules concernant la terminologie, les tournures et le typage n'est qu'une première ébauche. Nous n'avons, en particulier, pas encore trouvé d'icônes satisfaisantes pour tous les boutons de rétroaction. Par contre, les icônes des curseurs LIDIA sont homogènes avec les boutons d'information, et pourraient être réutilisées telles quelles.

Dans une version ultérieure, il faudra aussi implémenter différents niveaux de dialogues adaptés aux compétences de l'utilisateur. Le type de dialogue que nous avons proposé permet seulement à l'utilisateur de choisir la bonne analyse. Un autre niveau de dialogue devrait permettre à l'utilisateur d'avoir des informations et des exemples sur les ambiguïtés à résoudre. L'utilisateur pourrait ainsi apprendre à rédiger plus clairement ou apprendre à insérer des marques de désambiguïsation.

4.2. Les techniques d'implémentation

L'implémentation que nous avons réalisée permet de proposer les bases d'une architecture logicielle pour la TAFD, avec trois mots-clés : intégration, distribution et extensibilité.

La mise en œuvre de quatre serveurs qui communiquent par messages et fichiers permet d'intégrer des outils qui travaillent dans différents environnements matériels et/ou logiciels. La gestion des messages (acheminement et contrôle des files d'attente) est prise en charge par le système d'exploitation et permet de se concentrer sur la réalisation des services.

Le serveur de communication nous a permis d'utiliser un outil très rudimentaire de communication avec le serveur de traduction, sans que la performance du système global en soit affectée. Pour la "TAFD à la maison", un serveur de communication adéquat piloterait un modem (autonome ou celui du minitel) pour travailler non plus avec un serveur de traduction, mais avec un serveur LIDIA distribué. Avec une telle architecture, la machine de Monsieur Toutlemonde sera suffisante pour servir de station de rédaction.

Chaque fonctionnalité du système peut être facilement modifiée puisqu'elle est parfaitement identifiée et localisée. On peut changer définitivement une fonctionnalité en remplaçant son code, ou provisoirement en modifiant, dans la fonction d'installation du message concerné, le nom de la procédure de gestion. L'ajout d'une nouvelle fonctionnalité passe par la création et l'installation d'un nouveau message et de procédures de réponse associées. Il suffit d'insérer dans le code existant l'appel de la nouvelle fonctionnalité et la gestion du résultat qu'elle produit.

4.3. Les outils d'implémentation

Les dictionnaires utilisés par le liguiciel Ariane-G5 sont construits à partir de Parax [10]. Pour un prototype, nous avons besoin d'un outil de construction de base lexicale multilingue plus puissant que Parax. Cet outil est aussi décrit dans [10].

Le linguiciel a été développé avec Ariane-G5 qui a été conçu pour permettre une programmation heuristique dans le contexte de sous-langages. Nous avons prévu de travailler sur de nouveaux langages spécialisés pour la programmation linguistique qui permettraient de faire de la "programmation ambiguë" [13].

4.4. Le processus de désambiguïsation

Nous savions bien que nous ne trouverions pas, pour chaque classe d'ambiguïtés que nous avons définie, une méthode unique de résolution. En gardant à l'esprit le type de dialogue que nous

voulions produire, nous avons examiné une grande quantité de configurations et nous sommes parvenus à définir neuf types de problèmes.

La définition d'une stratégie qui permet d'organiser le processus de désambiguïsation, l'utilisation de patrons et de méthodes réalisées avec un ensemble fixé d'opérateurs de base rend le processus de désambiguïsation modifiable très facilement. C'est pourquoi nous pensons que le développement d'un environnement de description de processus de désambiguïsation serait une bonne chose.

Un tel environnement intégrerait trois modules : un module de description de patrons et de schémas de patrons, un module de définition de la méthode de production du dialogue associée à chaque patron, et finalement un module de description de stratégies de désambiguïsation.

Conclusion

L'implémentation de la maquette LIDIA-1, première expérience concrète en vue de la "TAFD pour tous", s'est développée "en largeur" sur certains points, et "en profondeur" sur d'autres. Ainsi, nous avons prototypé toutes les fonctionnalités prévues, mais n'avons implémenté complètement que le processus de désambiguïsation, traité en profondeur. Cela était inévitable au niveau d'une maquette que nous avons voulu extensible grâce à l'architecture utilisée.

Il était très important que tous les aspects aient été abordés, car les expériences passées montrent qu'une conception globale est nécessaire pour qu'un système de TAO réussisse. En effet, et nous l'avons bien vu en travaillant sur LIDIA-1, des objectifs de nature ergonomique peuvent contraindre certains choix informatiques et linguistiques, et réciproquement pour les objectifs informatiques et linguistiques.

L'idée d'intégrer des composants de clarification interactive dans les systèmes de traitement des langues naturelles semble mise en avant par une communauté scientifique.

En ce qui concerne la TAO, on peut citer les travaux du LATL à l'université de Genève [14], ceux du département d'informatique de la "Tokyo University of Agriculture and Technology" [15], ainsi que les travaux qui continuent sur JETS [7]. Pour les systèmes où la parole intervient, la clarification interactive est aussi une nécessité comme cela est montré par [16] et proposé dans [17] et [18].

En ce qui concerne les étapes futures, nous avons commencé d'étudier avec ATR-ITL la désambiguïsation interactive multimodale [19], dans un cadre plus large que celui de LIDIA-1. Nous pensons aussi avoir les moyens de développer un prototype de plus large envergure dans les prochaines années.

Bibliographie

[1] **Boitet, C.**, (1990). *Towards Personnal MT : general design, dialogue structure, potential role of speech.* Proc. Coling-90. Helsinki. 20-25 Août 1990,. vol. **3/3** : pp. 30-35.

[2] **Kay, M.**, (1973). *The MIND system.* in Courant Computer Science Symposium 8: Natural Language Processing. Algorithmics Press, Inc. New York. pp. 155-188.

[3] **Melby, A. K.**, (1981). *Translators and Machines - Can they cooperate ?* in META. vol. **26**(1) : pp. 23-34.

[4] **Wood, M. M.**, (1989). *Japanese for speakers of English: The UMIST/Sheffield Machine Translation Project.* in Recent Developments and Applications of Natural Language Processing. Kogan Page Limited. London. pp. 56-64.

[5] **Sadler, V.**, (1989). *Working with analogical semantics: Disambiguation techniques in DLT.* Floris Publications. Dordrecht, Holland. 256 p.

[6] **Rimon, M., *et al.*** , (1991). *Advances in Machine Translation Research in IBM.* Proc. Machine Translation Summit III. Washington, D.C. 1-4 juillet 1991,. vol. **1/1** : pp. 11-18.

[7] **Tsutsumi, T., *et al.***, (1993). *Example-Based Approach to Machine Translation.* Proc. Premières journées franco-japonaise sur la traduction assistée par ordinateur. Ambassade de France au Japon, Tokyo, Japon. 15-16 mars 1993,. vol. **1/1** : pp. 161-169.

[8] **Goodman, K. & Nirenburg, S.** (ed.), (1991). The KBMT Project: A case study in knowledge-based machine translation. Morgan Kaufmann. San Mateo, California. pp. 331.

[9] **Boitet, C. & Blanchon, H.**, (1993). *Dialogue-based MT for monolingual authors and the LIDIA project.* Proc. NLPRS'93. Fukuoka, Japon. 6-7 décembre 1993, : pp. 208-222.

[10] **Sérasset, G. & Blanc, É.**, (1993). *Une approche par acception pour les bases lexicales multilingues.* Proc. T–TA–TAO 93. Montréal, Canada. 30 septembre-2 octobre 1993 : 15p. À paraître.

[11] **Seligman, M. & Boitet, C.**, (1994). *A "whiteboard" architecture for automatic speech translation.* Proc. International Symposium on Spocken Dialogue. Waseda University, Tokyo. 1-12 novembre 1993, 4p.

[12] **Blanchon, H.**, (1992). *A Solution to the Problem of Interactive Disambiguation.* Proc. Coling-92. Nantes, France. 23-28 juillet 1992,. vol. **4/4** : pp. 1233-1238.

[13] **Boitet, C.**, (1993). *Crucial Open Problems in Machine Translation and Interpretation.* Proc. Symposium on Natural Language Processing in Thailand. Bangkok. 17-21 March 1993 : pp. 1—29.

[14] **Wehrli, É.**, (1993). *Vers un système de traduction interactif.* in La traductique. Les presses de l'Université de Montréal, AUPELF/UREF. pp. 423-432.

[15] **Yamaguchi, M., *et al.***, (1993). *An Interactive Method for Semantic Disambiguation in Sentences by Selecting Examples.* Proc. NLPRS'93. Fukuoka, Japon. 6-7 décembre 1993 : pp. 208-222.

[16] **Frankish, C., *et al.***, (1992). *Decline in accuracy of automatic speech recognition as a function of time on task: fatigue or voice drift?* in International Journal of Man-Machine Studies. vol. **36**(6) : pp. 797-816.

[17] **Ainsworth, W. A. & Pratt, S. R.**, (1992). *Feedback strategies for error correction in speech recognition systems.* in International Journal of Man-Machine Studies. vol. **36**(6) : pp. 833-842.

[18] **Saito, H.**, (1992). *Interactive Speech Understanding.* Proc. Coling-92. Nantes, France. 23-28 juillet 1992,. vol. **3/4** : pp. 1053-1057.

[19] **Boitet, C.**, (1993). *Multimodal Interactive Disambiguation: first report on the MIDDIM project.* Rap. ATR Interpreting Telecommunications. Technical Report. Aug. 93.