Industrie Canada
Centre d'innovation en technologies de l'information (CITI)

TransSearch: un concordancier bilingue

Michel Simard simard@citi.doc.CA
George F. Foster foster@citi.doc.CA
François Perrault perrault@citi.doc.CA

Laval
octobre 1993

Ce document fait état de travaux de recherche réalisés dans le cadre des activités du Centre d'innovation en technologies de l'information (CITI), Industrie Canada. Les opinions exprimées dans ce document n'engagent que les auteurs.

Also available in English under the title: "TransSearch: A Bilingual Concordance Tool"

No catalogue Co28-1/130-1996F

No ISBN 0-662-81093-7

TransSearch: un concordancier bilingue

Michel Simard simard@citi.doc.CA

George F. Foster foster@citi.doc.CA

François Perrault perrault@citi.doc.CA

Centre d'innonvation en technologies de l'information
1575, boul. Chomedey
Laval (Québec)
CANADA H7V 2X2

Sommaire

TransSearch est un système permettant de construire et d'exploiter une *mémoire de traduction*, c'est-à-dire une base de donnée textuelle, constituée de paires de documents qui sont des traductions réciproques. Chaque paire de documents ainsi insérée dans la base de donnée passe par un processus d'*alignement*, lequel retrouve les liens existants entre les phrases des deux textes. Un utilisateur peut alors, au moyen d'un langage de requête graphique, rechercher des solutions toutes faites à des problèmes de traductions spécifiques.

1 Introduction

Avec la croissance du marché de la traduction, l'informatisation fait peu à peu son chemin dans cette industrie. Non pas dans le sens de l'automatisation du processus même de traduction, comme on aurait pu l'espérer, mais par la voie d'environnements adaptés et d'outils d'aide à la traduction. On s'est d'abord intéressé aux tâches périphériques à l'acte de traduction: traitement de texte, vérification orthographique, accès informatisé à des ressources lexicales et terminologiques, storage et transfert de documents électroniques, etc. Le PTT du CITI est un exemple classique de ce genre d'environnement [9]. On voit toutefois de plus en plus de systèmes qui sont conçus dès le départ avec la traduction en vue. Le plus connu est sans doute *TSS*, de ALPNET (anciennement ALPS), mais bien d'autres ont vu le jour depuis: on pense par exemple à des produits comme le *Trados Translator's Workbench* et *Translation Manager*[1]. Au coeur de tous ces systèmes, on retrouve un élément commun: le concept de *mémoire de traduction*.

1. *Trados Translator's Workbench* est un produit de Trados GmbH, Rotebühlstraße 87, 7000 Stuttgart 1, Allemagne; *Translation Manager* est un produit IBM; Nous ne savons pas si ALPNET distribue encore *TSS*.

Il est clair que les traductions existantes renferment plus de solutions à plus de problèmes de traduction que toute autre ressource présentement disponible. C'est pourquoi tous les services de traduction dignes de ce nom archivent plus ou moins systématiquement leur production. En effet, il n'est pas rare qu'un même client fasse traduire à des moments différents des documents distincts qui sont néanmoins presqu'identiques, ou qui renferment plusieurs passages communs. Face à un nouveau texte à traduire, un traducteur qui est en mesure de retrouver des documents apparentés, avec leur traduction, pourra s'en servir comme références terminologiques, voire "récupérer" des passages entiers.

Le problème qui se pose alors est l'accès à cette information. Si les archives sont bien tenues, même sur papier, on peut imaginer qu'il ne sera pas trop ardu de retrouver une version antérieure d'un document à traduire. C'est toutefois une autre histoire lorsqu'il s'agit de localiser des documents qui sont plus ou moins apparentés. Dans la mesure où l'on dispose d'un archivage électronique, les systèmes de recherche documentaire (*document retrieval systems*) solutionnent partiellement ce problème. Mais il reste alors au traducteur à apparier les documents ainsi trouvés à leur traduction, et à retrouver les correspondances qui existent entre les éléments de ces paires de documents.

La mémoire de traduction apporte une solution à ce problème. L'idée en est simple: archiver des traductions existantes de façon à permettre leur *réutilisabilité*. On conserve donc, sur support magnétique, non seulement les documents originaux et leur traduction, mais aussi les liens intimes qui les unissent. Ainsi, un système comme *Trados Translator's Workbench*, face à une nouvelle phrase à traduire, peut détecter si cette phrase, ou une phrase très similaire, apparaît dans sa mémoire, et en localiser instantanément la traduction, que le traducteur peut alors réutiliser ou modifier à son gré (voir [2]).

Cette idée de mémoire de traduction doit toutefois pouvoir nous mener plus loin qu'à la simple récupération de phrases. Il a été suggéré qu'un outil de *concordance bilingue*, c'est-à-dire un programme qui permette d'observer dans leur contexte "bilingue" des expressions ou des paires d'expressions, serait une ressource précieuse pour les lexicographes (voir par exemple [3]). En fait, il semble évident qu'un tel outil serait tout aussi utile pour des traducteurs. Lorsque confronté à une expression idiomatique telle que *se mettre le doigt dans l'oeil* ou une formule toute faite comme *des démêlés avec la justice*, il n'est pas rare qu'un traducteur hésite quant à une traduction appropriée: les ressources traditionelles, telles que les dictionnaires bilingues, sont souvent relativement pauvres à cet égard. Un concordancier bilingue permettrait d'extraire d'une mémoire de traduction des exemples en contexte de telles expressions, accompagnés des différentes traductions qui ont été trouvées pour ces expressions. Cette idée est développée avec plus de détails par Macklovitch dans [10].

C'est d'un tel outil qu'il est question dans les pages qui suivent. Le programme que nous décrivons, appelé TransSearch, a été conçu et mis sur pied par l'équipe de traduction assistée par ordinateur

(TAO) du CITI, et constitue le premier d'un ensemble d'outils visant à aider les traducteurs dans leur travail (voir [7]).

2 Concordances bilingues

Avant d'aborder le sujet des concordances bilingues, il est nécessaire de parler un peu des concordances "ordinaires" (qu'on appelle ici des concordances "unilingues"). Une concordance, dans le sens traditionnel du terme, est une liste ordonnée de toutes les occurrences de mot d'un texte, à l'intérieur de laquelle chaque occurrence apparaît dans une "sous-chaîne" du texte original, typiquement au milieu de celle-ci. On appelle généralement cette sous-chaîne le "contexte" de l'occurrence (d'où le synonyme anglais pour ce type de concordance: *keywords in context*, ou *KWIC*). En supposant un contexte minimal, disons cinq mots de chaque côté de l'occurrence qui nous intéresse, la concordance fera onze fois la taille du texte original. Pour un corpus de taille intéressante[2], et pour peu qu'on désire plus qu'un contexte minimal, une telle masse de texte devient extrêmement difficile à manier.

Dès lors, l'idée s'impose d'un *programme de concordance*, ou *concordancier*, qui calcule, sur demande, des sous-ensemble de la concordance "intégrale". De nombreux programmes existent qui permettent ce genre de fonctionnalité, par exemple, *lq-text*, *PAT* et *gptx*[3]. En plus de contourner les problèmes liés à la manipulation d'une telle masse de texte, ce genre de programme permet une flexibilité accrue quant à la quantité de contexte à afficher avec chaque occurrence d'un mot, et quant à la sélection des sous-ensembles de la concordance intégrale qui nous intéressent. Celle-ci se fait typiquement au moyen d'un langage de requêtes spécialisé, qui permet de décrire les mots ou expressions dont on désire examiner les contextes d'utilisation. Un tel langage favorise habituellement les requêtes de nature "linguistiques", par opposition à des requêtes "documentaires". Par exemple, lorsque l'utilisateur spécifie une suite de mots, on considérera généralement que l'ordre relatif dans lequel apparaissent ces mots a une importance (*valet de chambre*, et non *chambre de valet*). Par ailleurs, même si on permet aux mots d'une telle suite d'apparaître de façon non-contigüe dans le texte, ou même dans le désordre, on exigera en général qu'ils appartiennent au même "contexte linguistique", par exemple, à la même phrase.

L'utilisation d'un concordancier devient particulièrement intéressante à partir du moment où l'on peut effectuer des recherches en temps réel. Pour arriver à ce genre d'intéraction sur un corpus de taille arbitraire, il est nécessaire de faire appel à une structuration particulière du texte. Quelque soit la nature exacte de celle-ci, elle comprendra forcément une forme d'*indexation* du texte, qui per-

2. Dans la grande majorité des contextes d'utilisation, la concordance n'a de valeur que dans la mesure où elle est produite à partir d'une quantité de texte appréciable.

3. *lq-text* (écrit par Liam Quin) et *gptx* (écrit par François Pinard) sont des programmes du domaine publique. *PAT* est un produit de OpenText Corporation, Waterloo, Ontario.

mettra de localiser toutes les occurrences d'une chaîne de caractères donnée, dans un laps de temps qui soit proportionnel au nombre d'occurrences, plutôt qu'à la taille du corpus.

La concordance bilingue peut être définie dans des termes similaires à ceux qu'on a employé pour décrire la concordance unilingue: une liste ordonnée de toutes les occurrences de mot d'une paire de texte qui sont des traductions réciproques, à l'intérieur de laquelle chaque occurrence est accompagnée de son contexte, et de la traduction de celui-ci dans l'autre langue. Inutile de dire que les problèmes liés à la manipulation d'une concordance unilingue sont multipliés par deux lorsqu'on parle de concordance bilingue. La démonstration de l'utilité d'un programme qui calcule automatiquement des sous-ensemble de cette concordance n'est donc pas à faire.

La mise sur pieds d'un concordancier bilingue est plus complexe que celle d'un concordancier unilingue. D'abord, il y a le problème du "contexte bilingue". Le concordancier unilingue affiche soit un nombre fixe de caractères ou de mots de part et d'autre de l'occurrence du mot qui nous intéresse, soit un contexte de nature plus "linguistique", tel que la phrase ou le paragraphe dans lequel cette occurrence apparaît. Si on désire en plus afficher la traduction de ce contexte, il faut être en mesure de la localiser exactement, ce qui n'est pas toujours possible pour une sous-chaîne quelconque du texte. En général, on pourra se contenter d'une approximation, et à cet égard, il est préférable que le concordancier bilingue pèche par excès que l'inverse, en autant que l'utilisateur ne soit pas submergé par l'information. Par ailleurs, on peut démontrer que le laps de temps minimum requis pour localiser la traduction d'un contexte donné, même de façon approximative, est proportionnel à la taille des deux textes, de telle sorte qu'il n'est pas possible d'effectuer cette opération en temps réel. Il faut donc pré-calculer les correspondances et conserver cette information avec le texte, d'une manière qui assure des accès rapide. C'est cette structuration de l'information qui donne lieu à la notion de *mémoire de traduction*.

Enfin, le langage de requêtes du concordancier bilingue doit permettre à l'utilisateur de tirer profit de cette structuration "bilingue" du texte, en lui offrant, par exemple, la possibilité de spécifier des paires d'expressions de langues différentes. Grâce à cette particularité du langage de requête, il devient possible d'examiner dans quels contextes une certaine expression se traduit par une autre.

Dans les sections qui suivent, nous présentons d'abord notre modèle de mémoire de traduction, puis les mécanismes qui permettent d'en extraire des concordances bilingues, et finalement l'interface-utilisateur qui donne accès à ces mécanismes.

3 La mémoire de traduction

Nous appelons *TransBase* notre structure de mémoire de traduction. Une structure TransBase, qui est spécifique à une paire de langue particulière, contient un ensemble de documents, constitués chacun d'une paire de textes, soit un pour chaque langue. Lors de son insertion dans la structure,

chaque document est soumis à une *analyse de traduction*, qui consiste en fait en deux analyses unilingues et une analyse bilingue. C'est le résultat de cette analyse qui est storé dans la structure. Celle-ci prend la forme d'une base de donnée, constituée de trois composantes interreliées, correspondant aux trois analyses effectuées. Ce processus est illustré à la Figure 1.

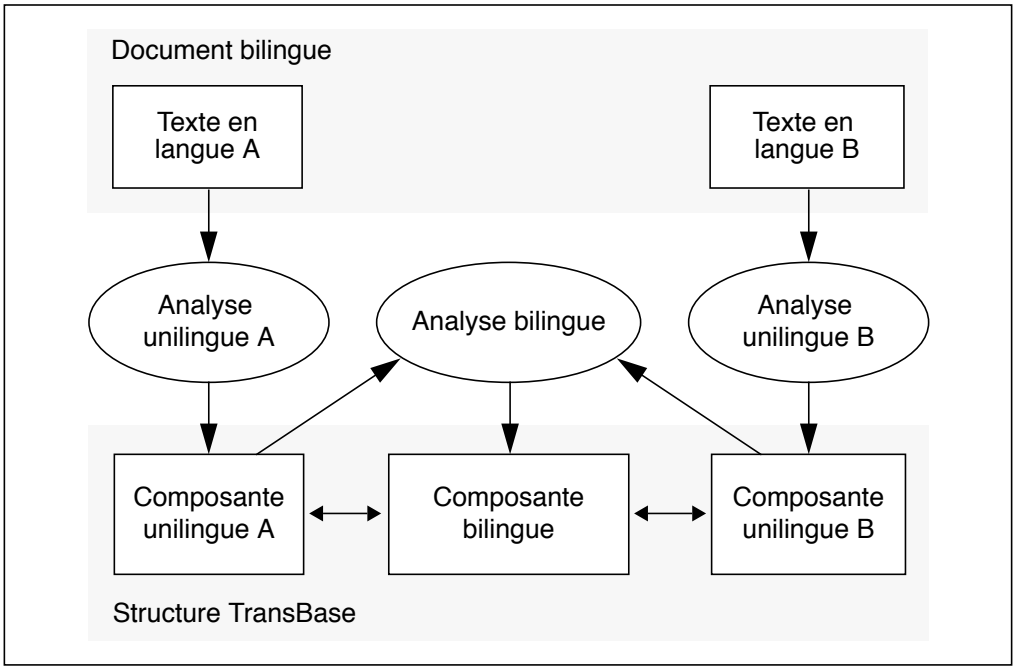

Figure 1: Insertion d'un document dans la structure TransBase

Nous commençons par décrire en quoi consiste l'analyse de traduction, pour ensuite présenter l'organisation de l'information ainsi obtenue dans la structure TransBase.

3.1 Analyse de traduction

L'analyse de traduction à laquelle est soumis chaque document qu'on insère dans la structure TransBase procède en quatre étapes distinctes: *prétraitement*, *atomisation*, *segmentation* et *alignement*. Les trois premières étapes constituent ensemble ce qu'on pourrait appeler une *analyse de texte unilingue*, qui est appliquée séparément à chacun des deux textes qui forment un document. La quatrième étape utilise le résultat de ces deux analyses unilingues pour effectuer une *analyse bilingue*, dont le but est de retracer les liens intimes qui doivent exister entre deux textes qui sont des traductions réciproques.

L'objectif du *prétraitement* est de normaliser le contenu d'un fichier-texte en préparation de l'étape suivante (l'atomisation). Essentiellement, il s'agit de ramener le contenu de ce fichier à une suite de caractères standards (ISO-Latin 1), de s'assurer que le texte adhère aux conventions typographiques normales pour la langue concernée, et d'éliminer les annotations codées et les codes de

formattages inconnus des traitements subséquents. Les choses étant ce qu'elles sont, chaque format d'encodage requiert un prétraitement différent, et il est donc nécessaire de créer un nouveau module de prétraitement pour chaque nouveau format rencontré.

L'*atomisation* a pour but d'identifier et d'isoler les mots, les signes de ponctuation, les expressions numériques ou symboliques et les annotations codées du texte. Outre que cette information soit requise pour les étapes ultérieures de segmentation et d'alignement, elle est essentielle pour l'implantation de mécanismes de recherche axées sur les mots plutôt que sur des chaînes de caractères: lors de l'insertion d'un document dans la mémoire de traduction, on procède à une *indexation* des mots du texte, dont le but est de rendre possible l'exécution de recherches en temps réel. L'atomisation ne se limite donc pas aux frontières habituelles (espacement, retours de chariot, ponctuation, etc.), mais utilise aussi des règles qui sont spécifiques à chaque langue, pour segmenter des chaînes de caractères qui sont en fait des *aggrégats* de plusieurs mots. Ainsi, *jusqu'alors* devient *jusqu'* + *alors*, et *women's-rights* devient *women* + *'s* + - + *rights* (le caractère '+' est utilisé ici simplement pour marquer l'endroit où une segmentation a été faite). Par ailleurs, il incombe au processus d'atomisation de lever l'ambigüité qui réside sur la nature des 'points': marqueur de fin de phrase ou marqueur d'abbréviation? Cette désambiguisation se fait en comparant le contexte des points avec des listes d'abbréviations standards et des patrons d'abbréviations courants. Le produit de cette étape est une liste ordonnée d'atomes, augmentés d'étiquettes qui en spécifient le type: mot, ponctuation, expression numérique, etc.

La *segmentation* recrée, à partir de la liste d'atomes obtenue dans l'étape précédente, les phrases, les paragraphes et les principales divisions du texte. En fait, les frontières entre ces objets ont généralement déjà été reconnus et identifiés lors des étapes précédentes. Par exemple, le processus de prétraitement pourra reconnaître des codes spécifiques à un certain format, qui marquent le début d'un chapitre ou d'une section, et les remplacer par des annotations qui seront reconnues lors des étapes subséquentes. Par ailleurs, le processus d'atomisation reconnaît une suite de lignes blanches comme un marqueur de fin de paragraphe, et certains signes de ponctuations comme des marqueurs de fin de phrase, et les étiquette explicitement comme tel. Le travail de la segmentation consiste alors essentiellement à rendre explicite la structure du texte qui découle de l'existence de ces frontières.

Finalement, l'*alignement* désigne le procédé par lequel nous retraçons les liens qui existent entre un texte source et sa traduction. Ce procédé consiste à segmenter parallèlement les deux textes d'une façon telle que le n^{e} segment d'une version du texte soit la traduction du n^{e} segment de l'autre. Chaque paire de segments ainsi alignés constitue un *couple*. Dépendant des unités sur lesquelles la segmentation est faite (mots, phrases, paragraphes, etc.), on voudra que celle-ci soit *maximale*, c'est-à-dire que chaque segment contienne le plus petit nombre d'unités possible. La Figure 2 illustre l'alignement d'une paire de paragraphes au niveau des phrases.

Couple	Version anglaise	Version française
1	*The crisis our farmers are in right now will affect all of us at a certain point in time.*	*La crise que vivent en ce moment nos agriculteurs se répercutera sur tous et chacun de nous à un certain moment.*
2	*We are all consumers and we all need a strong and healthy agricultural sector.*	*Nous sommes des consommateurs.*
		Nous avons tous besoin d'une agriculture saine et forte.
3	*I am glad that the Hon. Member for Algoma (Mr. Foster) mentioned figures in his remarks.*	*Heureusement que le député d'Algoma (M. Foster) a mentionné des chiffres dans ses remarques, sans cela ce gouvernement s'en serait sorti en douce encore une fois.*
	Otherwise , the Government might have eluded the problem once again .	
4	*The Hon. Member for Algoma suggested Tuesday night that the Government had to take a clear position and make a commitment to assist our farmers before it is too late.*	*Le député d'Algoma suggérait mardi soir qu'il fallait que le gouvernement se prononce clairement et s'engage à aider nos agriculteurs avant qu'il ne soit trop tard .*

Figure 2: Alignement d'une paire de paragraphes anglais et français

Les alignements de TransBase sont constitués de segmentations en phrases, et sont effectués automatiquement à partir d'une méthode inspirée du programme de Gale et Church [6]. Pour une paire de textes donnée, ce programme considère tous les alignements qu'il est possible de produire à partir d'un nombre restreint de *schèmes de traduction*. Ces derniers représentent les différentes "stratégies" qu'un traducteur emploie lorsqu'il traduit un texte: traduire une phrase par une phrase, scinder une phrase du texte-source en deux phrases dans la traduction, combiner deux phrases du texte-source en une seule dans la traduction, etc. Chaque couple de chaque alignement ainsi produit reçoit ensuite une cote indiquant le degré d'adéquation des deux segments qui le composent. Enfin, le programme détermine, à partir de ces cotes et au moyen d'un modèle de programmation dynamique, la meilleure séquence de couples aboutissant à un alignement valide.

La fonction de cotation de Gale et Church repose sur un modèle stochastique, et produit une approximation de la probabilité que deux segments soient des traductions réciproques. On observe en effet que le rapport des longueurs (en nombre caractères) de segments qui sont des traductions réciproques suit une distribution normale. En combinant cette observation avec la distribution observée des différents schèmes de traduction (Tableau 1), on arrive à modéliser mathématiquement le lien qui existe entre un texte et sa traduction. En dépit de son apparente simplicité, cette méthode produit d'excellents résultats lorsque la traduction ne dévie pas trop du texte original.

Notre méthode d'alignement est essentiellement identique à celle décrite plus haut, sauf que nous en avons amélioré la robustesse dans les situations difficiles en misant sur les *mots apparentés*, i.e. les paires de mots de langues différentes qui, le plus souvent de par une étymologie commune, partagent à la fois des propriétés phonologiques ou orthographiques et des propriétés sémantiques, de telles sorte qu'ils sont fréquemment utilisés comme traductions réciproques. On pense par exemple au mot français *génération* et à l'anglais *generation*, ou à *raison* et *reason*. Ces mots abon-

Schème de traduction	*Distribution observée*
1-1	*0.89*
1-0 ou 0-1	*0.0099*
1-2 ou 2-1	*0.089*
2-2	*0.011*

Tableau 1: Distribution observée des différents schèmes de traduction. Source: [6].

dent entre ces deux langues, comme entre bien d'autres paires de langues d'ailleurs. L'analyse statistique de textes traduits révèle que le nombre de paires de mots apparentés dans une paire de segments qui sont des traductions réciproques suit approximativement une loi binômiale *B(n, p)*, où *n* est le nombre total de mots dans les deux phrases, et $p = 0{,}3$. Par ailleurs, si l'on observe des paires de segments qui, bien qu'issus de paires de textes qui sont des traductions réciproques, ne constituent pas eux-même des traductions réciproques, le nombre de paires de mots apparentés suit aussi une loi *B(n, p)*, sauf qu'ici, *p* prend pour valeur 0,1. On peut en déduire que le nombre de paires de mots apparentés est en moyenne trois fois plus élevé dans une paire de segments "bien alignée" que dans une paire "mal alignée".

Dans les situations difficiles, il arrive souvent que le programme de Gale et Church rencontre plusieurs alignements dont les cotes globales sont relativement proches les unes des autres. Plutôt que d'effectuer un choix difficile parmi ces alignements, notre programme conserve l'ensemble des meilleurs alignements, qui est alors soumis à un deuxième traitement. Celui-ci commence par une estimation du nombre de paires de mots apparentés apparaissant dans chaque couple de chaque alignement. Dans la pratique, on observe qu'on obtient une bonne approximation de ce nombre en appariant les mots dont les quatre premiers caractères sont identiques. On attribue ensuite une nouvelle cote à chaque couple, qu'on obtient en comparant la probabilité d'observer ce nombre sous l'hypothèse que les deux segments sont des traductions réciproques, et sous l'hypothèse qu'il s'agit de deux segments sans lien. À l'issue de ce deuxième traitement, on ne conserve que l'alignement qui a obtenu la plus forte cote globale.

Lorsqu'appliquée au Journal des débats de la Chambre des communes, notre programme d'alignement a obtenu un taux de succès supérieur à 98 p. cent. Il est par ailleurs relativement efficace, puisqu'il permet d'aligner plus de 80 000 mots de chaque langue à la minute, sur un ordinateur SPARCstation 1+. On en trouvera une description détaillée, de même qu'une analyse des résultats qu'il produit, dans [11].

3.2 Structure

Conceptuellement, la structure *TransBase* peut être vue comme une base de donnée constituée de six types d'entités, qu'on peut en fait regrouper en trois composantes: une composante unilingue

pour chaque langue, formée par les entités *Atome* et *Forme*, et une composante bilingue regroupant les entités *Document* et *Couple*. La Figure 3 présente une vue éclatée de la structure TransBase telle qu'elle apparaissait dans la Figure 1, sous la forme d'un diagramme entités-relations simplifié (les attributs sont inscrits à l'intérieur des rectangles des entités auxquels ils se rapportent).

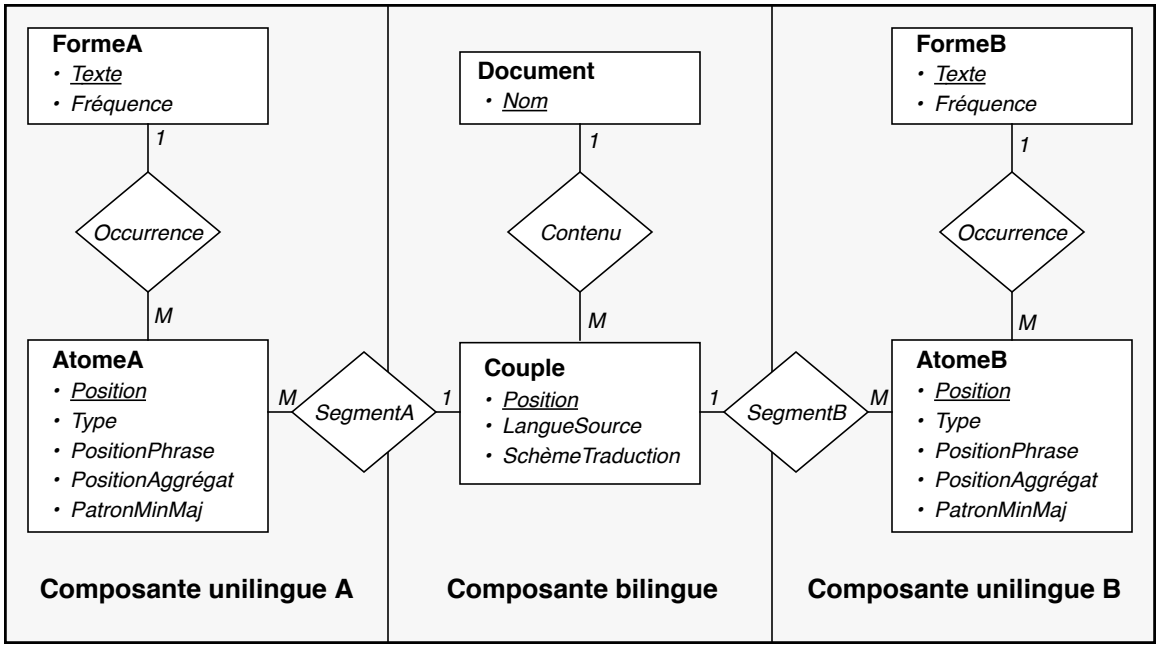

Figure 3: Structure TransBase sous forme de diagramme entités-relations

La composante unilingue A constitue une représentation de l'ensemble des textes de langue A (c'est-à-dire des versions en langue A des documents) que contient la mémoire de traduction. À chaque atome identifié lors de l'étape d'atomisation (voir la section 3.1) correspond une entité *AtomeA* dans cette composante. L'ordre d'apparition des atomes dans le texte est conservé dans la base de donnée au moyen de l'attribut *Position*. Cet attribut prend une valeur numérique entière qui reflète la position relative de l'atome: l'atome dont la position est *n* suit immédiatement l'atome *n-1* et précède immédiatement l'atome *n+1*. De plus, Les valeurs de *Position* sont attribuées de façon continue lors de la création de la base de donnée, de telle sorte que cet attribut identifie chaque atome de langue A de façon unique. Les attributs *Type*, *PositionPhrase* et *PositionAggrégat* enregistrent respectivement le type de l'atome (tel que déterminé lors de l'atomisation), sa position à l'intérieur de la phrase dans lequel il apparaît (au début, au milieu ou à la fin, tel que déterminé lors de la segmentation), et sa position au sein de l'*aggrégat* dont il fait partie, s'il y a lieu (début, milieu ou fin). Le rôle de l'attribut *PatronMinMaj* est décrit plus bas.

La *forme* de chaque atome de langue A, c'est-à-dire la chaîne de caractères qui représente celui-ci dans le texte, constitue une entité distincte *FormeA*. La relation *Occurrence*, qui lie les entités

AtomeA aux entités *FormeA*, est du type un-à-plusieurs, puisque si chaque atome n'a qu'une forme, à une forme peuvent correspondre plusieurs atomes. Ce nombre d'occurrences est conservé dans l'attribut *Fréquence*. La chaîne de caractères elle-même est storée dans l'attribut *Texte*, sous une forme *normalisée*, c'est-à-dire où l'on a fait abstraction de l'usage des majuscules et des minuscules, sauf dans les cas atypiques. On store dans l'attribut *PatronMinMaj* de chaque entité *AtomeA* l'information nécessaire pour reproduire la structure originale, en terme de majuscules et de minuscules, de cette occurrence de la forme correspondante.

La composante unilingue B joue un rôle analogue à celui de la composante unilingue A, pour les textes de langue B.

La composante bilingue contient une entité *Document* pour chaque document bilingue inséré dans la base de donnée. À cette entité n'est associé qu'un seul attribut, soit le *Nom* du document. Toutefois, dans le contexte d'une utilisation "corporative" de la structure TransBase, on lui attacherait probablement beaucoup plus d'information: le nom du ou des auteurs du document, les noms des traducteurs, des réviseurs, du client, la date de soumission, etc. Pour nos fins, toutefois, cette information minimale est suffisante.

Finalement, ce sont les entités *Couple* qui constituent le coeur de la composante bilingue de la base de donnée. On comptera une entité de ce type pour chaque paire de segments produit par le processus d'alignement. L'attribut *Position* y joue le même rôle que pour les entités *AtomeA* et *AtomeB*, c'est-à-dire qu'il est à la fois indicateur de la position relative de la paire de segments dans le document où ils apparaissent, et identificateur unique pour le couple. L'attribut *SchèmeTraduction* spécifie le nombre de phrases qu'on retrouve dans chaque segment du couple, c'est-à-dire la "stratégie" particulière que le traducteur a employé lors de la traduction de ce segment. Quant à l'attribut *LangueSource*, il spécifie lequel des deux segments (langue A ou langue B) est l'original et lequel est une traduction. On pourrait croire que cette information sera la même pour tous les couples provenant d'un même document. En réalité, il existe des documents où ce n'est pas le cas. Le Journal des débats de la Chambre des communes (mieux connu sous le nom de "Hansard") en est un exemple. Par ailleur, on peut supposer que la langue source ne changera pas à l'intérieur d'un couple, même si celui-ci contient plus d'une phrase de chaque langue.

Ce sont les relations *SegmentA* et *SegmentB* qui constituent le lien entre la composante bilingue de TransBase et les deux composantes unilingues. Ces relations lient les atomes de chaque langue avec les segments auxquels ils appartiennent dans l'alignement des documents. Par ailleurs, la relation *Contenu* attache chaque couple au document auquel il appartient.

L'implantation de la structure TransBase relève plus de la plomberie que de l'informatique, et pour cette raison, nous en dirons assez peu de choses. On peut quand même mentionner que toutes les composantes de la base de donnée résident entièrement sur fichier, et ont été conçues de façon à minimiser la quantité de mémoire vive que requièrent les programmes qui les utilisent. En particu-

lier, chaque type d'entité est storée dans un fichier distinct. La structure de ces fichiers est telle qu'on peut garantir un temps d'accès à peu près constant aux entités *AtomeA*, *AtomeB* et *Couple*, dans la mesure où on y accède via l'attribut *Position*. Par ailleurs, une table de hachage fournit un accès rapide aux entités de type *FormeA* et *FormeB*, à partir de la chaîne de caractères de l'attribut *Texte*. Pour finir, la "moitié" de la relation *Occurrence* qui décrit l'ensemble des atomes correspondant à une forme donnée, consiste en fait en une liste de valeurs de l'attribut *Position*. En implantant ces listes sous forme de *B-tree* (voir [1]), on peut accéder rapidement aux occurrences d'une forme qui se trouvent dans une zone particulière de texte. Bien que ces optimisations soient spécifiques à notre application, on peut supposer qu'elles demeurent assez générales pour profiter aussi à d'autres applications.

4 L'outil de recherche

Notre programme de concordance bilingue s'appelle *TransSearch*. Ce programme utilise un algorithme de recherche relativement traditionnel pour extraire d'une structure TransBase (section 3.2) des sous-ensemble de la concordance intégrale. Nous décrivons d'abord le langage de requête au moyen duquel ces sous-ensembles sont spécifiés, et ensuite l'algorithme de recherche.

4.1 Langage des requêtes

Nous avons défini un langage de requête abstrait pour TransSearch. La syntaxe de ce langage est décrite au Tableau 2, sous la forme de règles hors-contexte (les symboles terminaux sont en caractères gras, ou entre doubles crochets « » lorsqu'une confusion est possible; la barre verticale 'l' marque une disjonction, lorsque cette notation est préférable à une liste de règles; les items en italiques sont des commentaires). Cette grammaire est spécifiquement adaptée à des requêtes sur des textes anglais et français. Pour cette raison, le symbole *langue* produit les terminaux **e** et **f** (*English* et *français*), et *caractère* ne génère que des caractères qui sont possibles dans ces deux langues. Pour une autre paire de langue, il est clair que certains ajustements seraient requis.

Chaque requête définit un ensemble de solution (possiblement vide), constitué de *contextes bilingues* de la mémoire de traduction, qui correspondent en fait aux couples produits par le processus d'alignement. Ainsi, une base de donnée TransBase peut être vue comme un ensemble de contextes bilingues *c*, chacun consistant en une paire de chaînes d'atomes t_A et t_B. Le Tableau 3 définit récursivement la sémantique des expressions produite en utilisant la grammaire du Tableau 2. Dans la colonne "Expression" de ce tableau, les variables *r*, r_1 et r_2 désignent des *requêtes*, *l* désigne une *langue*, *e*, e_1 et e_2 sont des *expressions*, *d* est un *descripteur*, et *f* est une *forme.* Dans la colonne "Sémantique", la variable *c* désigne un contexte bilingue, les variables indicées *t* sont des chaînes d'atomes, *a* est un atome isolé, et *i*, *j* et *k* sont des entiers positifs. $S(x)$ dénote l'ensemble des solutions de l'objet *x*. Dans le cas d'une requête *r*, il s'agit d'un ensemble de contextes *c* (ou de

requête → langue **(** expression **)**
requête → négation langue **(** expression **)**
requête → requête requête.

expression → descripteur
expression → **(** expression **)**
expression → expression opérateur expression

opérateur → « » *(caractère "espace")*
opérateur → **...**
opérateur → **..** entier **..**
opérateur → **«|»**

descripteur → forme
descripteur → forme expanseur

forme → caractère
forme → caractère forme

entier → chiffre
entier → chiffre entier

langue → **e** | **f**

négation → **~**

expanseur → **+**

caractère→ **a** | **A** | **à** | **À** | ... | **z** | **Z** | **-** | «**'**» *(caractère "apostrophe")*

chiffre → **0** | **1** | ... | **9**

Les opérateurs de juxtaposition (« ») et d'ellipse (**...** et **..***k***..**) ont la même priorité, qui est moindre que celle de l'opérateur de disjonction («**|**»).

Tableau 2: Syntaxe des requêtes de TransSearch

couples), pour une expression *e*, d'un ensemble de chaînes d'atomes *t*, et pour une descripteur *d*, d'un ensemble d'atomes *a*. Dans la dernière ligne du tableau, *Morpho*(*f*) désigne l'ensemble des formes *fléchies* de *f*: Si *f* correspond à une ou plusieurs *formes de citation* d'un dictionnaire de la langue de *f*, alors *Morpho*(*f*) est l'ensemble de toutes les flexions morphologiques de ces formes de citation.

Expression	**Sémantique**
$r = l(e)$	$c \in S(r) \Leftrightarrow t_l \in S(e)$, où t_l est le segment de c en langue l
$r = \sim l(e)$	$c \in S(r) \Leftrightarrow t_l \notin S(e)$, où t_l est le segment de c en langue l
$r = r_1 r_2$	$c \in S(r) \Leftrightarrow c \in S(r_1)$ et $c \in S(r_2)$
$e = d$	$t \in S(e) \Leftrightarrow \exists k$ tel que $t = t_1 ... t_k ... t_n$ et $t_k \in S(d)$
$e = e_1 e_2$	$t \in S(e) \Leftrightarrow \exists k$ tel que $t = t_1 ... t_k t_{k+1} ... t_n$, $t_1^k \in S(e_1)$ et $t_{k+1}^n \in S(e_2)$
$e = e_1 ... e_2$	$t \in S(e) \Leftrightarrow \exists i, j$ tel que $t = t_1 ... t_i ... t_j ... t_n$, $t_1^i \in S(e_1)$ et $t_j^n \in S(e_2)$
$e = e_1 ..k.. e_2$	$t \in S(e) \Leftrightarrow \exists i, j$ tel que $t = t_1 ... t_i ... t_j ... t_n$, $t_1^i \in S(e_1)$, $t_j^n \in S(e_2)$ et $j - i < k$
$e = e_1 \mid e_2$	$t \in S(e) \Leftrightarrow t \in S(e_1)$ ou $t \in S(e_2)$
$d = f$	$a \in S(d) \Leftrightarrow a = f$
$d = f+$	$a \in S(d) \Leftrightarrow a \in Morpho(f)$

Tableau 3: Sémantique des requêtes

En pratique, ce langage nous permet une grande flexibilité dans la formulation des requêtes. D'abord il permet de rechercher des suites de mots spécifique dans l'une ou l'autre des deux langues, ou dans les deux à la fois:

f(contre toute attente) e(against all odds)

retournera toutes les occurrences de l'expression française *contre toute attente*, lorsqu'elle apparaît dans le même contexte bilingue que l'expression anglaise *against all odds*. La négation (**~**) nous permet de formuler des restrictions additionelles:

f(adresse) e(address) ~f(postale)

trouvera l'ensemble des contextes où "*adresse*" et "*address*" apparaissent ensemble, sauf dans les cas où ils co-occurent avec "*postale*" dans la partie française[4]. L'ellipse (**...**) et l'ellipse restreinte (**..*k*..**) offrent la possibilité de rechercher des expressions qui sont "éparpillées" dans le texte, notamment des expressions à l'intérieur desquelles on admet un complément où un modificateur. Par exemple

f(rivalise...avec)

trouvera à la fois des occurrences de *X rivalise avec Y*, de *X rivalise presqu'avec Y* et de *X rivalise d'adresse avec Y*. Cet opérateur est aussi utile pour abréger la formulation d'expressions où deux ou trois mots déterminent tous les autres, comme c'est le cas, par exemple, dans l'expression *ménager la chèvre et le chou*:

f(chèvre..2..chou)

Quant à la disjonction (**|**), elle permet évidemment de trouver toutes les formes d'une expression qui admet des variantes:

f(augure mal | (rien de bon))

(Comme la priorité de l'opérateur de disjonction **|** est supérieure à celle des autres opérateurs, il est nécessaire d'utiliser les parenthèses dans cette expression.) Enfin, l'expansion morphologique (l'affixe **+**) épargne à l'utilisateur la tâche fastidieuse de spécifier dans une disjonction toutes les possibilités lorsqu'une expression contient des mots qui sont sujet à des variations de nature flexionelle:

f(rendre+ justice)

trouvera non seulement les occurrences de *rendre justice*, mais aussi les formes *rendons justice*, *rendu justice*, etc.

4. Pour des raisons pratiques, le programme exige en réalité qu'au moins un des membres d'une requête soit "positif", de telle sorte que notre opérateur de négation est en fait équivalent au 'ET-NON' qu'on retrouve dans d'autres langages de requêtes.

4.2 Résolution des requêtes

La sémantique des requêtes produites au moyen du langage présenté à la section précédente suggère une méthode récursive de résolution des requêtes. La Figure 4 contient le coeur d'un pro-

```
solution_requête([], SENSIBILITE, COUPLE) :-
    !,
    nonvar(COUPLE).

solution_requête(requête(MEMBRE, RESTE_REQUETE), SENSIBILITE, COUPLE) :-
    (MEMBRE = négation(membre(LANGUE, EXPRESSION)),
     solution_requête(RESTE_REQUETE, SENSIBILITE, COUPLE),
     nonvar(COUPLE),
     (solution_expression(EXPRESSION, LANGUE, min_indéterminé, max_indéterminé, SENSIBILITE, _, COUPLE)
      -> fail
      ;  succeed)
    ;MEMBRE = membre(LANGUE, EXPRESSION),
     solution_expression(EXPRESSION, LANGUE, min_indéterminé, max_indéterminé, SENSIBILITE, _, COUPLE),
     solution_requête(RESTE_REQUETE, SENSIBILITE, COUPLE)).



solution_expression(disjonction(EXPRESSION_1, EXPRESSION_2),
                    LANGUE, MIN, MAX, SENSIBILITE, DERNIER, COUPLE) :-
    (solution_expression(EXPRESSION_1, LANGUE, MIN, MAX, SENSIBILITE, DERNIER, COUPLE)
    ;solution_expression(EXPRESSION_2, LANGUE, MIN, MAX, SENSIBILITE, DERNIER, COUPLE)).

solution_expression(conjonction(EXPRESSION_1, DISTANCE, EXPRESSION_2),
                    LANGUE, MIN_1, MAX_1, SENSIBILITE, DERNIER_2, COUPLE) :-
    solution_expression(EXPRESSION_1, LANGUE, MIN_1, MAX_1, SENSIBILITE, DERNIER_1, COUPLE),
    MIN_2 is DERNIER_1 + 1,
    (DISTANCE = indéterminée -> MAX2 = max_indéterminé ; MAX_2 is MIN_2 + DISTANCE),
    solution_expression(EXPRESSION_2, LANGUE, MIN_2, MAX_2, SENSIBILITE, DERNIER_2, COUPLE)).

solution_expression(forme(FORME, PATRON_MIN_MAJ),
                    LANGUE, MIN, MAX, SENSIBILITE, DERNIER, COUPLE) :-
    transbase(LANGUE, ATOME, FORME, COUPLE, MIN_MAJ)
    (SENSIBILITE = insensible -> succeed ; MIN_MAJ = PATRON_MIN_MAJ),
    (MIN = min_indéterminé -> succeed ; ATOME >= MIN),
    (MAX = max_indéterminé -> succeed ; ATOME =< MAX),
    DERNIER = ATOME.
```

Figure 4: Programme Prolog de résolution des requêtes TransSearch

gramme Prolog qui effectuerait une telle résolution[5]: un appel au prédicat `solution_requête` retourne l'identificateur d'un couple qui satisfait la requête fournie, et en forçant le backtracking, on obtient l'ensemble des solutions.

Ce programme accepte des requêtes qui ont été préalablement transformées en structures arborescentes, possiblement par un programme Prolog dérivé des règles de grammaire du Tableau 2. Cette transformation ne se limite toutefois pas à un simple parsage de la requête: elle inclut en outre une atomisation, une normalisation et une expansion morphologique des formes de la

5. Bien que les modules de résolution des requêtes de TransSearch ne soient pas écrits en Prolog, il nous a semblé approprié d'utiliser ce langage pour les décrire, parce que le moteur d'inférence de Prolog incorpore des mécanismes qui sont analogues à ceux qui sont employés dans l'implantation de TransSearch.

requête. L'atomisation et la normalisation sont appliquées à toutes les formes, en utilisant les mêmes règles que lors de l'analyse du texte et son insertion dans la base de donnée (voir sections 3.1 et 3.2). Notons que l'atomisation peut entraîner l'expansion de certaines formes en juxtaposition de formes. L'expansion morphologique, quant à elle, ne concerne que les formes marquées d'un **+**. À l'aide de deux dictionnaires morphologiques (un pour chaque langue), on convertit chacune de ces formes en une disjonction de formes fléchies.

Dans cette implantation simplifiée, nous avons regroupé dans un seul prédicat l'information de la base de donnée TransBase qui est nécessaire pour effectuer la résolution des requêtes:

```
transbase(?LANGUE, ?ATOME, ?FORME, ?COUPLE, ?MIN_MAJ).
```

Il existe une clause de ce prédicat pour chaque atome de la base de donnée: c'est l'argument `LANGUE` qui détermine s'il s'agit d'une entité *AtomeA* ou *AtomeB.* L'argument `ATOME` correspond à l'attribut *Position*, et `MIN_MAJ` correspond à *PatronMinMaj*. L'argument `FORME` prend la valeur de l'attribut *Texte* de l'entité *FormeA* (ou *FormeB*) associé à l'atome par la relation *Occurrence*, et `COUPLE` prend la valeur de l'attribut *Position* de l'entité *Couple* auquel l'atome est lié par la relation *SegmentA* (ou *SegmentB*).

TransSearch utilise une méthode qui, bien qu'inspiré du programme de la Figure 4, se distingue de celui-ci par deux optimisations notoires. Premièrement, on limite les recherches dans la base de donnée aux zones pertinentes: tel que mentionné à la section 3.2, l'implantation de la relation *Occurrence* permet d'utiliser les valeurs des variables `MIN` et `MAX` pour restreindre le champ de recherche aux seuls atomes dont les identificateurs se trouvent entre ces deux bornes. Deuxièmement, en utilisant le nombre d'occurrences de chaque forme (l'attribut *Fréquence*), on peut estimer la production maximale de chaque élément des conjonctions de la requête, et utiliser cette information pour déterminer l'ordre de traitement optimal de ceux-ci: en traitant d'abord les éléments les moins productifs, on arrive à minimiser le backtracking lors de la résolution.

5 Interface graphique

Le logiciel TransSearch s'adresse d'abord aux traducteurs, aux terminologues et aux lexicographes. Ce ne sont donc pas nécessairement des cracks de l'informatique. Notre désir de leur fournir un outil convivial entraînait forcément le développement d'une interface de type graphique[6]. Pour des raisons pratiques, nous avons choisi d'effectuer le prototypage de TransSearch sur des stations de travail Sun (donc, dans un environnement UNIX), avec le langage de programmation C.

6. Une interface de type 'terminal' a aussi été développée, pour les besoins internes de l'équipe de TAO. Nous ne décrivons ici que l'interface graphique, mais les deux permettent d'effectuer essentiellement les mêmes recherches.

Étant donné ce choix de plateforme, l'interface a été élaborée dans l'environnement de fenêtrage OpenWindows, au moyen de l'API XView.

L'essentiel de l'interface réside dans deux fenêtres: la fenêtre d'affichage des résultats, qui est celle qui apparaît lorsqu'on démarre l'application, et la fenêtre de recherche. Ces deux fenêtres constituent en fait les réponses aux deux principales questions que pose le problème de la convivialité dans le contexte des concordances bilingues: "Comment formuler les requêtes?" et "Comment afficher les résultats?". (Voir la Figure 5).

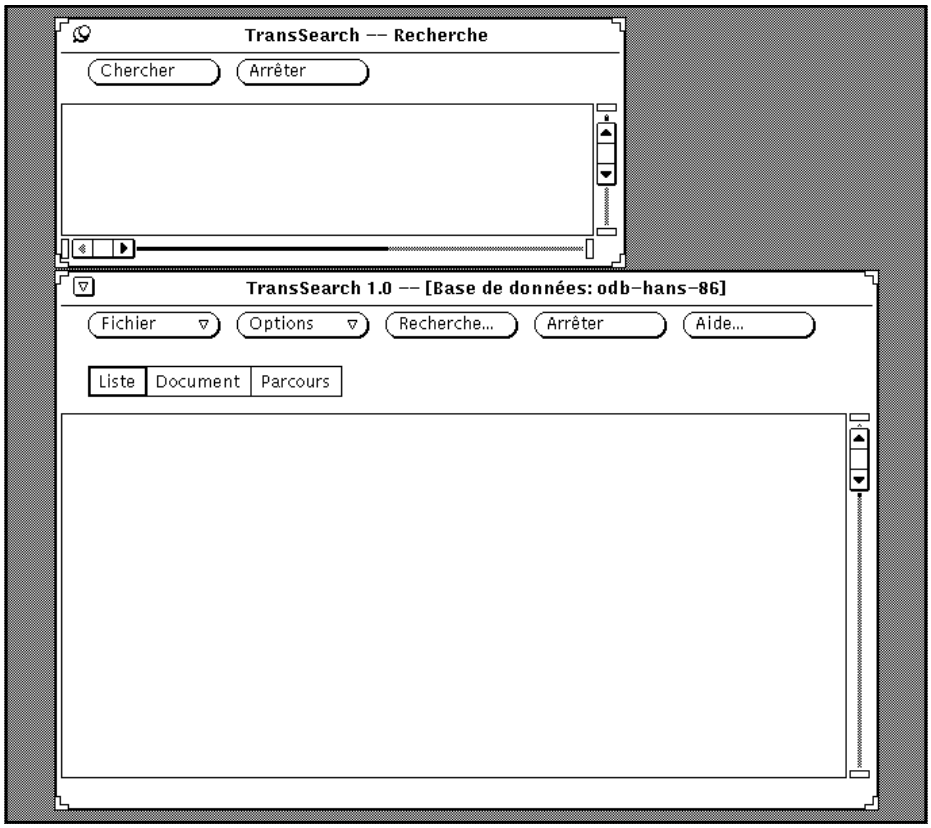

Figure 5: L'interface graphique de TransSearch

La première question nous a amené d'abord à concevoir un langage graphique intuitif pour représenter les requêtes que comprend TransSearch, ensuite à mettre au point une méthode relativement simple pour formuler des requêtes dans ce langage. La fenêtre de recherche, qui apparaît lorsque l'utilisateur appuie sur le bouton 'Recherche' de la fenêtre principale, consiste donc essentiellement en un *éditeur graphique de requêtes*. Un tel éditeur permet en outre d'éliminer presque totalement le problèmes des erreurs de syntaxe.

Dans notre langage graphique, une requête apparaît comme une liste verticale d'expressions à satisfaire, chacune étant précédée d'une étiquette marquant la langue de l'expression, et possible-

ment d'un symbole marquant une négation. Les expressions simples, c'est-à-dire consistant en une suite de formes contigües, sont représentées comme des listes horizontales de formes séparées par des espaces (en d'autres mots: comme du texte "ordinaire"), et les formes sur lesquelles une expansion morphologique sera appliquée sont soulignées. Une ellipse entre deux formes apparaît comme un trait horizontal brisé (ellipse non-restreinte) ou comme un trait surmonté d'un nombre (ellipse restreinte). Finalement, les membres d'une disjonction sont listés verticalement, et reliés entre eux par des traits. La Figure 6 illustre la représentation graphique de la requête "**f(arriver+ ... (à temps) | pile) e(arrive+ ..5.. on | in time)**".

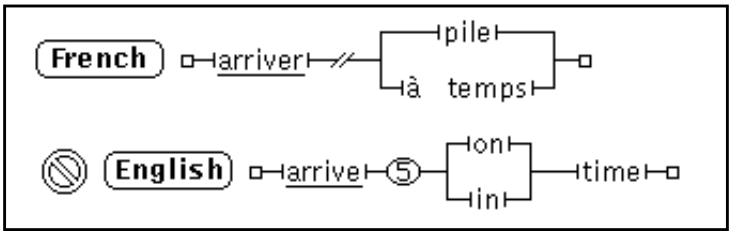

Figure 6: Représentation graphique d'une requête dans l'interface graphique de TransSearch

L'éditeur de requêtes utilise à la fois la souris et le clavier: tous les déplacements à l'intérieur d'une requête peuvent se faire avec la souris, et toutes les commandes d'édition sont accessibles via un menu d'édition. Seule l'édition des chaînes de caractères qui constituent les formes d'une requête requiert l'utilisation du clavier. Il est toutefois possible d'éditer entièrement et de soumettre une requête sans jamais avoir à utiliser la souris.

Lorsque l'utilisateur a fini de formuler sa requête et qu'il en lance la résolution, le programme commence par vider sa zone d'affichage, laquelle se trouve au centre de la fenêtre principale de l'application. L'utilisateur peut alors visualiser les résultats à mesure que ceux-ci sont produits, de même qu'il peut interrompre la recherche quand il le désire, à l'aide du bouton 'Arrêter'.

Comment les résultats sont-ils représentés? Il est évident qu'on ne veut pas employer la méthode "concordance classique", c'est-à-dire une ligne par résultat, les occurrences proprement alignées au centre de l'affichage. D'abord, ce format est difficile à concilier avec des requêtes complexes telles que celles que nous permet de formuler notre langage de requêtes. Ensuite, il s'adapte mal à l'affichage d'un contexte bilingue. Finalement, il est clair qu'il s'agit d'un archaïsme, datant d'une époque où les concordances étaient imprimées intégralement sur papier. (Quiconque a déjà vu un linguiste trimbaler une concordance sait que s'ils avaient voulu plus de contexte, il aurait aussi fallu leur fournir une brouette.) Nous avons conçu et implanté deux modes distincts de visualisation des résultats, qui nous semblaient mieux adaptés aux besoins des utilisateurs.

Le *mode liste* est ce qui ressemble le plus au format classique: tous les couples trouvés sont affichés, un en dessous de l'autre. Des traits horizontaux séparent les couples, et le texte de chaque

couple est affiché en deux colonnes, c'est-à-dire une pour chaque langue. Une flèche apparaît entre les deux colonnes, pour indiquer la *direction* de la traduction (quel est le texte-source, quel est le texte-cible). Les occurrences de mots qui ont permis de satisfaire la requête sont affichés en caractères gras[7] (Figure 7). L'utilisateur peut alors se déplacer à son gré dans la liste, soit en utilisant le scrollbar, soit avec des commandes au clavier qui permettent de centrer l'attention sur un résultat particulier, que nous appelons le *résultat courant*. De l'information additionelle sur ce résultat apparait au dessus de la zone d'affichage: nom du document, identificateur du couple, etc.

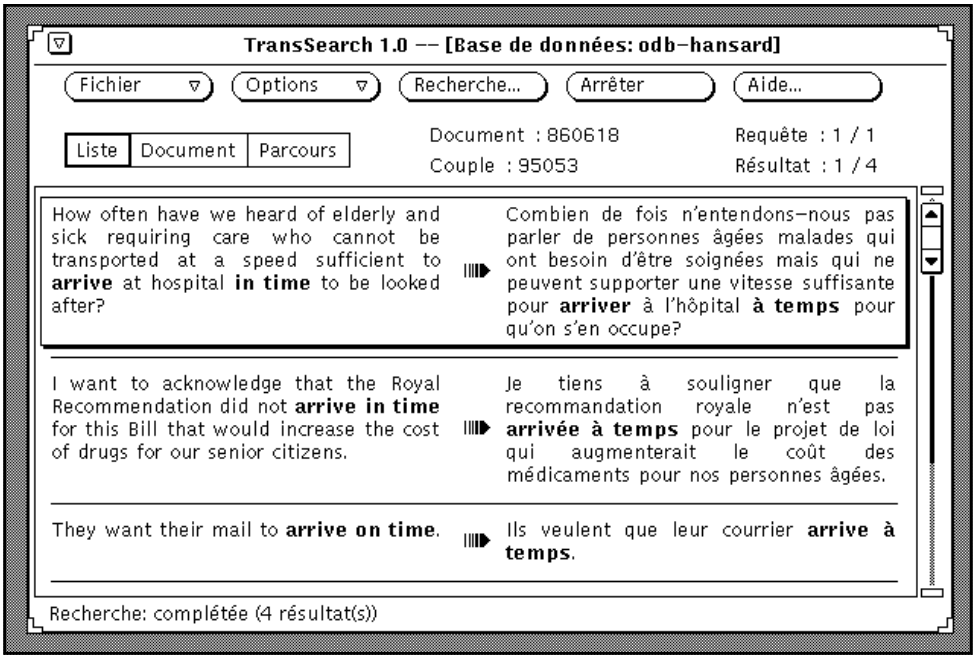

Figure 7: Affichage d'une concordance bilingue dans la fenêtre principale de TransSearch

L'autre mode d'affichage est le *mode document*. Lorsque l'utilisateur passe dans ce mode, la zone d'affichage se consacre entièrement au résultat courant, et affiche la totalité du document contenant ce couple. Encore une fois, les couples qui constituent le document sont affichés verticalement et en deux colonnes. Pour le distinguer des autres, le résultat courant apparaît encadré, et les occurrences ayant permis de satisfaire la requête y sont en caractères gras. L'utilisateur se sert alors obligatoirement des commandes au clavier pour passer d'un résultat à l'autre, puisque le scrollbar ne permet que des déplacement à l'intérieur du document.

7. À ce sujet, il est intéressant de remarquer que la façon dont TransSearch interprète une requête n'est pas nécessairement conforme à la perception qu'en a l'utilisateur "naïf": alors que TransSearch recherche des couples qui satisfont une requête, l'utilisateur recherche des occurrences d'une certaine expression. Pour cette raison, lorsqu'un couple est trouvé, TransSearch calcule et met en évidence *toutes* les combinaisons de formes qui satisfont la requête à l'intérieur de ce couple, ceci afin de répondre à l'attente de l'usager.

Bien que l'interface ne se concentre que sur une requête à la fois, toutes les requêtes qui ont été soumises au cours d'une session, de même que tous les résultats obtenus pour ces requêtes, sont conservés, et l'utilisateur peut y avoir accès rapidement.

L'interface offre aussi un *mode parcours*, qui permet de visualiser le contenu d'un document de la base de donnée, simplement en en spécifiant le nom, sans qu'il soit nécessaire de soumettre une requête. Finalement, l'interface permet de n'afficher qu'une seule des deux langues, de telle sorte que l'outil peut être utilisé pour effectuer des concordances "ordinaires".

Conclusions

Le prototype UNIX de TransSearch en est maintenant à sa deuxième version, et est utilisé de façon quotidienne au CITI, par les linguistes et informaticiens de l'équipe TAO. Le logiciel a fait l'objet de plusieurs démonstrations, notamment à Ottawa (ATIO '93), à Paris (Premier Colloque Génie-Linguistique AUPELF-UREF) et au Japon (MT Summit '93), et à chaque fois, les réactions des experts et des utilisateurs potentiels ont été extrêmement positives. Un essai opérationnel dans un service de traduction est prévu pour l'automne '93. Toute cette activité ne nous empêche cependant pas d'envisager plusieurs améliorations techniques au système, à court et à moyen terme.

D'abord, il est clair que les utilisateurs potentiels de TransSearch ne disposent pas de stations Sun, et qu'ils ne courreront pas s'en acheter la semaine prochaine. Il est donc impératif d'effectuer le portage du logiciel vers une plateforme plus répandue (PC ou Mac).

L'analyse linguistique des textes qui sont insérés dans la mémoire de traduction est présentement très rudimentaire, et tout porte à croire que nous gagnerions beaucoup à l'approfondir. En particulier, il semble qu'un étiquetage morpho-syntaxique des mots de la mémoire de traduction permettrait un meilleur filtrage du "bruit" dans les ensembles de solution, par exemple en éliminant les occurrences de la forme *est* du verbe *être,* lors d'une recherche sur les points cardinaux. Un tel étiquetage ouvre aussi la porte à une atomisation plus fine, de même qu'à certaines transformations des textes de la base de donnée, qui permettraient une résolution plus intelligente des requêtes, en détectant des solutions qui se cachent derrière certaines particularités du langage. Par exemple, le découpage de la forme *des* en *de* + *les* permettrait d'identifier correctement la forme *en dépit des apparences* comme une occurrence de *en dépit de*. Par ailleurs, on ne peut identifier la phrase *Mary's blue* comme une occurrence de l'expression *to be blue* que si on a reconnu dans la forme *'s* une flexion contractée du verbe *to be*. Ces découpages ne sont toutefois possibles que si l'on est en mesure de résoudre certaines ambiguités lexicales. Nous envisageons, pour ce faire, utiliser des étiqueteurs lexicaux basés sur des modèles stochastiques du langage, tel que celui de Foster (voir [5]).

La suggestion qui revient le plus souvent lors des séances de consultation avec les utilisateurs, est la mise en évidence, dans le contexte en langue 'B', des mots qui constituent la traduction de la requête, lorsque celle-ci ne contient que des expressions en langue 'A'. Par exemple, l'utilisateur qui recherche des occurrences de l'expression anglaise *to be out to lunch*, aimerait qu'on affiche en caractère gras, dans la portion française des contextes trouvés, la traduction exacte de *out to lunch* (*se met le doigt dans l'oeil* ici, *est dans la choucroutte* là, etc.). Ce genre de capacité exige des correspondances plus fines que celles sur lesquelles TransSearch se base présentement. Des algorithmes de *correspondance mot-à-mot* existent (voir par exemple [4] et [8]), mais il reste à évaluer l'applicabilité de ces méthodes, et à les intégrer au système.

Finalement, si notre structure de mémoire de traduction résoud certains problèmes pour les traductions bilingues, elle n'est présentement pas adaptée aux traductions *multilingues*. Ce problème est des plus intéressant, car il semble d'une part que l'existence de plusieurs versions d'un même document permettra d'arriver à de meilleurs correspondances, et d'autre part que des concordances multilingues mettront en évidence des équivalences de traduction entre des paires de langues pour lesquelles les références terminologiques font défaut. Cette question fait présentement l'objet de travaux exploratoires au CITI.

Références

[1] Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman. *Data Structures and Algorithms*, Addison-Wesley, 1982.

[2] Berry, Mark. "The Trados Translator's Workbench II". *Proceedings of the 33rd Annual Conference of the American Translators Association*, San Diego, Californie, novembre 1992.

[3] Church, Kenneth Ward et William A. Gale. "Concordances for Parallel Texts". *Proceedings of the 7th Annual Conference of the UW Centre for the NOED and Text Research*, Oxford, 1991.

[4] Dagan, Ido, Kenneth Ward Church et William A. Gale. "Robust Bilingual Word Alignment for Machine Aided Translation". *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, Ohio State University, Columbus, Ohio, juillet 1993.

[5] Foster, George F. "Statistical Lexical Disambiguation". Master's thesis, McGill University, School of Computer Science, Montréal, Canada, 1991.

[6] Gale, William et Kenneth Ward Church. "A Program for Aligning Sentences in Bilingual Corpora". *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91)*, Berkeley, Californie, juin 1991.

[7] Isabelle, Pierre, et al. “Translation Analysis and Translation Automation”. *Proceedings of the Fifth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-93)*, Kyoto, Japon, juillet 1993.

[8] Kay, Martin et Martin Röscheisen. “Text-Translation Alignment”. *Computational Linguistics*, Vol. 19, No. 1, 1993.

[9] Macklovitch, Elliott. “Le PTT, ou les aides à la traduction”. *La traductique : études et recherches de traduction par ordinateur*, Pierrette Bouillon et André Clas (éditeurs), Les Presses de l’Université de Montréal, 1993.

[10] Macklovitch, Elliott. “Corpus-based Tools for Translators”. *Proceedings of the 33rd Annual Conference of the American Translators Association*, San Diego, Californie, novembre 1992.

[11] Simard, Michel, George Foster et Pierre Isabelle. “Using Cognates to Align Sentences in Bilingual Corpora”. *Quatrième Colloque international sur les aspects théoriques et méthodologiques de la traduction automatique (TMI-92)*, Montréal, Canada, juin 1992.

Remerciements

Nous tenons à remercier tout particulièrement Pierre Plamondon et Nicolas Viau pour leur inestimable contribution à la mise sur pieds de TransSearch, de même que Pierre Isabelle pour sa patience et son support continuel. Nous sommes également infiniment redevables à Elliott Macklovitch, Marc Dymetman, Marie-Louise Hannan, Jean-Marc Jutras, Xiaobo Ren, Benoît Robichaud et Caroline Viel pour leurs précieux conseils lors de l’élaboration des multiples prototypes.