Multilingual Dialogue-Based MT for Monolingual Authors: the LIDIA Project and a First Mockup

Christian BOITET & Hervé BLANCHON
GETA, IMAG-campus
(UJF & CNRS)
BP 53
38041 Grenoble Cedex 9, FRANCE
boitet@imag.fr, blanchon@imag.fr

Abstract

Dialogue-Based Machine Translation (DBMT) is a new paradigm for translation situations where other approaches, such as the Linguistic-Based (LBMT) and the Knowledge-Based (KBMT) approaches, are not adequate. In DBMT, although the linguistic knowledge sources are still crucial, and extralinguistic knowledge might be used if available, emphasis is on indirect pre-editing through a negotiation and a clarification dialogue with the author in order to get high quality translations without revision. Authors are distinguished from "spontaneous" writers or speakers by the fact that they want to produce a "clean" final message and may be willing to enter into such dialogues. After having described the main situational, linguistic and ergonomic issues in DBMT for monolingual authors, we describe ongoing work on the LIDIA project. The typical translational situation considered is the production of multilingual technical documentation in the form of HyperCard stacks. Notable points in the linguistic design include multilevel transfer with interlingual acceptions, properties and relations, the "guided language" approach (typed textual fragments and lexical preferences), and a TEI-inspired representation of texts and structures. The current mockup, LIDIA-1.0, demonstrates the majority of these ideas on a HyperCard stack, to be translated from French into German, Russian and English. Some of its aspects are discussed in detail, in particular the user interface, the object-oriented implementation, and the production of disambiguation dialogues.

Keywords

Dialogue-Based MT, implicit indirect understanding, interlingual acceptions, guided language.

Introduction

Linguistic-Based Machine Translation (LBMT) is the dominant paradigm among today's commercial MT systems. Knowledge-Based MT (KBMT) is beginning to show its practical potential in situations where a complete domain can be modelled as an ontology. However, Dialogue-Based MT (DBMT) seems to be the only viable approach in situations where "clean" texts are to be translated into several languages, translators are unavailable or too expensive, and the construction of LBMT or KBMT system is also impossible or too expensive. Such situations include the production of relatively small amounts of technical documentation, the exchange of technical notes within a multilingual project over a network (a frequent case in Europe nowadays), the production of written comments from visual or auditory scenes, and even the creation of messages in several languages "without a source text".

The idea of DBMT dates back to the sixties, but previous attempts have not produced usable systems because, we think, the dialogues required users to be specialists, the linguistic coverage was too limited, and cheap, user-friendly and multilingual interactive environments were not yet available. At COLING-90 [9], we outlined the distinctive features of a new incarnation of that approach, which we then called "personal MT", or "MT for the writer". Since then, we have prototyped some aspects of our design, and refined some concepts, in particular our view of the very scope of the approach, which we now prefer to call "DBMT for (monolingual) authors", or, even more optimistically, "DBMT for all". On one hand, "author" is less restrictive than "writer": an author is somebody who wants to create a text, and may do so by writing, speaking, and/or interactively creating it. On the other hand, "author" is more restrictive than "writer", "speaker", "commentator", in that an author wants to create a "clean" final product, whereas the latter terms may refer to persons just wanting to produce "spontaneous" text or speech for the sake of immediate communication, and not ready to enter into any form of heavy dialogue to make it "clean", where "clean" means conforming to some grammar (allowing for incorrect constructs if

they appear in some kind of "natural" text) *and* devoid of any "self-modifying" parts so frequent in spontaneous speech (hesitations, false starts, repetitions, corrections…).

After having presented our approach to the main issues in DBMT for monolingual authors in general, we describe ongoing work on the LIDIA project. In the first phase the typical translational situation considered is the production of multilingual technical documentation in the form of HyperCard stacks. Notable points in the linguistic design include multilevel transfer with interlingual acceptions, properties and relations, the "guided language" approach (typed textual fragments and lexical preferences), and a TEI-inspired representation of texts and structures. The current mockup, LIDIA-1.0, demonstrates the idea on a HyperCard stack, presenting short ambiguous French sentences in context. This stack is translated into three stacks, German, Russian and English. Some aspects are presented in more detail, in particular the user interface, the object-oriented design, and the production of disambiguation dialogues. Although this mockup does not implement all features of the general design, because a complete implementation would have called for considerably more human resources than were available, we feel it demonstrates the potential of the approach and is a first step towards a usable prototype, where the linguistic engineering aspects and the reactions of real users could be studied.

I. A general approach to the situational, linguistic & ergonomic issues crucial in DBMT for MA

I.1. Situations for DBMT

1.1. Motivations

The main motivations for DBMT are the limitations of current MT paradigms, the increasing importance of national languages in the global context of internationalization, and the recent technological advances.

Linguistic-Based MT (LBMT) relies solely on dictionaries and grammars (most often partially procedural) to translate texts on the basis of the linguistic forms. That approach works very well if the goal is only to access information in a foreign language (MT for the "watcher"), because intelligent and usually specialized users are able to get what they want from "rough" machine translations, even if linguists, translators or language teachers all agree on their bad quality. Lexical coverage and speed are paramount. LBMT is also very good for producing "raw" translations, good enough to be revised by professionals, in the context of large amounts of texts pertaining to the same "sublanguage", e.g. weather bulletins or maintenance manuals. But, even in the case of very restricted sublanguages such as that of weather bulletins [16], LBMT cannot be used without revision, and there must be as many revisions as there are target languages. According to figures given by producers of MT systems [20], the creation of a new (operational) LBMT system from scratch costs between 200 and 300 man-years with highly specialized developers. Also, the cost to adapt an existing LBMT system to a new domain and a new typology of texts is in the order of 5 to 10 man-years, which makes it impractical if there are not at least 10,000 to 15,000 standard pages to translate[1].

Knowledge-Based MT (KBMT) has been advocated for decades, before the first prototype, KBMT-89, was built at CMU [28] with an ontology of about 1,600 concepts and 1,200 lexical items in each language. Interaction with a human user through the "augmentor" component [13] was still necessary to solve ambiguities unsolvable by syntax and semantics. More recently, KANT [29], a system derived from KBMT-89, has been developed for translating heavy equipment documentation at Caterpillar, with the aim of suppressing any interaction during processing. However, the price to be paid is a high degree of interaction at creation time, in order to force the input into a strongly controlled language. After about 3 years of development, KANT contained 14,000 word senses for general terms (which correspond to less than 5,000 terms) and a few hundred specialized terms. The biggest part of the effort seems to concern the creation of the ontology. Also, the translations obtained are grammatically and stylistically very good, but would often be rejected by professional translators as inexact paraphrases. Let us take examples given by Nyberg & Mitamura [29]:

[1] Translating and post-editing 10,000 pages typically takes 8 man-years. It can for example be done in a year with 6 translators and 2 post-editors, each working 1,700 hours.

English source	French target	German target
`In order to prevent a fire hazard, do not overload AC outlets.`	`Afin d'éviter tout risque d'incendie, ne jamais surcharger les prises CA.`	`Vermeiden Sie Feuergefahren, indem Sie die Netzanschlüsse nicht überlasten.`
More exact translations (topic and focus have been reversed in German, "tout" should come from "any", "jamais" from "never") .	*Afin d'éviter les risques d'incendie, ne pas surcharger les prises CA.*	*Um Feuergefahren zu vermeiden, überlasten Sie die Netzanschlüsse nicht.*
`If the TV set has been dropped, a shock hazard may exist.`	`La chute du téléviseur peut provoquer un risque de choc électrique.`	`Wenn Sie das Fernsehgerät fallen lassen, kann die Gefahr eines Elektroschocks bestehen.`
More exact translations (semantic and temporal relations have been incorrectly translated, "Sie" should come from "you").	*Si on a laissé tomber le téléviseur, il peut y avoir un risque de choc électrique.*	*Wenn man das Fernsehgerät fallen lassen hat, kann die Gefahr eines Elektroschocks bestehen.*

From these examples, we get the impression that there is still the need for revision, and that the translations are not really better than LBMT translations obtained on similarly controlled inputs. Although the KBMT approach has now been proved to be technically viable, and may lead to a higher asymptotic quality than LBMT, it seems to be less widely applicable than LBMT, because it needs not only linguistic knowledge bases of at least the same degree of detail, but also a world knowledge sufficient to represent the considered domain of discourse in a complete and adequate way. The cost of constructing and maintaining such "ontologies" seems to exceed considerably that of constructing and maintaining the linguistic knowledge bases, even in relatively small domains, and to grow far faster[2]. Also, if one wants to avoid post-editing, it seems necessary to work only on severely controlled languages[3], and not on observed natural sublanguages.

A strong motivation for DBMT is also the increasing importance of the use of national languages in the global context of internationalization. The use of one's mother tongue is not only a political issue, but a matter of efficiency. In cooperative European projects, for example, communication is hampered by the practical need to read and even more to write in English. For the vast majority of participants, writing in English, if at all possible, takes considerably more time than writing in their own language, and the resulting text is often quite difficult to understand or outright unreadable.

Finally, the recent availability of powerful but cheap personal computers, user-friendly interactive environments, and convenient telecommunication facilities, have made DBMT a practical idea.

1.2. Criteria for choosing the DBMT approach

We propose four basic criteria for choosing the DBMT approach:

- quality should be important and revision impossible or costly ;
- the context should be truly multilingual context : 1—>n, as in the dissemination of technical documentation, or n<—>n, as in cooperative international projects ;
- the language or the context should not be too constrained or controlled (otherwise, LBMT or KBMT would be more indicated) ;
- authors should be willing to conduct normalization and disambiguation dialogues ;
- dialogues should be made acceptable by leaving the initiative to the user, by providing ways to control and reduce their importance (using parameter settings, direct insertion of disambiguation marks, etc.), and by letting the user choose between several media.

---

[2] To alleviate this cost problem, one might consider using the same ontology for a variety of applications beyond translation, e.g. for producing multilingual summaries, parts descriptions, etc.

[3] We suspect this to be inherent in the KBMT approach, and not contingent on the KANT project, because all terms and linguistic constructs of the input language must be precisely, unambiguously and completely translated into the ontology and the interlingua before any translation can be produced.

1.3. Examples of likely & unlikely situations for DBMT

a. Written input

Favorable situations involving written input include:

- the translation of relatively small amounts of technical documentation in several languages, typically 5,000 to 8,000 pages to be distributed on one CD-ROM, e.g. in the 9 languages of the EC (and maybe some others, too, like Russian, Arabic, Japanese, or Chinese).
- the broadcasting of information in several languages (e.g. about traffic/weather conditions, or in conferences, sport events, emergencies…), which requires spoken as well as written output.
- the exchange of working notes and documents in international cooperative work using computer networks.

b. Spoken input

Considering that state-of-the-art speech recognition systems still cannot cope at the same time with a large vocabulary, continuous speech, and a multi-speaker situation, there seem to be very few favorable situations for DBMT with spoken input:

- the production of comments or abstracts from visual and auditory scenes (e.g. for subtitling foreign TV sequences).
- the interpretation of very constrained dialogues, such as telephone greetings between parents of children "swapped" between families to study the other language, or telephone assistance services for foreign travellers (visit to a doctor, booking). Here, disambiguation dialogues should not exceed 30% of the utterances[4], and a combination of KBMT and DBMT would be possible.

c. Created input

Finally, there are many interesting situations where the source message is only created to check the content of the message(s) in the target language(s) resulting from a negotiation with an expert (take for example formal letters, which have very different structures in different cultures). Somers &al. [34] have proposed the term of "translation without a source text", but it would perhaps be more appropriate to speak of interactive multilingual text generation rather than of DBMT.

I.2. Linguistic approaches to DBMT

In the majority of the situations just considered, a DBMT system should have a wide coverage, which precludes the use of KBMT techniques, even if the system can in a way "specialize itself" to the particular situation at hand. This gives rise to the following linguistic issues:

- In strongly multilingual situations, the interlingua approach seems indicated. But *how can one overcome the engineering difficulties experienced in the construction of a large interlingual lexicon* by recent interlingua-based Japanese projects (ATLAS, PIVOT, EDR, CICC)?
- *How can one achieve the very large coverage needed?* Typically, an LBMT system contains from 3.10^4 to 3.10^5 terms, usually in 2 languages. The case of METEO ($3.10^{3)}$ is atypical, reflecting a very restricted domain. But a DBMT system aiming at the general public and not restricted to a particular domain might need from 3.10^5 to 3.10^6 terms, in several languages!
- Knowing that good results are only obtained on restricted languages, *how can one construct a linguistic knowledge base usable as a union of sublanguages?* Is it possible to separate the grammatical and lexical aspects?
- Finally, it is crucial that non-specialists be able to easily understand the questions of the system, and possibly ask it about the reasons for some questions, and understand the answers. Hence, a big (and new) issue is *how to make the linguistic knowledge base of a DBMT system accessible to the naive user.*

Although our answers to these questions are certainly not complete and even definite, they are founded on a long practice of LBMT and on an extensive study of many past and present MT systems and projects, in particular those relying on interactive disambiguation.

---

[44] This is the maximum reported clarification rate in human interpretation.

2.1. Multilevel transfer with interlingual acceptions, properties & relations

In classical multilevel transfer, in the sense of Vauquois [38, 39], the source abstract structures produced by analysis are decorated trees, where some attributes are universal (logico-semantic relations, semantic features, abstract time, discourse type…), while others are language-specific (morphosyntactic class, gender, number, syntactic functions…). In particular, the lexical attributes (form, lemma and "lexical unit", or derivational family) are language-specific.

We propose to go one step further towards interlingua by adding the level of *interlingual acceptions* (or "word senses"), but we stop short of constructing a language-independent ontology, where concepts have to be explicitely defined. The underlying multilingual lexical database (MLDB) then contains one monolingual dictionary for each language handled by the system, and one interlingual dictionary for the interlingual acceptions. Each interlingual acception is reflected in each monolingual base, with an appropriate definition in the language of the base, so that interactive sense disambiguation is possible[5].

Note also that "interlingual" does not mean "independent of *all* languages", but rather "intermediate between *some* languages", namely those considered in the MLDB. For example, if the system works with English, French and Russian, there will be only one acception for "wall" as a concrete object. As soon as we add German or Italian, however, we have to add the refinements "wall as seen from outside" (Mauer, muro) and "wall as seen from inside" (Wand, parete).

Given the number of lexical items to include in a universal system, the engineering problems are staggering. However, the recent development of large-scale MLDBs by EDR[6] and the CICC project[7], plus the encreasing interest in MLDBs in Europe (ACQUILEX, GENELEX, MULTILEX European projects), are encouraging for the future.

2.2. Suboptimization: the "guided language" approach

a. The notion of sublanguage in "sub-optimized" MT for revisors

The notion of "sublanguage" in MT has been introduced and studied by R. Kittredge [22, 23] after his experience in MT as director of the TAUM project in the seventies. He gave a number of criteria, mainly lexical and syntactic, to evaluate the difficulty of a "sublanguage" in the context of MT for revisors, and applied them to a number of corpuses to determine whether they belonged to sublanguages and could be handled with MT systems using the "second generation transfer approach with suboptimization".

Kittredge considers the lexical and grammatical aspects, the more or less strong link with a closed semantic domain, and the possibility of writing a text grammar handling more than a sentence. He proposes the formal notion of "lexical closure", which roughly means that the number of new terms encountered in a new page diminishes quickly and tends towards zero or a very low value when the number of pages increases. But there is no such formalized notion for the grammatical aspect: a "sublanguage" is experimentally defined as the set of sentences (or utterances) produceable in fixed conditions (e.g. weather bulletins, calls for tenders of an official body, maintenance manuals of a certain plane…).

b. Separate the lexical and grammatical aspects of sublanguages in DBMT "for all"

Although this analysis is very complete, and quite adequate in the context of LBMT with "sub-optimization", it is inadequate in our context:

- Kittredge's notion of sublanguage combines grammatical and lexical restrictions. But, if use of (DB)MT is to be generalized, it will be impossible to produce and maintain collections of large lexical and grammatical data bases for each type of use. As the dictionary must

[5] It is also possible to *explain* to the writer why such a question is asked, and even to show the words in question. The introduction of "self-learning" aspects in such systems would make them more acceptable.

[6] 300,000 terms in Japanese and English: 100,000 terminological terms corresponding to 100,000 concepts and 200,000 general terms corresponding to 300,000 concepts, with an intersection of 60,000, giving a total of 640,000 concepts.

[7] This international MT project, supported by the Japanese ODA, uses the interlingua approach and the same dictionary design as EDR. We understand it has produced about 60,000 entries in Japanese, Chinese, Malay, Thai, and Indonesian (considered different from Malay and developed independently), all related through a common concept dictionary.

ultimately be as exhaustive as possible, there is no point in using the notion of lexical closure to limit it. The same is true of the grammatical aspect.

- A sublanguage of a language is not a subset of that language ("English" means just textbook English, and not the union of all sublanguages of English). But we would like to work with set-theoretic models of sublanguages, so that, at the simplest level, the sublanguage of a class of utterances[8] can be defined from others by simple operations such as intersection and union.

We propose, then, to separate the lexical and grammatical aspects of sublanguages. A further use of the "divide and conquer" technique is to define them first "coarsely" with a simple symbolic formalism, and then add parametric constraints to allow for finer granularity.

Although our proposals in this direction are far from complete, and have not yet been implemented, let us present them in more detail, in the hope to trigger useful suggestions from other researchers.

c. Lexical preferences

The MLDB contains a "symbolic" skeleton, with terms, acceptions, etc. viewed as nodes, and relations viewed as arcs, in a very classical way. Relations include thesaurus relations, in particular synonymy and quasi-synonymy, as well as a hierarchy of generality (an acception can be more specific or more general than another one), and the interlingual equivalent of lexico-semantic relations (entity—>quality, action—>Arg.i of action…).

Weights are then attached to acceptions as well as to relations between them. They may be stored as *lexical profiles.* The idea is that each document or user, or each class thereof, shares the symbolic information of the same MLDB, but has its own lexical profile. As time goes on, each lexical profile may vary as a result of interaction, allowing for automatic tuning and for the definition of new lexical profiles reflecting lexical preferences. Weights on acceptions should then reflect their "degree of pertinence" to the task at hand, and might be used to indicate the preferred term among a cluster of (quasi-)synonyms (e.g. airplane, aircraft, ship, plane). They might also be used for word-sense disambiguation, as shown by [41].

Note that, in the context of DBMT, where the ultimate goal is a single system for all users, the lexical database should contain a great variety of terms, even incorrect or dubious, whereas terminological databases are usually restricted to normalized or recommended terms.

d. Text types: "utterance types" and "text genres"

As far as the grammatical aspect is concerned, we propose again to break the problem in two, by defining *utterance styles* for sentences or other individually translatable utterances (titles, homogeneous elements in long enumerations…) and *text genres* for the longer texts[9]. A given text type would then be defined by a set of lexical weights relative to the MLDB and as belonging to an utterance type or a text genre with some numerical restrictions. This, we hope, will considerably reduce the number of ambiguities produced by the analyzer, and consequently also reduce the amount of interactive disambiguation.

Suppose that we have collected a very large set R of declarative "rules", expressing all the well-formed constructs of the language at hand, including incorrect or dubious ones, provided they appear in real texts. Any simple declarative formalism can be used[10]. An *utterance style* is then a subset of R, together with appropriate parametric constraints (on, say, the degree of embedding, ellipsis, or coordination). For example, M1 could be the style of simple active sentences without subject (such as: "is used to save a copy of your file on disk"), frequently found in technical documents, and M2 the style of simple explicative sentences.

We want our *text genres* to be testable in the future by SGML-based[11] editors of structured documents, which don't have the severe length limits of current linguistic tools. Hence, we propose to define a text genre as an algebraic expression on utterance styles. For example, the text genre S1

---

[8] Such as "interrogative subordinate clause used alone as title" (e.g. "How to prepare French fries").

[9] In [9], we used for the same notions the terms "microlanguage" and "sublanguage", which proved overloaded and counterintuitive.

[10] For example, CFG with or without attributes, TAG, STCG [12, 40, 48]. In our current mockup, we use ROBRA and test in each transformational rule whether the expected utterance style is one of the utterance styles containing that rule.

[11] Standard Generalized Markup Language [37].

of paragraphs beginning with a sentence of utterance type M1 followed by a (possibly empty) sequence of sentences of utterance style M2 could be defined in SGML by:

```
<!ELEMENT     S1         - 0   (M1, M2*)                          >
```

while the text genre of a document beginning by a title (utterance style M3), and containing a non-empty list of paragraphs (M2 or M3) and/or sections of the same structure could be defined by:

```
<!ELEMENT     Document   - 0   (Title, Content)                    >
<!ELEMENT     Title      - 0   (M3)                                >
<!ELEMENT     Content    - 0   (Paragraph | Document)+             >
<!ELEMENT     Paragraph  - 0   (M2 | M4)+                          >
```

Symbols following `<!ELEMENT` denote text genres, "|" indicates alternation, "*" and "+" repetition. Parametric constraints on attributes associated with the main symbols might be added.

More research needs to be done on how to guide writers towards selecting a particular text genre or utterance style, so that text critiquing, negotiation and clarification can be performed efficiently.

2.3. Towards text encodings suitable for direct pre-editing in a multilingual setting

The basic idea of DBMT is to replace post-editing by indirect pre-editing. That means that the text will be enriched, normally indirectly, through interaction with the author. But experienced users may well want to do part of that pre-editing directly, in order to bypass lengthy dialogues. This is why we represent a text, possibly enriched with disambiguation marks and analysis results, as a *portable* and *readable* string of characters. The tags, entities and transcriptions, should be defined in the spirit of the Text Encoding Initiative (TEI).

a. Multilingual text encoding in a universal character set

The first problem encountered is how to handle texts in various languages using different scripts. Until the eighties, computer systems offered a poor choice of character sets (such as roman without diacritics, mixed cyrillic/roman without lower case), so that the use of transcriptions was mandatory. Ten years later, almost all computer makers started to offer extended character sets and "localized" operating systems. Mac.OS-7.1, available since the end of 1992, is the first ever fully multiscript[12] operating system: with any text processor using the standard Script Manager, it is possible to write a document containing parts in almost all European languages, as well as Arabic, Japanese, Chinese, etc. This was one of the reasons to develop our author's station on a Macintosh.

However, Mac.OS-7.1 is still a unique case[13], and the problem remains in full force if one wants to exchange textual material across computers of different brands, or simply transmit them through electronic networks. For example, French ASCII is not the same on a PC and on a Macintosh. And we will later advocate a distributed architecture, using possibly different computers and operating systems for the MT servers and the author's worksations.

Our solution is to use roman transcriptions for the internal representations of texts, grammars and dictionaries. A transcription consists of a character set and a method for representing textual material (not only the words, but the text structure, the layout, and possibly other information) using only these characters. Which character set and which method to use depends on whether portability or readability is preferred.

For MT proper, we have long used a basic character set almost identical with that of PL/I (no lower case letters, no diacritics, and only the usual punctuation marks and a few special signs). This gives total portability[14], at the expense of readability.

For example, `'*A!2 *NOE!4L , *MAC*ALLISTER VA AUX **USA'`
codes `'À Noël, MacAllister va aux USA'.`

[12] Mac.OS-7.1 is not yet multilingual itself: although all programs are language-independent, each distributed version has the messages and other language-specific resources in only one language.

[13] Xerox Star™ (or "Documentor") document preparation system has been the only truly multilingual text processor until WinText™ became available on the Macintosh in 1987, but the underlying operating systems were respectively stricly monolingual, or only localizable. No OS but Mac.OS-7 is yet fully multiscript.

[14] In countries such as Thailand, lower case roman letters are replaced by a local character set in bilingual terminals.

For DBMT, readability is more important, and we can use upper and lower case letters. In our transcription, diacritics are still represented by "special sequences" introduced by "!" because, in technical documentation, parts are often referred by identifiers where letters and digits are mixed (e.g., XA1). Information relative to the structure or the layout of the text is represented by tags, or "markups", in the spirit of SGML and TEI (`<parag>`, `<section>`, `<greek>`, etc.). The same mechanism is used to indicate a change of language and/or character set (some languages use more than one).

b. Encoding of linguistic information obtainable by (indirect) preediting

Special fixed phrases should be transformed into special strings, in order to help analysis and translation. For example, "Hide Balloons" could become `&FXN_Hide_Balloons`, which can be treated by an appropriate morphological subgrammar.

After word-sense disambiguation, we have to attach to each occurrence its appropriate sense number in the MLDB, e.g. `glace.1` for "ice cream" and `glace.2` for "mirror". But that is not very readable and precludes direct preedition. Hence, we propose to optionnally add to the sense number a "semantic key", usually another word or term, such as a fragment of the corresponding definition, getting such encodings as `glace.1=aliment` or `glace.2=miroir`.

As more than one semantic key can usually be associated to an acception, e.g. `glace.1=a_manger` or `glace.1=dessert`, we would like to allow a user to enter `glace=dessert`, and to have the system look up the MLDB, find that the acception of "glace" nearest to those of "dessert" is sense 1, and transform `glace=dessert` into `glace.1=dessert` or even `glace.1=aliment`.

Annotations concerning grammatical information relative to words, like category (verb, noun…), number, gender, case, tense… and even syntactic function (subject, object…) are attached to the occurrences in a similar fashion.

The last type of annotation concerns the tree structures. To delimitate syntagmatic groups, we can use special brackets such as `{&rel…}` for a relative clause[15], or simply `{…}` if the syntagmatic category is not known or too arcane for general users (e.g., 'adjectival group', or 'cardinal group'). To represent an anaphoric link, we attach to a pronoun a copy of its referent. In the case of elision, we add hidden words `(centrale .&eld=inertielle)`. Further grammatical and semantic information may also be attached to non-terminals and terminals. For example, before full disambiguation, "Devant cette somme, il ne rend pas sa glace" could have the following intermediate representation:

```
{ {&grd,cause *devant.&vrb cette somme.2&nf , } il ne rend.2 pas=ne sa glace.1 }
```

which would disambiguate between "Owing this sum, he doesn't vomit his ice cream", and, among others, "Facing this summa [Opus Magnum], he doesn't give back his mirror".

The main point here is that, in the views accessible to the user, *the system of annotations concerns several levels of linguistic description, but is incomplete at each level*, because no unfamiliar notion should appear. For example, 'verb' is a concept familiar to almost every literate adult, but not 'modal verb'. At the level of functions, 'subject', 'object' and 'complement' are familiar terms, but not 'attribute', 'epithete', 'head', and so on.

To make the discussion more concrete, let us take as example the following 3 sentence text contained in a field of Ariane documentation stack.

```
Un processus de traduction en ARIANE-G5 se compose d'une suite de trois étapes
(analyse, transfert et génération). Chaque étape est constituée d'une suite de
différentes phases de traitement. Chaque phase est relative à l'emploi d'un LSPL
précis.
```

Here is a reduced view of an associated tree structure obtainable after optional preediting, analysis and interactive disambiguation.

[15] Or we add "rel" to the information attached to its head, as in the following example ("compose.&v,phvb").

```
{ { *UN.&art PROCESSUS.&n,suj { DE.&prep TRADUCTION.&n,comp { EN.&prep **ARIANE-
G5.&np,comp } } } SE.&refl COMPOSE.&v,phvb { D'UNE.&art SUITE.&n,obj1 { DE.&prep
TROIS.&card E!1TAPES.&n,comp { (.&lp ANALYSE.&n,app ,.&ponc TRANSFERT.&n,coord
ET.&cjcoord GE!1NE!1RATION.&n,coord ).&rp } } } ..&ponc } { { *CHAQUE.&art
E!1TAPE.&n,suj } EST.&v,aux CONSTITUE!1E.&v,phvb { D'.&prep UNE.&art
SUITE.&n,comp { DE.&prep { DIFFE!1RENTES.&adj,epit } PHASES.&n,comp { DE.&prep
TRAITEMENT.&n,comp } } } ..&ponc } { { *CHAQUE.&art PHASE.&n,suj } EST.&v,phvb {
RELATIVE.&adj,atsubj { A!2.&prep L'.&art EMPLOI.&n,obj1 { D'.&prep UN.&art
**LSPL.&np,comp { PRE!1CIS.&adj,epit } } } } ..&ponc }
```

Internal encoding may seem to be a second-order low level technical question, but there is more than meets the eye. Not only is it very important for the developers of grammars and dictionaries, but, to design it in a meaningful way, one has to understand the internal workings of an MT system, and in our case to satisfy the *accessibility constraint:* the normally hidden pre-editing marks have to be understandable by general users if they want to put some in themselves. It is also a challenge for linguists accustomed to making very fine-grained distinctions to prepare systems using only coarse information obtainable from non-specialists, yet it must be done if this sort of approach is to work.

c. Towards "self-explaining" documents

There are other potential uses of such annotated forms beyond analysis in DBMT. First, the same kind of annotated form can easily be generated in each target language. This answers an important objection to DBMT, namely, that it might be impossible to generate sentences or utterances reflecting the disambiguating choices of the author.

In a more general setting, keeping the annotated form of a document attached to the document would make it "self-explaining". Readers could could ask what any given part is supposed to mean. Such self-explaining documents could be very useful in all situations where the use of controlled languages is currently advocated to make texts less ambiguous. This approach would make it possible to produce completely unambiguous documents, which is very difficult even with controlled languages, without forcing the author into an unnatural kind of expression.

I.3. Ergonomic issues

Finally, the ergonomics of a DBMT system will be a crucial aspect. Here are some related issues:

- Should the system aim at real time performance, or is asynchronicity preferable?
- Should it run on low-priced personal computers, or on workstations? Is a server architecture possible? If so, could we simply hook a PC to the minitel[16] and use it as a DBMT author's workstation?
- How should the dialogues be organized? Is it necessary and/or possible to conduct them in a multimedia environment? More specifically, can the use of speech synthesis improve the efficiency and user-friendliness of disambiguation dialogues?

a. Asynchronicity

We advocate an asychronous kind of organization, analogous to that of CRITIQUE [30], for ergonomic and practical reasons. First, real time behavior would be called for only if we would want the user to answer immediately the questions asked by the system, which we don't. It would also appear somewhat counterproductive to immediately ask questions on a text under writing, of which the author knows perfectly well that it is not yet in a correct and achieved state.

Second, DBMT systems should be usable on cheap personal computers. Taking into account the inherent degree of complexity of any general, good quality DBMT system, we cannot hope for real time execution on these machines, although it might be achieved on powerful workstations.

b. Distributed architecture

Real time execution is also impossible with a distributed architecture, which we prefer. Obviously, using a powerful server to handle all or some of the non-interactive translation-related processes is

[16] There are about 6 millions minitel terminals in France, the majority of them being used at home for a variety of services, ranging from telephone directory assistance to train seat reservation to mailing services… and even to MT, proposed by Systran SA. It is very easy to set up a server. Fees are collected directly by France Telecom, included in the telephone bill, and redistributed to the server organizations.

preferable to storing a large system on each author's PC, and running it in the background, not to speak of all maintenance problems.

c. Non-preemptive, multimodal interactive disambiguation

One of our main guidelines is to leave the author completely free to decide when to enter into a dialogue requested by the system. We feel this to be essential for acceptability. Conversely, previous attempts at DBMT may have failed partly because of the modal character of the dialogues.

II. LIDIA-1.0, a first running mockup

II.1. Design choices

1.1. Selected type of situation

In the first phase of the LIDIA project, we are considering a particular situation, where a monolingual French engineer is supposed to create technical documentation, in the form of a HyperCard stack, on a middle-range Macintosh, and to help the system translate it into English, German and Russian. Our architecture is distributed: author's workstation on a Macintosh and MT server on a mini—IBM-4361 running the Ariane-G5 environment [8]..

1.2. Why HyperCard?

The choice of HyperCard is motivated by the fact that hypertexts are becoming popular supports for technical documentation, and on the assumption that writers will more readily agree to participate in a dialogue if the tool they are using is very interactive than if they use a more classical text processor. Finally, there are some linguistic advantages. First, the textual parts are clearly isolated, and not cluttered with images, formulas, tabs, markups, etc. Second, the textual parts may be typed, thus greatly facilitating analysis. For example, a given field may contain only titles, another only menu items, another only sentences without the initial subject (which is often contained in another field), etc. Examination of existing stacks suggest two levels of typing: *utterance styles* in the case of short textual fragments (sentences or phrases), and *text genres* (such as explanation, advice, commented program…) for longer textual fragments.

HyperCard documents are interactive "hypertexts", called "stacks", through which users can navigate. A stack is a collection of cards.

HyperCard interacts with the user by displaying one card at a time. A card has its own buttons and fields, and a background which in turn has buttons and fields. Buttons are "hot spots", which trigger actions if "clicked". Fields contain editable text.

A background may be shared by several cards and a stack can have several backgrounds. A card overlays its background (both have the same size). Pictures may be drawn on cards and backgrounds.

HyperCard offers tools for incrementally and interactively constructing stacks.

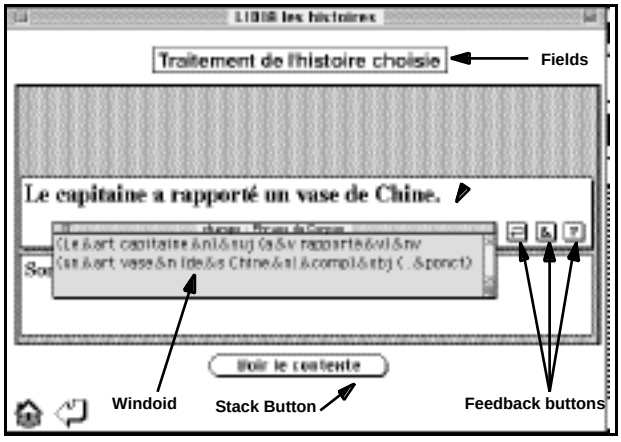

Figure 1: a HyperCard card and its objects

1.3. Organization of the automatic and interactive processes

The main processes are illustrated in figure 2 below. Many other organizations would be possible. We have settled on this one, which is relatively simple, for practical reasons only.

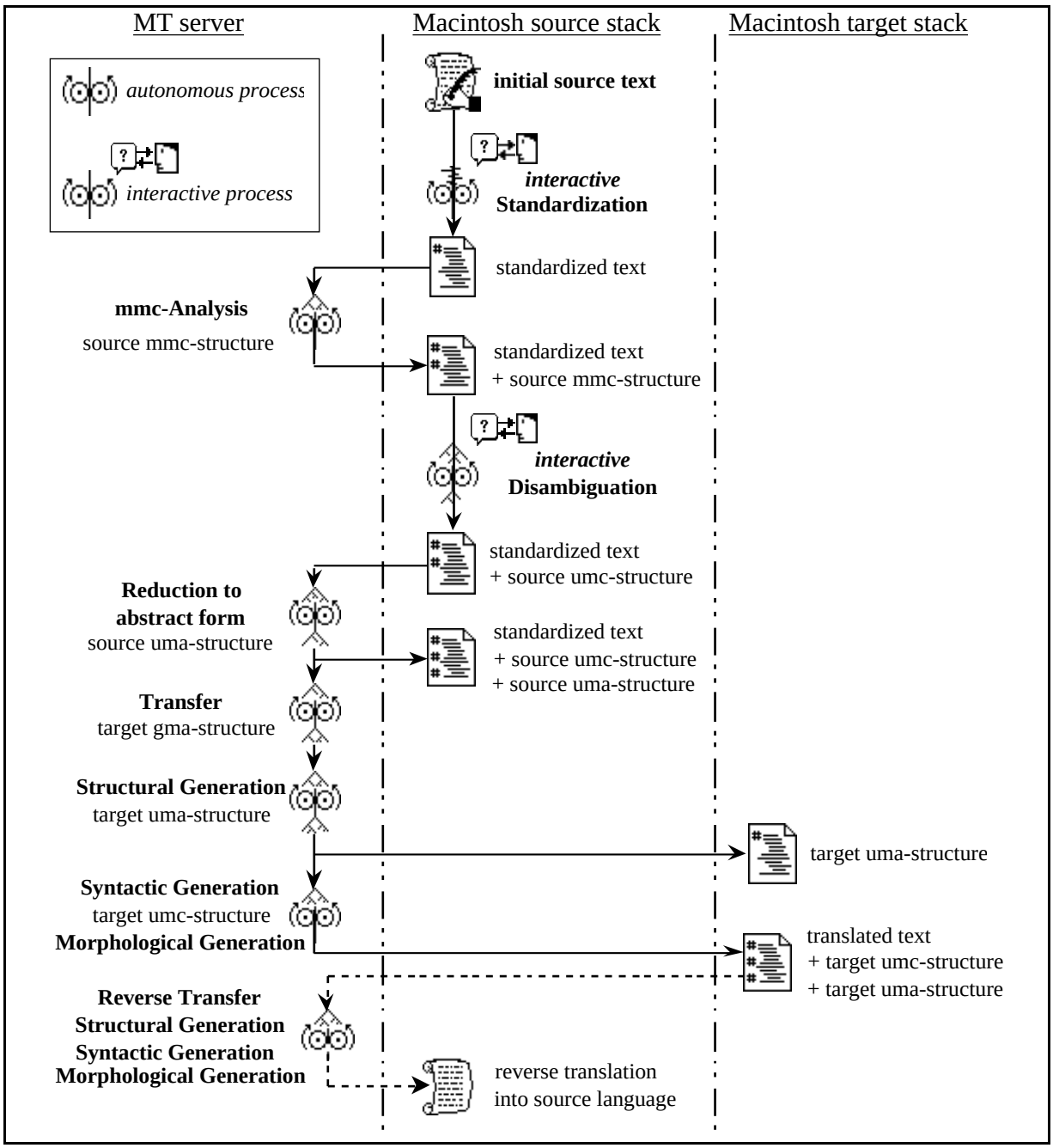

Figure 2: General organization of the translation process in LIDIA-1

1. The process of "standardization" aims at helping the automatic translation process, thereby reducing the required interaction. After standardization,

- all fields and buttons should be assigned a text type[17], in order to control stylistic checking and later on to help analysis *(text categorization module);*
- the texts should be well spelled, and conform to the stylistic parameters associated with the type of their containers *(grammar & style checkers);*

[17] In the case of "incomplete" texts, where for example the subject of the first sentence is contained in another field (as in tables containing command names and their explanations), this module also asks how to construct the complete text.

- fixed phrases behaving in special ways (such as the menu item Hide Balloons in "Hide Balloons turns Balloon help off") should be marked (the user should assist the *special fixed phrases module* in the construction of the list of fixed phrases used in the stack)[18];
- terminological preferences (e.g., between "plane", "aeroplane", and "aircraft") should be enforced by the *lexical preference module,* whether they loosely reflect local writing habits, or are mandatory because of the domain itself, or of some normalization effort.

2. The standardized text is then analyzed on the translation server. The internal representation of the *source mmc-structure* (Multisolution, Multilevel and Concrete) produced is then transformed into an external form directly readable by Lisp and sent back to the Macintosh.

3. The source mmc-structure is used to produce the disambiguation dialogue on the Macintosh. After disambiguation, it becomes an unambiguous *source umc-structure* (Unisolution, Multilevel and Concrete) corresponding to the analysis chosen by the author[19].

4. The source umc-structure is then abstracted, or "reduced" to a *source uma-structure* (Unisolution, Multilevel and Abstract).

5. The system then produces the *target gma-structures* (Generating, Multilevel and Abstract), using adequate transfer components. A gma-structure is in a way more "general" and "generative" than a uma-structure, because its surface-oriented levels (syntactic functions, syntagmatic categories…) may be empty, and if not are just preferences indicated by the transfer.

6. Structural generation produces a *target uma-structure* homogenous with what would be the result of analyzing (and disambiguating) the target text to be generated. It consists in choosing the paraphrase to be generated by computing the surface levels and a first approximation of the word order from the deeper levels (logical relations, semantic relations, semantic features, etc.).

7. The transtation process ends with syntactic generation and morphological generation. When all objects of the source document are translated, we get image stacks in the target language(s).

8. The target uma-structures may be used to help the (monolingual) user control the translation by performing a reverse translation into the source language.

II.2. LIDIA-1.0: choices, limitations, and a demo

2.1. Choices and limitations specific to LIDIA-1.0

a. Translatability constraints on HyperCard stack

How the user interacts with HyperCard is defined by a set of preferences. We have added a new preference item (a check box) to start or stop LIDIA.The structure of the stack must be fixed before invoking LIDIA, which means that LIDIA-1.0 can not track the creation or deletion of objects.

We have also put some restrictions on the structure of "translatable" stacks, which ensure that we don't need to translate the scripts[20], but only the names of the buttons and the textual contents of the fields[21]. Hence, the scripts should be written in a language-independent way. This implies that:

---

[18] The two preceding modules can work directly with the text as written by the author. From here on, however, the system works on a transcription contained in the "shadow record" associated with the card, as well as with intermediate forms of processing. This forces us to lock the original textual field (unless the author decides to change it and is willing to start all over again).

[19] "Concrete" means that the original text can be read from the structure in a direct way (by inorder traversal of the leaves in constituent structures or of all nodes in dependency structures). The nodes and/or edges of the structure may contain "surface" information as well as "deep" information (predicate/argument organization, semantic relations…). In "abstract" structures, negations and auxiliaries may have been suppressed as nodes and represented in decorations, elided elements may have been restored, ordering may have been normalized, etc.

[20] Every HyperCard object has a (possibly empty) *script*, written in the HyperTalk programming language. A script is a collection of *handlers*, each of the form "on <message> do <sequence of HyperTalk statements> end". A handler is *invoked* when its <message> (such as a mouse click) is received by the object whose script contains it.

[21] The assumption is that a stack is fully translated into as many stacks as there are target languages. Another possibility would be to make the stack multilingual by creating a copy of each text container for each language.

- messages should never be contained within scripts, but always retrieved from normal fields[22], which will be made invisible in the final version of the stack.
- button references should not be names, but numbers, invariant in translation;
- text drawn within pictures will not be translated;
- any customized version of the menu bar will have to be translated by hand;

b. Standardization and disambiguation dialogues

In this mockup, no standardization tools such as spell-checker, style-checker, and text categorizer have yet been included, although some preliminary experiments have been done with "Le Correcteur 101" by Machina Sapiens. There is only a very primitive typing of textual elements: Normal, Title, and Don't_Translate.

Dialogues are conducted only through the screen. We hope to experiment with the introduction of speech synthesis in disambiguation dialogues in the future. In order not to overload the users with new things to learn, we have also preferred to stick with menu-driven dialogues. For example, we have considered using graphic manipulation of structures (which might be represented as graphs, or as embedded boxes) for structural disambiguation, but potential users found it more difficult than to choose between textual rephrasings (examples below).

c. Linguistic coverage

The lexical coverage of the mockup is rather small: 134 French lemmas, corresponding to 526 acceptions, 304 English lemmas, 370 German lemmas, and 394 Russian lemmas. However, the entries have not been simplified in any ad hoc way. There are in fact a lot more lemmas for each of these languages under Ariane-G5 (≈30,000 for Russian, 10,000 for French and English, 5,000 for German), but they are not yet coded for the acceptions.

The grammatical coverage is medium. Not all constructs are treated, but a lot more are treated than appear in the demo corpus and in the test corpus. Also, grammars are not ad hoc. The test corpus comprises textual fragments (phrases, sentences) taken from some experimental HyperCard stacks. The demo corpus is quite small (10 stories, each consisting of 2 or 3 sentences).

2.2. A demonstration stack

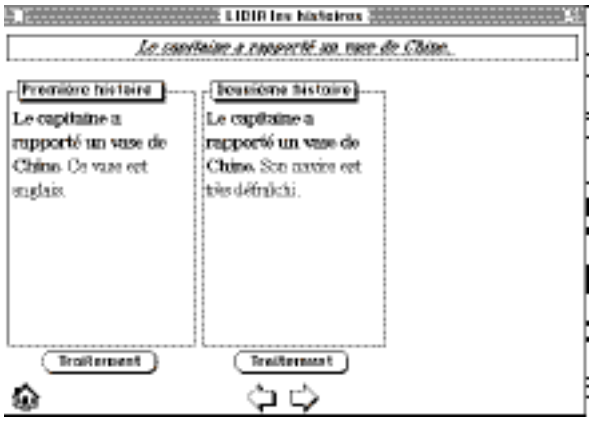

Figure 3: a story card[23]

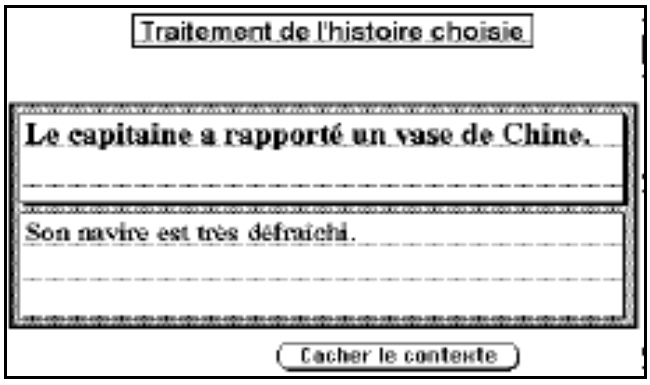

Figure 4: a sentence card

The demonstration stack, 'LIDIA les histoires', has two types of cards, story cards and treatment cards. The idea is to show that interactive disambiguation is both necessary and possible, by putting some ambiguities in contexts where no expert system could reliably solve them, and they would not carry over to the target languages.

[22] However, they may contain variables: "Please locate file &1" should certainly be translatable).

[23] The story of the left can be translated as: '**From China, the captain has brought back a vase**. This vase is English'. The second story can be translated as: '**The captain has brought back a Chinese vase**. His boat is soiled.'

A story card is a collection of two or three stories having an ambiguous sentence in common. The author is supposed to solve the ambiguities through his understanding of the stories. Each story is presented in a sentence card, where the context of the ambiguous sentence may be shown or hidden.

The DBMT process is activated by asking for the translation of any field of a sentence card. Note that the user is never interrupted by a question. Objects show they are waiting for answers, and the user decides which question to answer and when.

2.3. Example session with LIDIA-1.0

The author checks the LIDIA HyperCard preference item. The Macintosh starts to periodically connect to the MT server, in much the same way as a mail utility. The user may continue to navigate in the stack and edit fields. When s/he decides that some objects (fields, buttons) are ready to be translated, s/he chooses the translation tool in the LIDIA palette, selects the objects with LIDIA' selection tool ✓ (fig. 6), and continues to work while translation-related processes execute. The translation status of any object can be monitored through its a *status watcher*, which is automatically created when the object is selected. When clicking on it (fig. 7), a windoid or pop-up window appears (fig. 8).

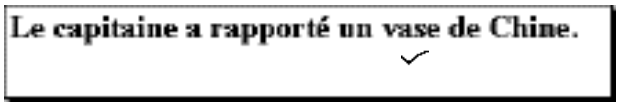

Figure 6: selection of an object

Figure 7: status watcher of a field

The task in progress is displayed in bold, the previous ones in plain, and the following ones in italic. Thus, in figure 8, the system is currently analyzing the text fragment.

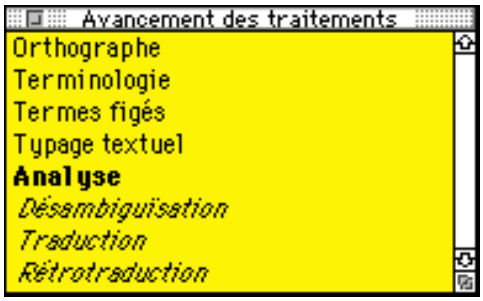

Figure 8: status watcher windoid

When intervention by the user is required, LIDIA sends out a (parametrizable) signal, exactly like background utilities such as PrintMonitor or Mail. The user is free to interact at that point or later.

To interact with an object (as small as a button or as big as the entire stack), one simply double-clicks on its status watcher and selects the appropriate item in a pull-down menu. The interaction mode can be exited at any time.

After analysis, the sentence may have to be disambiguated. A new item is added to the menu Message and a naw button appears over the concerned object as in figure 9. The user can choose to interact at once or later.

Suppose the user clicks on the  button. A first question appears (fig. 10). In the context of this story, the user should choose to attach 'de Chine' to 'vase' (Chinese vase). A second dialogue appears (fig. 11) to ask about the word sense of 'capitaine'.

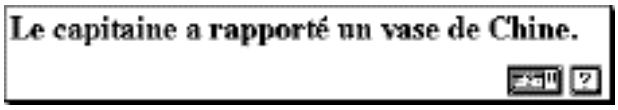

Figure 9: questions on an object are waiting

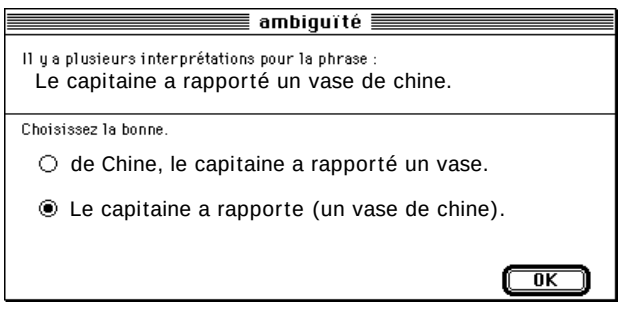

Figure 10: attachement problem (story 2)

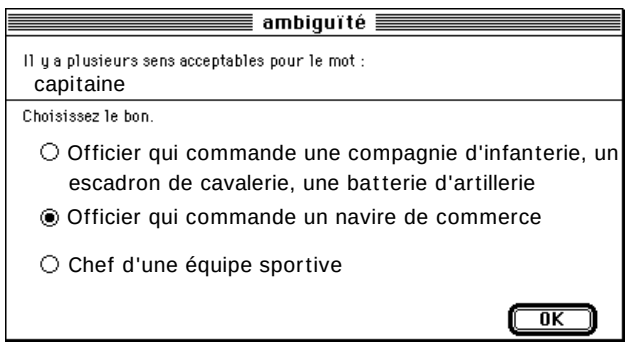

Figure 11: word sense disambiguation (story 2)

These word senses are retrieved from Parax, our HyperCard mock-up of a multilingual lexical database [33].

That is all for this sentence. To see the annotated form of the text, one selects the appropriate item in the LIDIA menu. In LIDIA-1.0, we produce only one view, containing the syntactic class of each occurrence and the syntactic function of each phrase (see fig. 12).

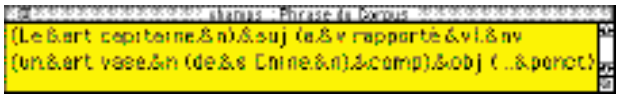

Figure 12: view of theannotated form

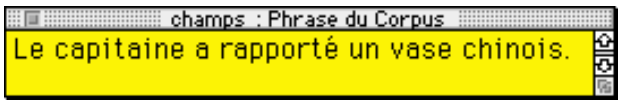

Figure 13: reverse translation[24]

To check the translation produced in a target language, the user can ask for the "reverse translation", produced from the abstract structure (uma-structure) of the target text. The example concerns the second interpretation of the example (fig. 13).

The system finally updates the corresponding story card (fig. 14) in the target stack.

Figure 14: translations into German

III. Some aspects of LIDIA-1.0 in more detail

III.1. User interface

1.1. Preferences

There are four preference profiles, concerning the task, the communication and MT servers, the user, and the lexical resources (fig 15).

Task-related preferences are shown in the figure: the user selects the active target languages, the translation unit (selection, card or stack), and the desired component processes: spelling and style checking, terminological normalization, special fixed phrases, text categorization, and translation[25].

User preferences concern the feed-back type and the dialogue level. The lexical profile determines the active spellchecker, personal dictionaries, and thesaurii.

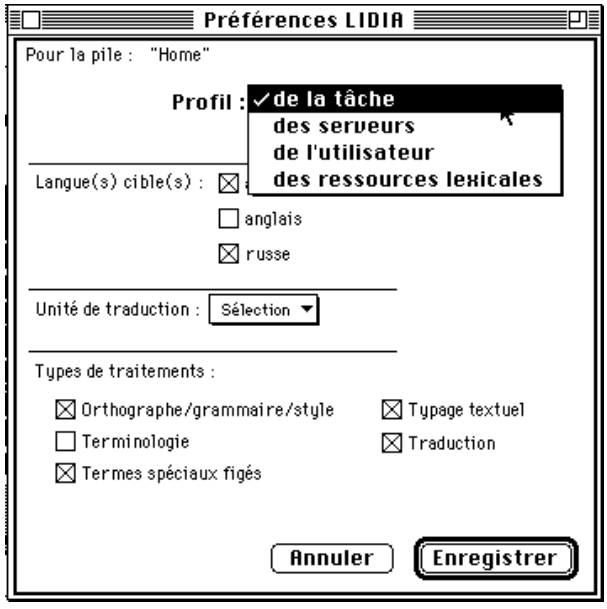

Figure 15 : LIDIA-1 preferences

1.2. Menu

Once the preferences have been defined, the author uses a menu and a palette to interact with LIDIA.

The interaction with the author is made through the LIDIA menu (fig. 16), the Messages menu, a palette (fig. 17), feedback buttons (fig. 1) and windoids (fig. 1).

[24] 'The captain has brought back a Chinese vase.'

[25] The only component process now available in the mockup is translation.

The menu shown here offers 8 choices: process the selected object according to the set of preferences, process some object with a particular preference set, show the treatments' progress, show the reverse translation, show the annotations, show the palette, modify the preferences and build the target stacks.

Figure 16: LIDIA-1 menu

1.3. Palette

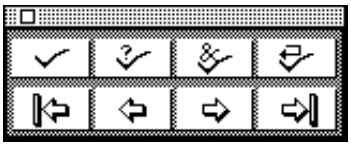

Figure 17: LIDIA-1 palette

The user can trigger the most frequent treatments by using the LIDIA-1 palette. The first line contains the LIDIA tools (process the selected object, show the treatment progress, show the annotations and show the reverse translation), and the second line the most frequent browsing tools.

III.2. Implementation issues

The implementation is characterized by the use of a distributed architecture, a whiteboard approach, and object-oriented techniques.

2.1. Distributed architecture

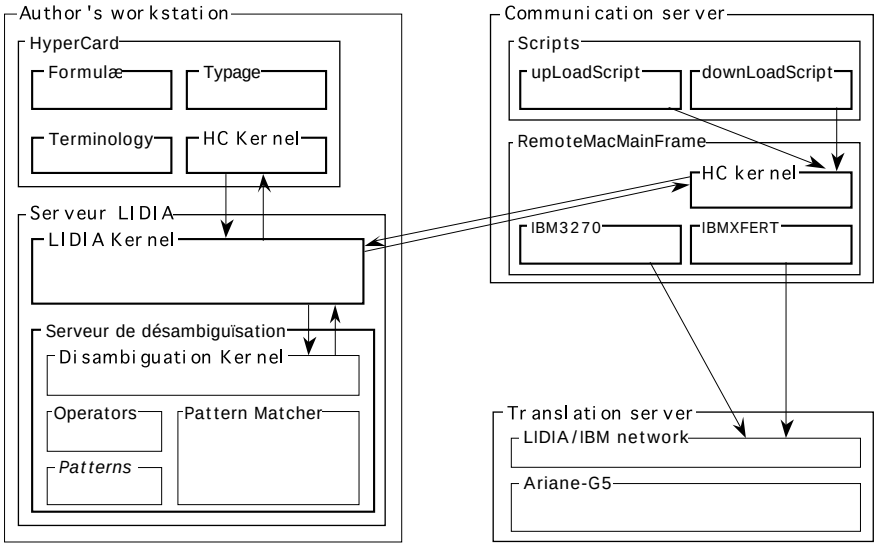

Figure 18: architecture of LIDIA-1

Three machines (fig. 18) are involved in the translation process.

On the author's workstation the HyperCard Kernel sends and receives messages from the LIDIA kernel which organises the translation process for each object. The LIDIA Kernel sends translation jobs to the Translation server via the Communication server. The LIDIA Kernel also asks to prepare the disambiguation questions.

2.2. Whiteboard approach

For each object to be translated, the LIDIA Kernel creates a *mirror object* (a text file) which stores all information required by the translation process and necessary for the construction of the target stack. We distinguish between *static* and *dynamic information*. Static information is what is attached by HyperCard to each object. It is necessary to construct target stacks. Dynamic information is any information used by LIDIA to translate the content of an object.

These files can be considered to constitute a *whiteboard* as defined in [32]. Unlike the blackboard, the whiteboard is accessed only by a *coordinator* (the LIDIA Kernel), and not by the components

(Disambiguation kernel and Ariane-G5). The main advantage of this architecture is to allow easy integration of existing or new components without having to modify them.

2.3. Object oriented techniques

All components but the lingware use object-oriented programming. The module for the Terminology, the idioms and the Typage as the kernel of the Communication server are written in HyperTalk, the HyperCard scripting language.

The LIDIA serveur is written in CLOS (MCL). Although encapsulated within the same environment, the LIDIA Kernel and the Disambiguation Kernel communicate by exchanging messages and might run on another machine.

Our use of messages and object-oriented programming techniques is close to the actor model used in the context of distributed cooperative systems.

III.3. Disambiguation dialogues

3.1. Types of ambiguities considered in LIDIA-1

In this mockup, we did not aim at handling all possible kinds of ambiguities, but a representative subset. In particular, pragmatic ambiguities are not considered.

It has been clear from the beginning that we would not be able to find, for each class of ambiguity, a unique resolution method. Keeping in mind the kind of dialogues we wanted, we have examined a large quantity of ambiguity configurations and have arrived at 8 problem patterns:

a.1. ambiguity of syntactic class:

1. ambiguity of syntactic class without ambiguous coordinated groups

Le pilote ferme la porte:	*The firm pilot carries her.* *The pilot shuts the door.*

2. ambiguity of syntagmatic class associated to a coordinated group

Il regarde la photo et la classe:	*He looks at the photograph and the class.* *He looks at the photograph and files it.*

a.2. ambiguity of geometry:

1. ambiguity of argument structure of the verb

Il parle de l'école de cuisine:	*He talks about the cooking school.* *He talks from the cooking school.* *He talks from the school about cooking.*

2. ambiguity of coordination

Il prend des crayons et des cahiers noirs:	*He takes pencils and black notebooks.* *He takes black pencils and black notebooks.*

3. ambiguity of subordination

L'école de cuisine lyonnaise est fermée:	*The lyonnaise cooking school is closed.* *The school of lyonnaise cooking is closed.*

a.3. ambiguity of syntactic function and/or logico-semantic relation

1. ambiguity of logico-semantic relation

Pierre fait porter des chocolats à Lucie:	*Pierre is having chocolates sent to Lucie.* *Pierre is having chocolates sent with Lucie.*

2. ambiguity of argument order for a direct transitive verb

Quel auteur cite ce conférencier:	*Which author is this lecturer quoting?* *Which lecturer is this author quoting?*

3. ambiguity of syntactic function

Il parle de la tour Eiffel:	*He is talking about the Eiffel Tower.* *He is talking from the Eiffel Tower.*

3.2. Strategy of disambiguation

a. Principle

A suggestion of [17] is to delay all interactions until transfer. We prefer to solve as soon as possible all the ambiguities which cannot be solved automatically later, or which can be solved later only with much difficulty.

Recall that *lexical ambiguities* concern not only the classical polysemy of terms (e.g., “diplôme” translates as “diploma” or ”degree”), but also *lexical ellipses*[26]. In both cases, our strategy is to generate a menu of the possible choices, ranked by their current weights[27].

The other ambiguities present in the mmc-structure are partitioned in three classes:

- There is an *ambiguity of syntactic class* if two or more syntactic classes are assigned to one occurrence of the source text in the solutions produced by the analyzer.
- There is an *ambiguity of geometry* if two solution trees have different graphs, with no difference in the syntactic classes assigned to the words.
- There is an *ambiguity of syntactic function and/or logico-semantic relation* if the analyzer produces two solutions having the same graph and the same syntactic classes. This means that some non-terminal nodes have a different labelling.

b. Strategy

If several problems appear in the same sentence, we use the following strategy:

1. determine the correct segmentation into simple phrases.
2. determine the correct subject, objects and adjuncts for each common predicate.
3. determine the correct links between simple phrases.

It seems to be a quite natural order to construct the sense of complex sentences.

For the system, this strategy entails solving the ambiguities of syntactic class first, then those of geometry, then those of functions and/or relations.

c. Method

The disambiguation process is organized around a pattern matcher. For five out of the eight classes of ambiguity considered in the mock-up, we have defined several sets of patterns. All patterns in a given set of patterns shares the same set of variables.

An ambiguity is recognized when a set of patterns matches an mmc-structure. It is true when each pattern of the set matches one or more trees of the multiple analysis and when the shared variables of the patterns have the same value in each match.

A paraphrase construction method is associated with each pattern to produce a selection dialogue. The methods rely on a set of 13 operators.

Here are some examples.

Agree — produces an inflected form starting from a lexical unit.

Distribute — distributes an occurrence or a group of occurrences on other groups of occurrences with a fixed link between the results of local distributions.

```
ex: Distribute(A, B C, D, 1, 2, or, 1, 3) -> A B C or A D
```

Substitute — substitutes an occurrence by another one according to conditions. The result is found in the lexical data base.

---

[26] Suppose a text is about a space ship containing a “centrale électrique” (“electric plant”) and a “centrale inertielle” (“inertial guidance system”). The complete form is often replaced by the elided one (“centrale”). Although it is crucial to disambiguate for translating correctly (by the corresponding elided forms : “plant”/”system”), no automatic solution is known. A given occurrence may be an elision or not. If so, it is even more difficult to look for a candidate to the complete form in a hypertext than in a usual text.

[27] Weights and lexical ellipses are not yet implemented in LIDIA-1.0.

Project — for a tree node, projects the syntactic class, and according to that class, some information that can be found in the lexical data base.

Text — produces the text associated with the parameters (parts of the mmc-structure)

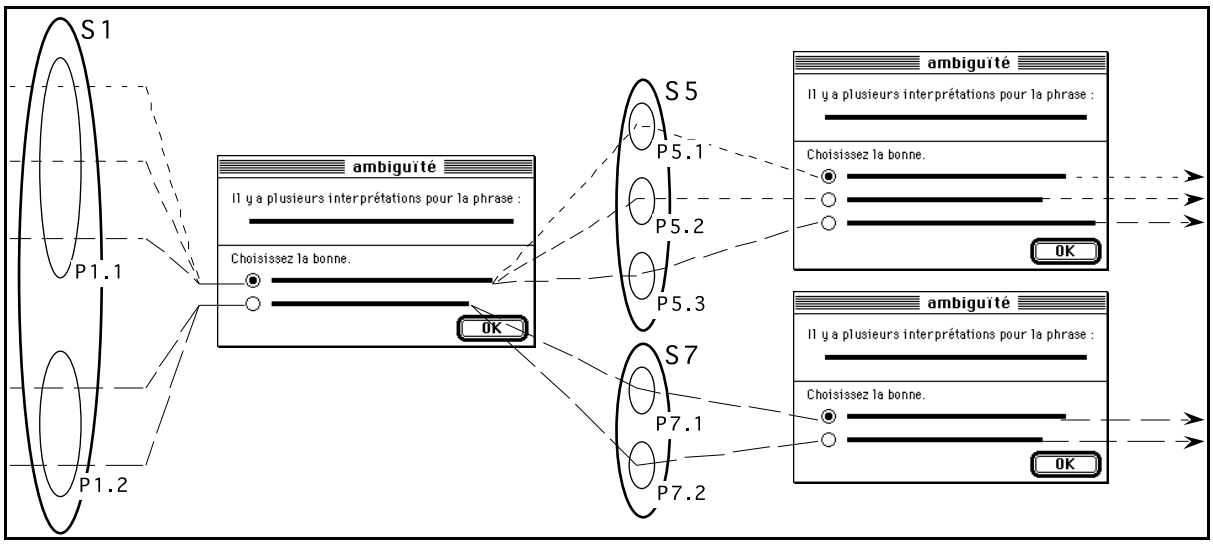

Figure 18: set of pattern matches and production of a question tree

3.3. An example

a. mmc-structure produced by the analyzer

Figure 19 shows the trees produced for the sentence ‘Le capitaine a rapporté un vase de Chine.’

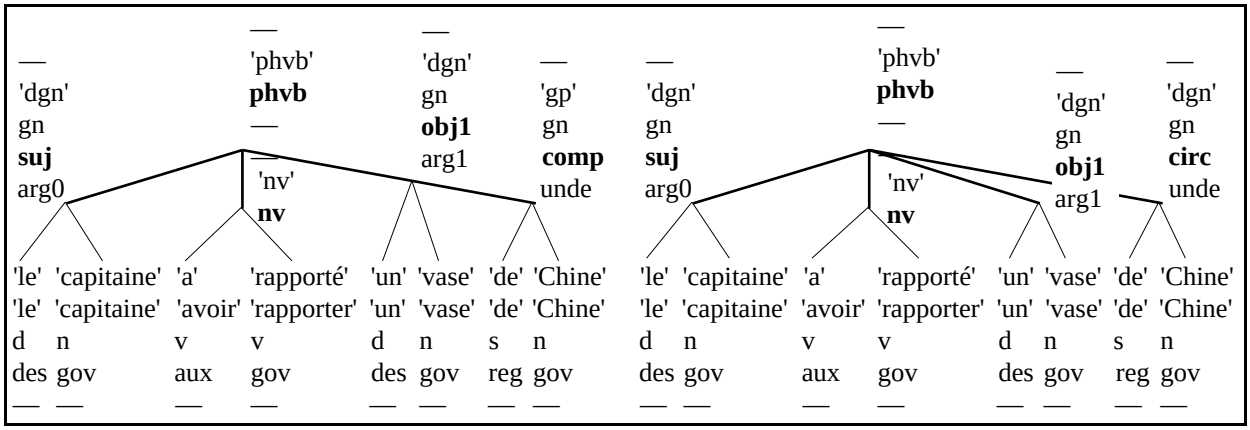

Figure 19: mmc-structure produced

b. Set of patterns marching the mmc-structure

The patterns (`Patron 12` & `Patron 13`) used to recognize the ambiguity in the previous sentence are shown in figure 20.

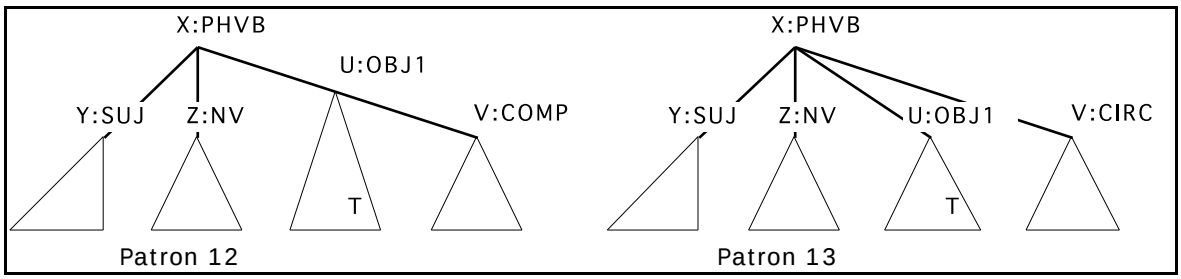

Figure 20: set of patterns used to recognize the ambiguity

c. Dialogue items produced

The method associated with pattern 12 is:

- Text(Y) Text(Z) Bracket(Text(T), Text(V))

which produces the following dialogue item:

- Le capitaine a rapporté (un vase de Chine.)

The method associated with pattern 13 is:

- Text(V), Text(Y) Text(Z) Text(T)

which produces the following dialogue item:

- de Chine, le capitaine a rapporté un vase.

Interactive disambiguation in some related work

Interactive MT was first proposed in the sixties by M. Kay for the MIND system [21], and several projects experimented with variations of this design, notably the ITS project [25] at Provo (BYU-TSI) from 1975 to 1981, the Alvey N-tran project [17] at Manchester (UMIST-CCL) from 1985 to 1987, and the DLT project [31] at Utrecht (BSO) from 1982 to 1988. In KBMT-89 [14] at CMU-CCL, questions were also asked by the "augmentor" if ambiguities could not be solved by the ontology. Ongoing work on 'MT for the target language inexpert' [19] has been described at COLING-90. JETS, IBM-Japan's Japanese-English MT system, is based on an interactive Japanese dependency parser [24]. We can also mention the ITS project [43] at the university of Geneva.

We are of course indebted to many of these projects, which have explored several disambiguation contexts and strategies. However, if our information is correct, there are still no practical systems based on these projects. In some cases, we think potential users are confused by the fact that the disambiguator often produces questions in a somewhat unpredictable and esoteric way. This is why we try to build a system which has a simple structure, and asks questions understandable by college graduates.

In some other cases, we feel that the situation chosen is not appropriate for DBMT. For instance, JETS is currently tested on writers of technical documentation, who are not normally responsible for translation, and are accustomed to simply give their texts to professional translators. As a result, they are quite reluctant to use the system, although it is quite good and would certainly be very successful with employees of small firms having to produce similar material in Japanese and English and willing to help the machine translate on the spot rather than to pay and wait for professional translation.

Perspectives

The concept of DBMT crystallizes many ideas from previous systems and research (text-critiquing, interactive MT, LBMT with preedition, KBMT with augmentation, controlled languages, sublanguages…). However, the constraint of interacting with an author having no knowledge of the target language(s), linguistics, or translation, puts things in an original framework.

Promotion of the National Languages is becoming quite important nowadays, but, apart from efforts to teach a few foreign languages, no technical solution has yet been proposed to help people write in their own language and communicate with other people in their own languages. We hope DBMT will be one of the tools used extensively in the future for cross-linguistic communication.

While the development of systems of this nature poses old problems in a new way, and offers interesting new possibilities to the developers, their acceptability and usefulness will perhaps result more from their ergonomy than from their intrinsic linguistic quality, however necessary that may be. From that point of view, it will be very important to research multimedia disambiguation techniques in the future. We have recently started a research on this topic in the framework of the cooperation between ATR and CNRS.

Acknowledgements

This research is being supported by University Joseph Fourier and CNRS. The contributors to LIDIA-1.0 are J.-Ph. Guilbaud, N. Nédobejkine, E. Blanc, M. Axtmeyer and D. Levenbach for the lingware; P. Guillaume, M. Quézel-Ambrunaz, G. Sérasset and M. Lafourcade for the software. Thanks also to M. Seligman and to our two anonymous referees, who reviewed our first draft in detail and made pertinent suggestions, which have led us,we hope, to improve the presentation.

-o-o-o-o-o-o-o-o-o-o-

References

[1] **Blanchon H. (1990)** *LIDIA-1 : Un prototype de TAO personnelle pour rédacteur unilingue.* Proc. X-èmes Journées sur les systèmes experts et leurs applications. Conférence spécialisée “Le traitement automatique des langues naturelles et ses applications”, Avignon, 28 mai-1 juin 1990, EC2, 51-60.

[2] **Blanchon H. (1992)** *A Solution to the Problem of Interactive Disambiguation.* Proc. COLING-92, Nantes, 23-28 July 1992, C. Boitet, ed., vol. 4/4, 1233-1238.

[3] **Blanchon H., Guilbaud J. P. & Nédobejkine N. (1992)** *LIDIA : the disambiguation process — le processus de désambiguïsation.* Exposition COLING-92, Nantes, 23-28 juillet 1992, Pile HyperCard.

[4] **Boitet C. (1988)** *Hybrid Pivots using m-structures for multilingual Transfer-Based MT Systems.* Jap. Inst. of Electr., Inf. & Comm. Eng., June 1988, NLC88-3, 17—22.

[5] **Boitet C. (1988)** *PROs and CONs of the pivot and transfer approaches in multilingual Machine Translation.* Proc. Int. Conf. on “New directions in Machine Translation”, Budapest, 18–19 August 1988, BSO, 13 p.

[6] **Boitet C. (1989)** *Motivation and Architecture of the LIDIA Project.* Proc. MTS-II (MT Summit), Munich, 16-18 août 1989, 12 p.

[7] **Boitet C. (1989)** *Speech Synthesis and Dialogue Based Machine Translation.* Proc. ATR Symp. on Basic Research for Telephone Interpretation, Kyoto, December 1989, 12 p.

[8] **Boitet C. (1990)** *Software and lingware engineering in recent (1980—90) classical MT : Ariane-G5 and BV/aero/F-E.* Proc. ROCLing-III (tutorials), Taipeh, Aug. 1990, 21 p.

[9] **Boitet C. (1990)** *Towards Personal MT : on some aspects of the LIDIA project.* Proc. COLING-90, Helsinki, 20-25 août 1990, H. Karlgren, ed., ACL, vol. 3/3, 30-35.

[10] **Boitet C. (1993)** *La TAO comme technologie scientifique : le cas de la TA fondée sur le dialogue.* In “Études et Recherches en Traductique”, A. Clas & P. Bouillon, ed., Presses de l’Université de Montréal, Montréal, 32 p.

[11] **Boitet C. & Blanchon H. (1993)** *Dialogue-Based MT for Monolingual Authors and the LIDIA project.* Proc. NLPRS’93 (Natural Language Processing Rim Symposium, Fukuoka, 6-7/12/93, H. Nomura, ed., Kyushu Institute of Technology, 208—222.

[12] **Boitet C. & Zaharin Y. (1988)** *Representation trees and string-tree correspondences.* Proc. COLING-88, Budapest, 22–27 Aug. 1988, ACL, 59—64.

[13] **Brown R. D. (1989)** *Augmentation.* Machine Translation, 4, 1299-1347.

[14] **Brown R. D. & Nirenburg S. (1990)** *Human-Computer Interaction for Semantic Disambiguation.* Proc. COLING-90, Helsinki, 20-25 août 1990, H. Karlgren, ed., ACL, vol. 3/3, 42-47.

[15] **Carbonnell J. G. & Tomita M. (1985)** *New Approaches to Machine Translation.* Proc. TMI-85 (Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages), Hamilton, N.Y., 14-16 Aug. 1985, S. Nirenburg, ed., 59-74.

[16] **Chandioux J. (1988)** *10 ans de METEO (MD).* In “Traduction Assistée par Ordinateur. Actes du séminaire international sur la TAO et dossiers complémentaires”, A. Abbou, ed., Observatoire des Industries de la Langue (OFIL), Paris, mars 1988, 169—173.

[17] **Chandler B., Holden N., Horsfall H., Pollard E. & McGee Wood M. (1987)** *N-tran Final Report.* Alvey Project, 87/9, CCL/UMIST, Manchester.

[18] **Heidorn G. E., Jensen K. & Miller L. A. (1982)** *The EPISTLE Text-Critiquing System.* IBM System Journal, **21**/1, 305-326.

[19] **Huang X. M. (1990)** *A Machine Translation System for the Target Language Inexpert.* Proc. COLING-90, Helsinki, 20-25 Aug. 1990, H. Karlgren, ed., ACL, vol. 3/3, 364-367.

[20] **JEIDA (1989)** *A Japanese view of Machine Translation in light of the considerations and recommendations reported by ALPAC, USA.* Japanese Electronic Industry Development Association, Tokyo, 197 p.

[21] **Kay M. (1973)** *The MIND system.* In “Courant Computer Science Symposium 8: Natural Language Processing”, R. Rustin, ed., Algorithmics Press, Inc., New York, 155-188.

[22] **Kittredge R. (1983)** *Sublanguage — Specific Computer Aids to Translation — a survey of the most promising application areas.* Contract n° 2-5273, Université de Montréal et Bureau des Traductions, mars 1983, 95 p.

[23] **Kittredge R. (1986)** *Analyzing Language in Restricted Domains.* In “Sublanguage Description and Processing”, R. Grishman & R. Kittredge, ed., Lawrence Erlbaum, Hillsdale, New-Jersey.

[24] **Maruyama H., Watanabe H. & Ogino S. (1990)** *An Interactive Japanese Parser for Machine Translation.* Proc. COLING-90, Helsinki, 20-25 août 1990, H. Karlgren, ed., ACL, vol. 2/3, 257-262.

[25] **Melby A. K. (1981)** *Translators and Machines - Can they cooperate ?* META, **26**/1, 23-34.

[26] **Melby A. K. (1982)** *Multi-Level Translation Aids in a Distributed System.* Proc. COLING-82, Prague, 5-10 juillet 1982, vol. 1/2, 215-220.

[27] **Melby A. K., Smith M. R. & Perterson J. (1980)** *ITS : An Interactive Translation System.* Proc. COLING-80, Tokyo, 30 septembre-4 octobre 1980, M. Nagao, ed., 424-429.

[28] **Nirenburg S. (1989)** *Knowledge-based Machine Translation.* Machine Translation, 4, 5-24.

[29] **Nyberg E. H. & Mitamura T. (1992)** *The KANT system: Fast, Accurate, High-Quality Translation in Practical Domains.* Proc. COLING-92, Nantes, 23-28 July 92, C. Boitet, ed., ACL, vol. 3/4, 1069—1073.

[30] **Richardson S. D. & Braden-Harder L. C. (1988)** *The experience of developing a large-scale natural language text processing system: CRITIQUE.* Proc. 2nd conference on Applied Natural Language Processing, 1988, 195-202.

[31] **Sadler V. (1989)** *Working with analogical semantics : Disambiguation technics in DLT.* T. Witkam, ed., Distributed Language Translation (BSO/Research), Floris Publications, Dordrecht, Holland, 256 p.

[32] **Seligman M. & Boitet C. (1993)** *A "Whiteboard" Architecture for Automatic Speech Translation.* Proc. International Symposium on Spoken Dialogue, Tokyo, 10–12 November 1993, Waseda University, 4 p.

[33] **Sérasset G. & Blanc É. (1993)** *Une approche par acceptions pour les bases lexicales multilingues.* Proc. IIIèmes journées scientifiques "Traductique-TA-TAO", Montréal, octobre 1993, Presses de l'Université de Montréal, 17 p.

[34] **Somers H. L., Tsujii J.-I. & Jones D. (1990)** *Machine Translation without a source text.* Proc. COLING-90, Helsinki, 20-25 Aug. 1990, H. Karlgren, ed., ACL, vol. 3/3, 271-276.

[35] **Tomita M. (1984)** *Disambiguating Grammatically Ambiguous Sentences by Asking.* Proc. COLING-84, Stanford, 2-6 juillet 1984, ACL, 476-480.

[36] **Tomita M. (1986)** *Sentence Disambiguation by asking.* Computers and Translation, **1**/1, 39-51.

[37] **Van Herwijnen E. (1990)** *Practical SGML.* Kluwer, Dordrecht, 307 p.

[38] **Vauquois B. (1988)** *BERNARD VAUQUOIS et la TAO, vingt-cinq ans de Traduction Automatique, ANALECTES. BERNARD VAUQUOIS and MT, twenty-five years of MT.* C. Boitet, ed., Ass. Champollion & GETA, Grenoble, 700 p.

[39] **Vauquois B. & Boitet C. (1985)** *Automated translation at Grenoble University.* Comp. Ling., **11**/1, January-March 85, 28—36.

[40] **Vauquois B. & Chappuy S. (1985)** *Static grammars: a formalism for the description of linguistic models.* Proc. TMI-85 (Conf. on theoretical and metholodogical issues in the Machine Translation of natural languages), Colgate Univ., Hamilton, N.Y., Aug. 1985, S. Nirenburg, ed., 298-322.

[41] **Veronis J. & Ide N. (1990)** *Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine-Readable Dictionaries.* Proc. COLING-90, Helsinki, 18—25 Aug. 1990, H. Karlgren, ed., 389—394.

[42] **Weaver A. (1988)** *Two Aspects of Interactive Machine Translation.* In "Technology as Translation Strategy", M. Vasconcellos, ed., State University of New York at Binghamton, Binghamton, 116-123.

[43] **Wehrli E. (1990)** *STS: An Experimental Sentence Translation System.* Proc. COLING-90, Helsinki, 20-25 Aug. 1990, H. Karlgren, ed., ACL, vol. 1/3, 76-78.

[44] **Wehrli E. (1991)** *Pour une approche interactive au problème de la traduction automatique.* Proc. Colloque "L'environnement traductionnel. La station de travail du traducteur de l'an 2001", Mons, 25-27 avril 1991, AUPELF & UREF, 59-68.

[45] **Wehrli E. (1992)** *The IPS System.* Proc. COLING-92, Nantes, 23-28 July 1992, C. Boitet, ed., vol. 3/4, 870-874.

[46] **Whitelock P. J., Wood M. M., Chandler B. J., Holden N. & Horsfall H. J. (1986)** *Strategies for Interactive Machine translation : the experience and implications of the UMIST Japanese project.* Proc. COLING-86, Bonn, 25-29 Aug. 1986, IKS, 25-29.

[47] **Wood M. M. G. & Chandler B. (1988)** *Machine Translation For Monolinguals.* Proc. COLING-88, Budapest, 22-27 Aug. 1988, 760—763.

[48] **Zaharin Y. (1986)** *Strategies and heuristics in the analysis of a natural language in Machine Translation.* Ph.D. thesis, Universiti Sains Malaysia, Penang.

[49] **Zajac R. (1988)** *Interactive Translation : a new approach.* Proc. COLING-88, Budapest, Aug. 1988.

-o-o-o-o-o-o-o-o-o-o-o-

Table of contents

-o-o-o-o-o-o-o-o-o-o-