

LA TRADUCTION AUTOMATIQUE FRANCAIS-ITALIEN UNE MICRO APPLICATION

Nous présentons ici le résumé du mémoire de maîtrise d'italien présenté en 1989 à Paris III sous le titre : *La Traduction Automatique français-italien : un essai de micro-programmation : traduction de recettes de cuisine du français vers l'italien.*

La Traduction Automatique consiste à traduire un texte ou un énoncé d'un langage naturel - ici le français - vers un autre - l'italien - au moyen d'une machine - l'ordinateur -, et sans intervention humaine. Plus simplement, c'est la faculté d'une machine à écrire dans une langue Y la traduction d'un texte que l'utilisateur a rentré dans une langue X.

Il est alors facile d'imaginer que la mise au point d'une telle machine pose un certain nombre de problèmes. Ce sont ces problèmes, ou tout au moins une partie d'entre eux, que nous avons tenté de déterminer dans cette recherche, afin de les résoudre.

Cette recherche touche à la fois les domaines de l'informatique et de la linguistique, la linguistique devant y définir le fonctionnement des langues et de la traduction, l'informatique devant, quant à elle, coder les informations fournies par la linguistique pour simuler la traduction.

Certaines difficultés rencontrées dans la mise au point d'un traducteur français-italien pourraient être rencontrées pour n'importe quel traducteur automatique, de même que la recherche en T.A. s'insère dans un champ plus large : l'Intelligence Artificielle où l'on tente de définir le comportement de l'homme et de construire une machine qui simule ce comportement. En développant notre traducteur français-italien, nous avons donc rencontré les questions communes à toute machine à traduire, et plus généralement, à toute recherche en Intelligence Artificielle. D'autre part nous sommes aussi confrontés à des aspects plus spécifiques à la traduction français-italien, dans

les limites que nous avons fixées à notre sujet, et que nous préciserons plus tard.

Contrairement à la recherche pure, dont les résultats ne seront utilisés que bien des années plus tard, s'ils le sont jamais, nous avons affaire ici à une recherche dont l'objet est de construire des applications. L'informatique est justement aujourd'hui un moyen de mettre à l'épreuve des théories linguistiques et de juger si elles sont applicables. Elle permet de rentrer les règles d'une grammaire et de juger si ces règles permettent de construire des phrases correctes ou non. C'est dans cet esprit que nous avons tenté de construire nous-mêmes une application, un traducteur français-italien réduit qui permette de découvrir les étapes nécessaires et les difficultés d'une telle programmation.

Nous avons alors dû définir ce que nous demanderions à notre automate de savoir faire. Les phrases qu'il devrait pouvoir traduire ont ainsi été définies à partir d'une grammaire et d'un vocabulaire, en même temps qu'était définie la façon dont cet automate devrait traduire les phrases. Ayant ainsi défini une grammaire et un vocabulaire limités, et non un nombre de phrases limité, nous donnons à la machine les instruments de l'homme, nous apprenons à la machine à traduire des phrases que nous n'avons même pas imaginées, de même qu'un étudiant à l'U.F.R. d'italien de Paris III pourra comprendre des phrases qu'il n'a jamais entendues et même les traduire, s'il en connaît la grammaire et le vocabulaire. En effet, nous connaissons un nombre limité de constructions grammaticales, et non de phrases, qui nous permettent théoriquement de construire une infinité d'énoncés. Ainsi la phrase « Cuire l'eau dans les carottes », qui semble absurde, est sans doute nouvelle pour nous, mais elle est compréhensible et traduisible. Notre machine doit donc aussi être capable de la traduire.

Nous avons aussi décidé dès le début de limiter notre grammaire aux infinitives et notre vocabulaire... à celui de la cuisine. Ainsi notre programme traduira des recettes de cuisine et nous définirons précisément plus loin la grammaire de notre traducteur. Il est à noter que si, en définissant ces limites, nous parvenons aux traductions escomptées, rien n'empêche de penser que, avec des limitations moindres, et toujours à condition de rentrer une grammaire correcte, la machine ne pourrait pas traduire plus d'énoncés. En créant une application simple qui fonctionne, nous ouvrons donc la porte à des applications plus complexes.

Avant de voir plus précisément comment notre application est construite, nous commencerons par étudier rapidement ce qui existe actuellement dans le domaine de la traduction automatique, quelles en sont les grandes lignes, les possibilités, et aussi quelles sont les recherches connues en cours. Ensuite nous décrirons notre propre recherche d'élaboration d'une

machine à traduire. Enfin nous étudierons les résultats de notre travail et les améliorations faites par rapport au projet de départ.

1) La traduction aujourd'hui, ses enjeux

La machine à traduire semble pour beaucoup inconcevable parce que, pour nous, traduction implique intelligence, compréhension. Un traducteur humain doit en effet commencer par lire un texte, pour le comprendre, avant de le traduire. Cette compréhension relève de ce que nous appelons l'intelligence et que nous considérons souvent comme propre à l'homme. De même, une fois le texte compris, il doit être retranscrit dans la langue cible. Il y a alors générations d'énoncés dans une langue naturelle, ce qui est encore pour nous une preuve d'intelligence. Parce que nous ne savons pas expliquer comment nous élaborons nos phrases, nous parlons d'intelligence.

Ici réside l'essentiel du problème de l'intelligence artificielle, et par là même de la traduction automatique : pour faire un automate qui agisse comme l'homme, il faut lui donner les règles de fonctionnement des actes humains, afin qu'il puisse suivre ces règles. L'ordinateur ne sait faire que ce qu'on lui apprend. Nous ne pouvons donc pas simuler l'intelligence de l'homme avec des machines tant que nous n'en connaissons pas les règles. Or, si notre intelligence semble « logique », nous ne savons pas en décrire toutes les règles de fonctionnement. L'homme est incapable d'expliquer comment il apprend à parler. Ainsi, s'il sait expliquer comment démontrer un théorème complexe, il ne peut expliquer comment il comprend les phrases les plus simples, comment il les analyse « inconsciemment ». C'est donc au niveau des règles les plus simples, de ce qui apparaît comme « évident », que l'intelligence artificielle pose le plus de problèmes. Les règles de grammaire complexes, qui sont décrites dans les livres de grammaire, seront par exemple moins difficiles à coder que des pronominalisations, des associations qui nous semblent pourtant simples. Dans la phrase « C'est pas juste », le « C » pourra se référer à toute une suite de raisonnements, d'associations ou d'actions, de références difficilement définissables grammaticalement, mais qui paraîtront naturels aux locuteurs français.

Nous devons donc nous aussi, pour notre traducteur, chercher à définir des règles et les modalités d'application de ces règles, pour faire faire à la machine ce qui nous semble évident.

La Traduction Automatique intéresse les chercheurs parce qu'elle représente en ce sens un réel enjeu pour l'homme, mais aussi bien sûr pour des raisons commerciales. Quantité de textes sont traduits chaque jour, et la C.E.E.

a ainsi mis en place un système de traduction automatique (Systran) pour ses propres traductions.

Historiquement, on est progressivement passé de la traduction au simple mot-à-mot à une traduction plus évoluée où la phrase est analysée grammaticalement avant d'être traduite. L'élément à traduire n'est plus le mot, mais la phrase.

Actuellement, deux grandes théories s'opposent dans la T.A. : la plus ambitieuse est celle du « langage pivot » où l'on cherche à définir un langage unique de codification des langues. Toute langue pourrait être traduite dans ce langage pivot. Ainsi, pour chaque langue, il suffirait de créer un traducteur de cette langue vers le langage pivot et un autre dans le sens inverse. On pourrait alors, avec simplement deux traducteurs par langue naturelle, traduire d'une langue vers l'autre en passant par ce langage intermédiaire. Ce langage pivot présente donc une grande économie de traducteurs (pour N langues naturelles, on se contente de $2 * N$ traducteurs, alors que l'on pourra avoir des traductions dans $N * (N-1)$ couples de langues), mais la création de ce langage pivot et des traducteurs vers ou à partir de ce langage demande des recherches considérables. Ce langage devrait en effet reconnaître des notions, des précisions propres à un petit nombre de langues, si ces précisions sont nécessaires à la traduction d'une seule langue vers une autre. Un grand nombre d'informations devrait être rentré, alors qu'une petite partie d'entre elles sera nécessaire à la traduction. Ainsi, si « frère » a plusieurs significations dans une langue, il faudra que le langage pivot fasse la différence entre toutes les définitions possibles de « frère », alors que ces précisions seront inutiles dans la plupart des couples de langues. Ce système peut donc être intéressant pour traduire des langues différentes l'une de l'autre, mais pour une traduction de langues relativement proches, comme c'est le cas du couple de langues français-italien, il ne peut apporter un progrès par rapport à une traduction plus traditionnelle : la traduction par couples de langues, où l'on ne passe pas par un intermédiaire.

C'est donc cette deuxième solution que nous utiliserons pour notre recherche. C'est en outre la solution qui a été choisie pour les machines à traduire commercialisées. Un même « moteur », une même logique sont utilisés pour faire les raisonnements de la traduction, quelles que soient les langues traitées. Mais les règles de grammaire ou de traduction sont données pour chaque couple de langues, en fonction des besoins. Ainsi, si « jaune » peut toujours se traduire par « yellow » en anglais, il n'est pas nécessaire d'analyser plus précisément « jaune » dans une traduction franco-anglaise. Mais pour le couple français-italien, il faut analyser « jaune » pour savoir s'il s'agit d'un jaune d'œuf, traduit « tuorlo » et non « giallo ». Pour une traduction dans un couple de langues donné, on ne garde que les informations pertinentes.

Cette méthode apporte donc une économie de traitement, d'informations, par rapport à l'autre méthode. Elle doit être utilisée pour des traductions entre des langues proches, et pour des couples de langues dans lesquels les traductions sont fréquentes.

La programmation des traducteurs est généralement faite dans des langages de programmation créés spécialement pour l'intelligence artificielle. C'est le cas de Prolog que nous avons choisi pour notre travail. Prolog est un langage de programmation déclaratif adapté à l'intelligence artificielle.

Contrairement aux langages de programmation traditionnels qui sont impératifs et où le programmeur doit prévoir, pour tous les cas, les chemins de passage dans le programme, les adresses par lesquelles on passe, où le programme est fait principalement d'instructions dans la programmation déclarative le programmeur écrit uniquement des faits et des règles. Ensuite le programmeur peut poser une question et Prolog tente de répondre à cette question en utilisant les faits et les règles qui lui ont été donnés et en utilisant son propre moteur d'inférence qui sait faire des raisonnements logiques. Ainsi, si le programmeur a rentré les faits :

F1 : « le » est un article,

F2 : « lait » est un nom,

et la règle :

R1. : un groupe nominal est formé d'un article suivi d'un nom,

si nous posons à Prolog la question : « le lait » est-il un groupe nominal ? il cherchera la définition d'un groupe nominal et trouvera la règle R1. Il cherchera alors à voir si « le lait » est formé d'un article suivi d'un nom. Il trouvera avec F1 que « le » est un article, et avec F2 que « lait » est un nom, ce qui correspond à ce qu'il cherchait pour résoudre R1. Il répond donc « oui » à la question posée.

En fait F1 s'écrit plutôt : article (« le »)

et F2 : nom (« lait »)

R1 s'écrit : gn (X,Y) si article (X) et nom (Y)

(Ceci n'est qu'un exemple de notation.)

Une règle s'écrit toujours :

(résultat) si (conditions)

et le moteur de Prolog fonctionne par chaînage arrière, c'est-à-dire que, si nous lui posons une question, il cherche un fait qui soit directement une réponse. S'il n'en trouve pas, il cherche une règle R dont le résultat (à gauche) réponde à la question. Il tente alors de vérifier les conditions notées dans la partie droite de cette règle. Pour cela il pourra avoir recours à une nouvelle règle qui dit que, pour remplir cette condition, il faut qu'une nouvelle condition soit remplie. Et à chaque fois, il tentera de vérifier ces conditions jusqu'à ce qu'il trouve une solution ou qu'il ait tout tenté sans en trouver.

Prolog nous intéresse donc parce qu'il permet de ne rentrer que des règles et des faits tandis que le moteur s'occupe de tous les « raisonnements ». Nous profitons ainsi des recherches qui ont déjà été faites par d'autres en logique appliquée. De plus, grâce à ce moteur, Prolog présente d'autres avantages pour nous. Il nous permet d'améliorer facilement notre programme, de le corriger ou de le modifier, simplement en ajoutant, en modifiant, ou en retirant des règles ou des faits. Nous pouvons commencer par élaborer une sorte de programme minimum, et l'améliorer à chaque fois que nous atteignons le but que nous nous sommes fixé. Nous avons ainsi commencé par créer un analyseur grammatical qui donnait simplement la structure grammaticale des phrases françaises. Puis nous avons transformé cet analyseur en traducteur.

II) L'élaboration de l'application

La situation de notre travail par rapport à ce qui se fait actuellement ayant été brièvement décrite, voyons maintenant comment nous avons procédé pour notre propre application.

1°) Découpage de la traduction en tâches successives

Il nous faut d'abord définir les étapes nécessaires à la traduction. Nous avons vu qu'il doit y avoir « compréhension » d'un texte avant que l'on puisse en générer la traduction. Mais la compréhension peut, elle aussi, se faire en plusieurs étapes et nous devons définir chacune d'entre elles avec précision si nous voulons enseigner à la machine comment « comprendre » assez les phrases que nous lui donnerons pour qu'elle puisse ensuite les traduire.

Avant de comprendre une phrase entière, nous tirons des informations des mots, à partir desquels nous pouvons commencer l'analyse de la phrase, certains mots nous apportant plus d'informations que d'autres. Notre programme commencera donc par découper la phrase en mots et par chercher dans un premier dictionnaire toutes les informations qu'il pourra tirer de

chaque mot. Ce dictionnaire pourra parfois indiquer d'emblée la traduction d'un mot donné. Ainsi « carottes » se traduira toujours « carote ». Mais il sera impossible de donner la traduction d'un article, puisque celle-ci dépend de la traduction du nom qui suit. Le premier dictionnaire donnera aussi la catégorie syntaxique des mots, élément indispensable pour l'étape suivante : l'analyse grammaticale.

Cette deuxième étape se fait elle-même en plusieurs temps. La phrase doit d'abord être découpée en groupes de mots de façon logique (on ne doit pas découper « couper les carottes » en « couper les » / « carottes »), après quoi on analyse chaque groupe de mot pour déterminer s'il s'agit bien d'un syntagme grammatical ou d'un mot. Si la réponse est positive pour chaque groupe de mots, on cherche à savoir si, et comment, cette phrase ainsi découpée en une suite de syntagmes ou de mots, suit bien les règles de grammaire de construction d'une phrase. S'il y a un échec, on essaye un autre découpage de la phrase.

C'est de plus au cours de cette analyse que sont déterminés les mots italiens que l'on ne pouvait trouver dans le premier dictionnaire. Ils sont calculés en application d'une « grammaire italienne » interne au programme.

La construction de la structure de la phrase italienne se faisant en même temps que l'analyse de la phrase française, il ne reste alors que l'étape finale d'écriture de la traduction italienne à partir de la structure qui a été construite au cours de l'analyse.

2°) Limitation et analyse du vocabulaire, définition d'une grammaire indépendante du contexte

Avant toute programmation, il nous faut définir, pour la première étape, le contenu du premier dictionnaire, et pour la seconde étape, les règles de reconnaissance d'un « syntagme grammatical », ou d'une phrase analysée comme une suite de syntagmes grammaticaux ou de mots.

D'emblée nous avons limité la grammaire et le vocabulaire de notre programme, prenant pour domaine d'application les recettes de cuisine et limitant la grammaire aux infinitives. Notre grammaire est définie par les règles d'une grammaire indépendante du contexte.

Une grammaire indépendante du contexte distingue deux types d'éléments :

- les éléments terminaux, qui sont les mots,
- les éléments non terminaux : groupes verbaux, groupes nominaux, catégories syntaxiques, tout sauf les mots.

Les règles de cette grammaire sont du type :

$$X \text{ --- } > W$$

où X est un élément unique du vocabulaire non terminal et est non nul, et où W peut être composé d'éléments du vocabulaire terminal ou non terminal. La flèche signifie: « se réécrit », « se décompose en ». Le tout se lit : X (par exemple GN, pour groupe nominal) est composé de W. Par exemple, pour X = GN, W= Article + Nom.

Pour notre grammaire, les éléments du vocabulaire non terminal sont au nombre de seize :

Imp (impératif),
 GV (groupe verbal),
 GN (groupe nominal),
 Gadv (groupe adverbial),
 VO (verbe intransitif),
 V1 (verbe transitif à objet simple),
 V2 (verbe transitif à objet double),
 Nom_à_c (nom qui se compose avec un complément de nom),
 Nom (nom "classique")
 Prep (préposition),
 Pr_art (préposition + article),
 Artdef (article défini),
 Artind (article indéfini),
 Artnum (article numéral),
 Coord (coordination),
 Adv (adverbe).

Les éléments terminaux sont tous les mots du vocabulaire. Nous n'en donnons donc que quelques exemples :

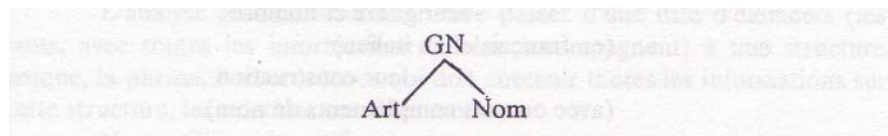
vi	--->	« cuire »
V2	--->	« ajouter »
VO	--->	« saler »
Nom	--->	« carotte »
Nom	--->	« poissons »
Nom_à_c	--->	« verre »
Artdef	--->	« les »
Prep	--->	« à »
Pr_art	--->	« aux »

Nous pouvons maintenant écrire les règles de notre grammaire, en dehors de celles qui définissent le vocabulaire terminal. Elles sont au nombre de quinze :

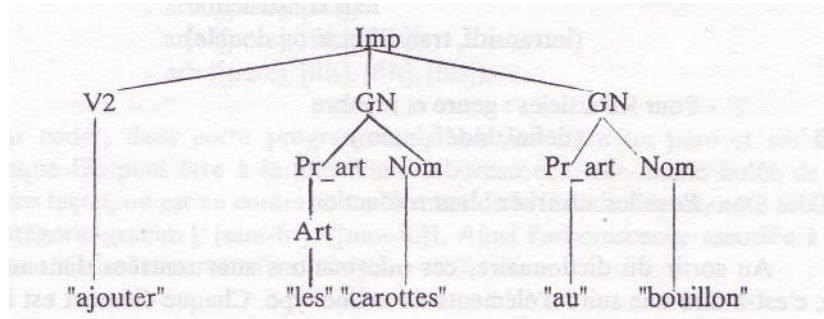
Imp	--->	GV,
Imp	--->	Gadv + GV,
GV	--->	VO,
GV	--->	V1 + GV,
GV	--->	V2 + GN + GN,
GN	--->	Pr_art + Nom,
GN	--->	Pr_art + Nom_à_c + GN,
GN	--->	GN + Coord + GN,
Pr_art	--->	Artdef,
Pr_art	--->	Artind,
Pr_art	--->	Artnum,
Pr_art	--->	Prep + Artdef,
Pr_art	--->	Prep + Artind,
Pr_art	--->	Prep + Artnum,
Pr_art	-->	Pr-art + Artnum.

Attention cette grammaire nous servira seulement à analyser les énoncés en français. Elle ne pourrait servir à engendrer des phrases car elle est trop puissante. Elle risquerait en effet de créer une multitude de phrases agrammaticales.

Traditionnellement on utilise pour ce type de grammaire une représentation arborescente où l'élément à gauche de la flèche (le père) est au dessus des éléments à droite de la flèche (les fils). « GN --> Art + Nom » est représenté :



L'analyse de la phrase « Ajouter les carottes au bouillon » donne alors :



Il nous reste à analyser plus précisément le fonctionnement de chacune des étapes du programme :

3°) Les différentes étapes de la traduction

a) Lecture de la phrase à traduire

Il n'y a pas ici d'analyse du contexte puisque nous avons toujours des recettes de cuisine. La lecture ne pose pas non plus de problème. Le texte « Entrez la phrase à traduire » est affichée à l'écran et l'utilisateur tape la phrase. Une instruction de Prolog permet ensuite de décomposer facilement la phrase en une succession de mots.

b) Le premier dictionnaire

Il nous faut alors rechercher chaque mot dans le premier dictionnaire. Ce dictionnaire a pour entrée les mots français qui composent le vocabulaire accepté. Il doit apporter le plus d'informations possible.

- Il donne donc toujours la catégorie syntaxique du mot. Si un mot peut appartenir à plusieurs catégories grammaticales, il est entré autant de fois dans le dictionnaire, et le deuxième emploi sera choisi si le premier ne convient pas pour l'analyse, puis le troisième, et ainsi de suite.

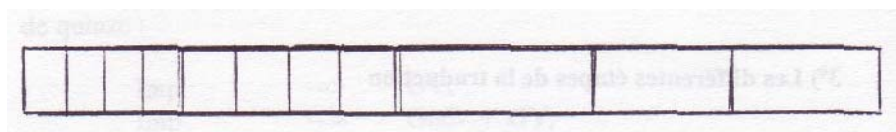
- Il donne de plus,

- Pour les noms : - leur traduction
 - leur genre et nombre
(en français et en italien)
 - leur construction
(avec ou sans compléments de nom)

- Pour les verbes : leur traduction
leur construction
(intransitif, trans. simple ou double)
- Pour les articles : genre et nombre
(défini, indéf., num.)
- Pour les adverbes : leur traduction

Au sortir du dictionnaire, ces informations sont rentrées dans une liste, c'est-à-dire une suite d'éléments du même type. Chaque élément est ici de la forme :

Infos-mot-fr. Infos-mot-it. arbOOO(catégorie, mot, trad.)



Certaines cases sont bien sûr vides au départ, et sont remplies au cours de l'analyse grammaticale. Pour un nom, toutes les cases peuvent être remplies en sortant du dictionnaire, alors que la case des informations italiennes est vide pour un article.

c) Apprentissage du vocabulaire nouveau

Un programme « intelligent » doit être capable d'apprendre ce qu'il ne sait pas faire. Nous nous contentons ici de permettre à l'utilisateur de rentrer dans le dictionnaire le vocabulaire qu'il désire y voir.

Si un mot de la phrase à traduire n'apparaît pas dans le dictionnaire, le programme affiche un message qui propose de l'y entrer. On peut ainsi ajouter des noms ou des verbes dans le dictionnaire. Le programme demande alors bien sûr la traduction de ces mots, leur construction et, pour les noms, leurs genres et nombres dans les deux langues.

d) La syntaxe et les déplacements dans l'arborescence

L'analyse grammaticale doit faire passer d'une liste d'éléments (les mots, avec toutes les informations qui les accompagnent) à une structure

unique, la phrase, l'arborescence qui doit contenir toutes les informations sur cette structure, les relations entre les mots.

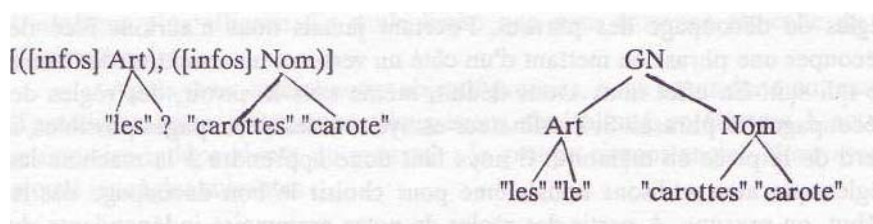
Nous utilisons la notation

```
arb ([père], [fils]);
arb ([père], [fils], [fils])
arb ([père], [fils], [fils], [fils]),
```

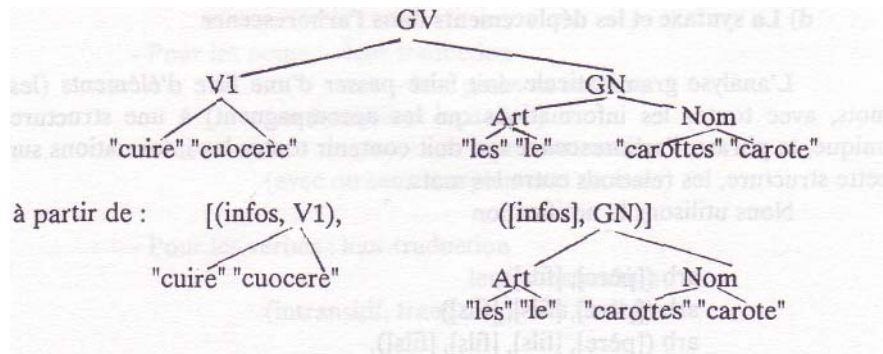
pour coder, dans notre programme, la relation entre un père et ses fils. Chaque fils peut être à la tête d'une arborescence, elle-même notée de la même façon, ou est au contraire un élément de la liste de départ, noté arb000 ([catégorie-gramm.], [mot-fr.], ([mot-it.])). Ainsi l'arborescence associée à la phrase « Cuire les carottes » est codée :

```
arb (Imp, arb000 (V1, « cuire », « cuocere »),
    arb (GN, (arb000 (Art, « les », « le »),
        arb000 (Nom, « carottes », « carote»))).
```

Pour déduire une structure de ce type à partir de la liste de départ, le programme découpe cette liste en deux ou trois morceaux et tente d'associer chacun de ces morceaux à une arborescence, c'est-à-dire d'en faire une analyse grammaticale. Si ce morceau de la liste n'est en fait qu'un seul élément, comme arb000 (V1, « cuire », « cuocere »), il n'y a rien à faire, on a déjà une arborescence associée. Sinon on découpe à nouveau ce morceau de liste et on refait la même recherche d'une arborescence associée à la liste. On ne commence à créer la structure générale que lorsque l'on est descendu jusqu'au niveau des mots. On utilise alors les premières règles qui associent une arborescence. Par exemple la règle « GN --> Art + Nom » associée à la liste



Quand les niveaux inférieurs de l'arborescence ont ainsi été construits, on cherche à construire, une arborescence à partir de ces petites arborescences associées aux morceaux de la liste de départ. Ainsi GV --> V1 + GN permet de construire



mais, chaque fois que l'on échoue à la construction d'une arborescence, il faut redécouper la liste d'une autre façon.

N'oublions pas que cette analyse n'a de sens que si elle permet de construire en même temps la traduction. Aussi, en même temps que se construit l'arborescence, rajoute-t-on à chaque étape les informations que l'on peut en déduire et qui peuvent être utiles à la traduction, dont en particulier la traduction des mots.

A partir de la connaissance des mots, on tire des informations sur le syntagme composé par ces mots, de même que l'on apporte des informations sur un syntagme à partir des informations sur les syntagmes qui le forment.

Ainsi, dans la construction du syntagme nominal, celui-ci prend le genre et le nombre du nom et est marqué par le type de l'article (défini, indéfini ou numéral), ce qui permet de calculer ensuite la traduction de l'article en italien. La traduction de l'article est donc donnée dès que la construction du syntagme nominal est faite.

e) Découpage de la phrase, nécessité d'une heuristique

Nous avons vu que, avant toute construction de structure, il faut découper la phrase en morceaux, et qu'ensuite, ces morceaux peuvent à leur tour avoir à être découpés. Si nous connaissons les règles de construction des phrases, décrites dans les grammaires, nous n'avons pas appris à l'école de règles de découpage des phrases. Pourtant jamais nous n'aurions idée de découper une phrase en mettant d'un côté un verbe et un article, et de l'autre ce qui suit. En effet, nous avons déduit, même sans le savoir, des règles de découpage des phrases. Si l'ordinateur essaye tous les découpages possibles, il perd de la place en mémoire. Il nous faut donc apprendre à la machine les règles que nous utilisons nous-même pour choisir le bon découpage dès le début, ou presque. A partir des règles de notre grammaire indépendante du contexte, nous pouvons en effet déterminer quelles catégories syntaxiques peuvent se trouver au début d'un syntagme, quelles catégories se trouvent à la fin, ainsi que le nombre de mots que peut alors contenir ce même syntagme.

Chaque fois que le programme doit découper une liste, il utilise donc ces données qui constituent une heuristique, une stratégie qui permet de choisir la meilleure solution, comme nous le faisons naturellement.

f) Le bloc italien

Il nous reste à définir la petite grammaire italienne qui permet de générer les articles et les prépositions italiennes, ainsi que leur contraction. Nous avons vu que toutes les informations nécessaires à la définition de ces mots sont apportées par la détermination d'un syntagme nominal, le(s) nom(s) apportant le genre, le nombre et l'initiale du nom (consonne, voyelle, 's' impur ou 'z'), et l'article ou/et la préposition permettant de savoir si l'on doit avoir une préposition en italien, et laquelle, et si l'article est défini ou non. En fonction de ces cinq données, la grammaire italienne donne la traduction de la préposition et/ou de l'article qui peut être une contraction préposition-article.

g) Ecriture de la traduction

Les opérations que nous avons décrites ont permis de créer une arborescence qui représente la structure de la phrase italienne où, dans chaque feuille, se trouve un mot de la traduction italienne (mot pouvant être vide si, par exemple, on a deux mots en français pour un seul en italien). L'écriture se fait alors simplement en descendant dans l'arborescence et en allant de feuille en feuille pour lire chacun des mots italiens. La traduction est finie.

III) Résultats et améliorations

Nous avons donc réussi à créer une machine qui traduit les phrases répondant aux définitions de la grammaire définie au début, ce qui permet de penser que, en améliorant cette grammaire, nous pourrions faire un traducteur plus efficace. La seule limite que nous trouvons est celle de la place mémoire.

Après avoir atteint notre objectif premier, nous avons donc pu tenter d'améliorer notre programme. Nous avons ainsi ajouté un élément à notre grammaire indépendante du contexte : le groupe circonstanciel. Nous avons ajouté les règles suivantes :

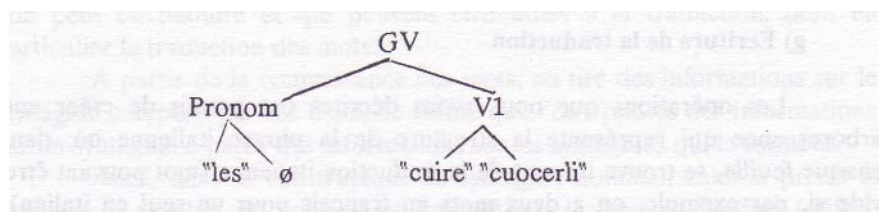
Circ ---> « dans »,
 Circ ---> « sur »,

Gcirc ---> Circ + GN,
 Imp ---> Gcirc + ' , ' + GV,
 Imp ---> GV + Gcirc

Ceci implique bien sûr de nouvelles règles dans la grammaire italienne qui doit connaître l'article du groupe nominal qui suit la préposition circonstancielle pour traduire « dans la » par « nella ». Ici donc la traduction italienne de l'article n'est donnée qu'au niveau du groupe circonstanciel et non plus du groupe nominal.

La deuxième amélioration faite au programme est plus apparente : l'unité de traitement n'est plus la phrase, mais le texte. L'utilisateur peut demander à la machine de traduire tout un texte.

Prendre le texte comme unité de traitement, cela signifie aussi pour nous pouvoir traduire un pronom qui se rapporte à un élément d'une phrase précédente. Nous avons considéré que le pronom se rapporte au dernier groupe nominal rencontré (ce qui n'est pas toujours exact). Le pronom italien prend donc les genre et nombre de ce syntagme. On a alors de plus une inversion et une contraction puisque « Les cuire » donne « Cuocerli ». La représentation en est :



De nombreuses autres améliorations pourraient être apportées à ce programme, mais il faut bien être conscient que, sur 640 KO, on ne peut faire tenir l'équivalent des connaissances linguistiques humaines.

Nous pouvons donc retenir de ce travail que, si l'on réussit à formuler les règles utilisées dans la traduction, le seul problème qui peut rester est celui de la place en mémoire. Cette conclusion donne raison aux linguistes qui font des recherches apparemment folles sur la traduction d'un seul mot ou d'une expression. D'autres recherches, bien plus complexes, sur la représentation des connaissances pourraient permettre de traduire des phrases en fonction du contexte, ce qui peut parfois sembler impossible, tant les connaissances mises en jeu peuvent être multiples et évoluées. Si les traductions de modes d'emploi peuvent être faites par des machines assez simples, les traducteurs d'Italo Calvino ou de Umberto Eco n'ont encore rien à craindre. Ils risquent simplement de trouver une aide précieuse dans ces machines à traduire.

Sarah LABAT

BIBLIOGRAPHIE

VAUQUOIS (Bernard), « La Traduction Automatique à Grenoble », *Documents de Linguistique Quantitative*, n° 24, Paris.

Recueil d'articles choisis par P. VANDEGINSTE, *La recherche en Intelligence Artificielle*, Paris, Seuil, La Recherche, 1987.

MORIN (Jean-Yves), « Théorie syntaxique et théorie du passage : quelques réflexions », *Revue Québécoise de Linguistique*, Québec, 1985, t.14, n° 2, p.9-48.

Reuves : *T.A. Informations*, revue internationale du traitement automatique du langage, publiée avec le concours du C.N.R.S.

RUWET (Nicolas), *Introduction à la grammaire générative*, Paris, Plon, 1967.

BIHAN (Patrice), *Turbo Prolog, une introduction à l'Intelligence Artificielle*, Paris, Eyrolles, 1987.