TR-IT-0072

On the design of MIDDIM-DB, a data base of ambiguities and disambiguation methods

Christian BOITET

1994.9.10

Interactive disambiguation will be necessary in future Machine Translation and Machine Interpretation systems to be used in non-restricted contexts by the general public. In order to determine which ambiguities end users could and should help the system disambiguate, and which (possibly multimodal) disambiguation methods are most appropriate, we have started to build MIDDIM-DB, a data base of real ambiguities and simulated or implemented disambiguation methods.

A first version has already been prototyped in HyperCard, and has shown the need for some design improvements. This paper elaborates on the motivations to build such a data base, and presents the main aspects of the design of a second version. An important byproduct of this enterprise is an exact and formal definition of what an ambiguity is, so that an ambiguity becomes a computational object.

Interpreting Telecommunications Research Laboratories
2-2 Hikari-dai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

On the design of MIDDIM-DB, a data base of ambiguities and interactive disambiguation methods

Christian Boitet*
GETA, IMAG (UJF & CNRS)
BP 53, 38041 Grenoble Cedex 9, France
Christian.Boitet@imag.fr

Abstract

Interactive disambiguation will be necessary in future Machine Translation and Machine Interpretation systems to be used in non-restricted contexts by the general public. In order to determine which ambiguities end users could and should help the system disambiguate, and which (possibly multimodal) disambiguation methods are most appropriate, we have started to build MIDDIM-DB, a data base of real ambiguities and simulated or implemented disambiguation methods.

A first version has already been prototyped in HyperCard, and has shown the need for some design improvements. This paper elaborates on the motivations to build such a data base, and presents the main aspects of the design of a second version. An important byproduct of this enterprise is an exact and formal definition of what an ambiguity is, so that an ambiguity becomes a computational object.

Introduction: context of the project and envisaged situations

The MIDDIM project (Multimodal Interactive Disambiguation / Désambiguïsation Interactive Multimodale) is a 3-year cooperative research project between ATR and CNRS† that started in July 1993. Its aims are to study multimodal interactive disambiguation methods, and more generally interactive disambiguation methods, within the contexts of personal machine translation of written documents by monolingual authors and of speech translation of dialogues between monolingual locutors (interpreting telecommunications).

It must be stressed that interactive disambiguation is not to be used to solve all ambiguities. On the contrary, as many ambiguities as possible should be reduced automatically. The remaining ones should be solved by interaction as far as practically possible. What is left would have to be reduced automatically again, by using preferences and defaults.

1. Written documents

In the intended situations, a (normally monolingual) author creates a document and helps the system translate it into one or more target languages, by guiding the automatic process through the insertion of optional marks and by selecting among competing analysis results in the course of a clarification dialogue. This Dialogue-Based approach (DBMT) is currently being pursued by several research groups, including IBM-Japan (JETS system [1]) and GETA-IMAG (LIDIA project [11]).

In the case of LIDIA, the emphasis is to go from a mockup to a usable prototype. The documents to translate are of two main types:

- purely textual ("unimodal") and relatively simple documents, such as *abstracts*;
- multimodal and textually very fragmented documents, such as *slides and HyperCard stacks* (both considered as hypertexts containing text plus graphics, and possibly images, sound, video, etc.).

2. Spoken dialogues

In the intended situations, two humans ignorant of each other's language (try to) communicate by using a MI (Machine Interpretation) system. Due to the limitations in the current state of the art, a human interpreter, an expert of the system, should be on hand to help when communication through the MI system becomes too difficult. We elaborate this point in II.2 below.

* Visiting researcher at ATR Interpreting Telecommunication Laboratories

† Centre National de la Recherche Scientifique, Paris, France.

Only two languages are tackled at a time, but there are heavy delay constraints which will usually make it impossible to solve all ambiguities interactively. Given the fact that two presumably intelligent humans are conversing, it seems actually better to ask them as few questions as possible, and to leave them free to choose whether or not to request the other person to clarify an unclear point.

In contrast to DBMT, where available devices are quite standard, we must distinguish here between two kinds of spoken dialogues, according to the available communication devices.

- In *telephone dialogues*, speech is almost exclusively used. Apart of using speech, the users may send very short messages by using the small telephone keypad, and receive short written messages on the message window which is incorporated in an increasing number of telephone sets (international phones, hotel phones, etc.).
- In *multimodal dialogues*, the users may also share graphics such as maps or forms, and see their interlocutor's face and scene through video. Multimodal analysis systems might use clues from different modalities to automatically solve ambiguities unsolvable in unimodal contexts. For example, the antecedent of a pronoun may be the reference of the object or icon pointed at with the mouse.

To sum up, then, there are three envisaged contexts:

(1) Translation of texts or hypertexts.
(2) Interpretation of telephone-only dialogues.
(3) Interpretation of multimodal dialogues.

3. Previous work

A small collection of abstracts and slides, and the written transcriptions of two dialogues from [8], has been selected to be the "test corpus" for building MIDDIM-DB [9]. One dialogue is a telephone dialogue and the other a multimodal dialogue.

A first version, MIDDIM-DB 1.0, has been implemented in HyperCard by J. Winship for her Erasmus project at GETA. Its design is discussed in [12], and a complete documentation may be found in [15].

It was planned to integrate sound and video recordings in the data base at a later stage, and thus the current design does not yet take them into account. However, experiments conducted by G. Fafiotte have shown that audio and video tapes from ATR can easily be used to attach appropriate audio and video sequences to HyperCard cards.

The remainder of the report is organized as follows. First, we try to make the notion of "ambiguity" operational, so that it makes sense to speak of ambiguities as objects to be put in a data base. Then we demonstrate the necessity of studying interactive disambiguation in general, and multimodal interactive disambiguation in particular, if one seriously wants to produce practical systems of the envisaged kinds. As the usefulness of constructing a data base of ambiguities may not be obvious, we then list its potential usages for studies and experiments. Finally, we present the design of a second version, MIDDIM-DB.2, with reference to that of the first version.

I. Ambiguities as formal objects

1. When do we say there is an ambiguity?

In the usual sense, there is an ambiguity in an utterance if there are at least two ways of understanding it. The utterance may be spoken or written, may be a sentence, a phrase, a sequence of words, syllables, etc. This, however, does not give us a precise criterion. Because human understanding heavily depends on the context and the communicative situation, it is indeed a very common experience that something is ambiguous for one person and not for another.

We will explicitly exclude the meaning of "ambiguous" illustrated by the sentence "river is ambiguous with stream". What this sentence says is that the same concept may be rendered by "river" or "stream". But that is homonymy and not ambiguity.

Note that, in the common understanding, ambiguities are justly felt to be of different kinds. Maybe what I have heard was ambiguous between "pen" and "pin", and if it was "pen", I don't know whether it is a writing tool or an enclosure for animals, etc. These distinctions should be captured in any formalization.

Another observation is that ambiguities "manifested" on fragments of utterances are usually not all the potential ambiguities shown by the fragments taken in isolation.

2. Any operational notion of ambiguity must be based on a "computable" representation system

As human understanding cannot be examined, we replace "different understanding" by "different representations". Then, we can say that there is a phonetic ambiguity in an utterance if it has at least two different phonetic representations, and so forth for all the levels of representation, from phonetic to orthographic, morphological, morphosyntactic, syntagmatic, functional, logical, semantic, and pragmatic.

Classical representation systems are based on lists of binary features, flat or complex attribute structures (property lists), labeled or decorated trees, various types of feature-structures, graphs or networks, and logical formulae.

Now, we are still left with two questions:

1) which representation system do we choose?
2) how do we determine the representation or representations of a particular utterance in a specific representation system?

The answer to the first question is a practical one. The representation system must be fine-grained enough to allow the intended operations. For instance, text-to-speech requires less detail than translation. On the other hand, it is counter-productive to make too many distinctions. For example, what is the use of defining a system of 1000 semantic features if no system and no lexicographers may assign them to terms in an efficient and reliable way? There is also a matter of taste and consensus. Although different representation systems may be formally equivalent, researchers and developers have their preferences. Finally, we should prefer representations amenable to efficient computer processing.

As far as the second question is concerned, two aspects should be distinguished. First, the consensus on a representation system goes with a consensus on its semantics. This means that people using a particular representation system should develop guidelines enabling them to decide which representations an utterance should have, at each level, and to create them by hand if challenged to do so. Second, these guidelines should be refined to the point where they may be used to specify and implement a parser producing all and only the intended representations for any utterance in the intended domain of discourse.

Definitions

A "computable" representation system is a representation system for which a "reasonable" parser can be developed.

A "reasonable" parser is a parser such as:

- its size and time complexity are tractable over the class of intended utterances;
- if it is not yet completed, assumptions about its ultimate capabilities, especially about its disambiguation capabilities, are realistic given the state of the art.

Suppose, then, that we have defined a computable representation. We may not have the resources to build an adequate parser for it, or the one we have built may not yet be adequate. *In that case, given the fact that we are specifying what the parser should and could produce, we may anticipate and say that an utterance presents an ambiguity of such and such types.* This only means that we expect that an adequate parser will produce at least two representations for the utterance at the considered level.

3. Definition of ambiguities as formal objects

Up to now, we have simply said that "an utterance" is ambiguous. In reality, we always narrow down to some "ambiguous part" in the utterance. This is what we want to formalize now.

Let us first take an example. Consider the utterance:

(1) `Do you know where the` <u>`international`</u> <u>`telephone`</u> <u>`services`</u> `are located?`

We would like to say that the underlined fragment has an ambiguity of attachment, because it has two different "skeleton" [6] representations:

```
[international telephone] services / international [telephone services]
```

However, it is not enough to consider this sequence in isolation. Take for example:

```
(2)  The international telephone services many countries.
```

The ambiguity has disappeared! It is indeed frequent that an ambiguity relative to a fragment appears, disappears and reappears as one broadens its context in an utterance. Hence, in order to define properly what an ambiguity is, we must consider the fragment *within an utterance*, and clarify the idea that the fragment is the smallest (within the utterance) where the ambiguity can be observed.

Informally, then, a fragment F presents an ambiguity of degree n ($n \geq 2$) in an utterance U if it has n different representations which can be extended in the same way to give a complete representation of U. To qualify as support of the ambiguity, F should further be minimal relative to that ambiguity, which means that F and its associated n representations cannot be reduced to a strictly smaller fragment F' and the n associated sub-representations without loosing the first property.

In example (1) above, then, the fragment "the international telephone services", together with the two skeleton representations

```
the [international telephone] services / the international [telephone services]
```

is not minimal, because it and its two representations can be reduced to the subfragment "international telephone services" and its two representations (which are minimal).

Definition

An ambiguity A of degree n ($n \geq 2$) relative to a representation system R, may be formally defined as:

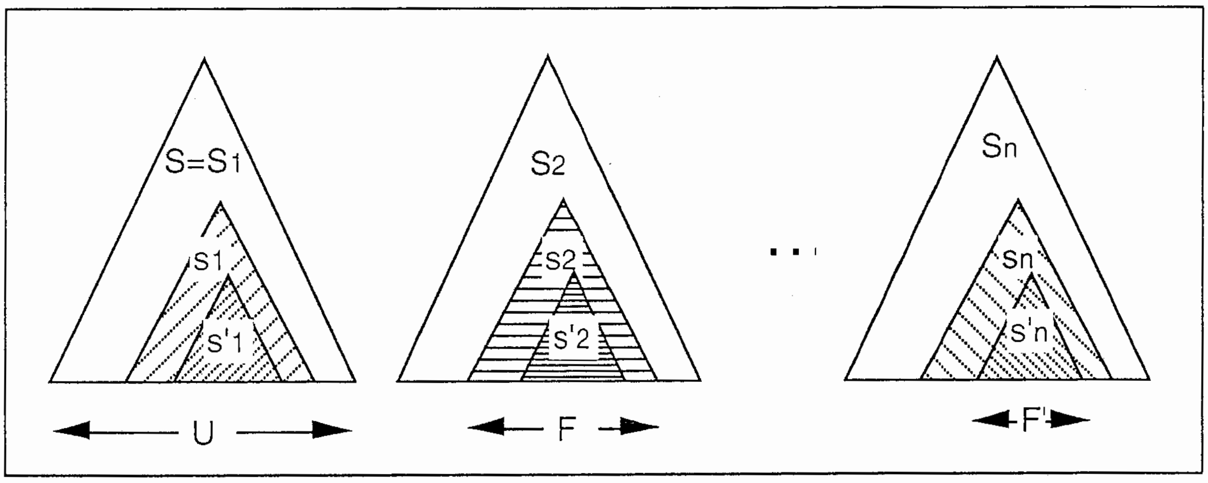
$$A = (U, F, S, <s_1, s_2 \ldots s_n>), \qquad \text{where:}$$

- U is a complete utterance, called the *context* of the ambiguity.
- F is a fragment of U, usually, but not necessarily connex, the *support* of the ambiguity.
- S is a representation of U in R, and s_1 is the part of S representing F.
- Each representation S_i obtained by substituting s_1 by s_i ($2 \leq i \leq n$) in S is a representation of U.
- Minimality condition:

  Let F' be any fragment of U strictly contained in F, and $s'_1, s'_2 \ldots s'_n$ be the parts of $s_1, s_2 \ldots s_n$ corresponding to F'. Then, for at least one i ($2 \leq i \leq n$), the result of substituting s'_1 by s'_i in S is different from S_i.

The *type* of A is the way in which the s_i differ, and must be defined relative to each particular R.

This may be illustrated by the following diagram, where we take the representations to be tree structures represented by triangles.



The white parts $S\backslash s_1, \ldots S\backslash s_n$ must all be equal, while not all $S\backslash s'_1, \ldots S\backslash s'_n$ are equal, or, equivalently, the differences $s_1\backslash s'_1, \ldots s_n\backslash s'_n$ can not all be equal.

Note that the same fragment F can participate in several ambiguities, if for example s_1 is a part of another structure S' for U.

II. What is interactive disambiguation and do we really need multimodal interactive disambiguation?

1. Definition of interactive disambiguation

Interactive disambiguation is not just any sort of help given by users to an MT or MI system, such as speaking clearly, using simple words, avoiding pronouns, and the like.

It is an active participation in the automated process:

- Input oriented interactive disambiguation, or GUIDANCE, consists in:
  - *inserting disambiguating marks* known to help the system, such as "stop" to separate two spoken utterances, or morphosyntactic tags, such as "service_N", or generic task names, such as "hotel-booking", "path-finding", or "appointment-setting" (used to narrow down the lexicon).

    In telephone dialogues, there would be very few such marks, because they would have to be spoken. In the other situations, they could be easily inserted by clicking the mouse or touching the screen with a finger.
  - *interrupting to correct the utterance being processed,* because it is realized that it was erroneous or too difficult for the system.
- Output oriented interactive disambiguation, or CLARIFICATION, consists in:
  - *answering clarification questions* presented by the system. These questions are all about ambiguities not solvable by the system.
  - *selecting among multiple intermediate results* while the system is processing the utterance, *or correcting* some of them. For example, a speech recognizer could show multiple candidates for words or phrases. Some of the ambiguities might be solved later by the system anyway, and some might not. Hence, this kind of interaction is best viewed as help to speed-up the whole process.

    This supposes some kind of feed-back, which will be available in DBMT and multimodal dialogues, but not in telephone dialogues, or only in a very limited way.

2. Interactive disambiguation will be unavoidable in the envisaged contexts

Automatic disambiguation of texts, even using extensive statistical tuning and relatively shallow representations, reaches only about 75% "Viterbi consistency" in the best experiments done so far with large corpuses [6]. This means that the solution which is output as having the best score according to the parser is wrong 25% of the time.

In contrast, the "structural consistency" can reach 95% to 96%. This means that the correct solution is almost always present among the solutions produced. However, the number of solutions produced for a string w is of the form $k^{|w|}$, where $k \approx 1.35$ and $|w|$ is the number of words in w. This gives about 8 parses for an utterance of 7 words, 34 for one of 12 words, 144 for one of 17 words, 576 for one of 24 words, and so forth[1]. Hence, it is possible to build an interactive disambiguation process based on a binary decision tree of height less then n/2 (as $\log_2 1.35 \approx 0.432$), which means asking as many as 27 questions for a sentence such as this one, which contains 54 words and punctuations.

This shows that interactive disambiguation is necessary, if good performance is pursued in relatively open contexts, even if only written input is considered. At the same time, the number of questions needed to reach full disambiguation may be practically excessive. In that case, only the "most important"

[1] We stop at 24, but longer written sentences and spoken utterances are present in our test corpus [9]. For example, the second sentence in the abstract of this report has 43 words, not counting punctuation marks.

questions should be asked. One remaining analysis result should then be automatically selected on the basis of whatever formal criteria are available.

The situation is far worse in speech recognition. The above figures seem to be attainable only in very restricted situations. Researchers at ATR mentioned Viterbi consistency rates of 30% in open contexts. Even if "input oriented" interactive disambiguation makes it possible to narrow down the vocabulary so that a Viterbi consistency of 60% and a structural consistency of 85% are attained – these are very optimistic figures – the overall Viterbi consistency of the whole speech recognition and analysis process would not exceed 45%, and the overall structural consistency 72%[2].

Hence, users will have to help. However, selecting between various possibilities cannot lead to an overall performance better than the structural consistency. Users will have to provide directly some information to cover the structurally inconsistent cases, where there is no satisfactory solution to select. In DBMT, this could mean to give a few properties of an unknown word, or to transform a (skeleton) parse into one not produced by the analyzer. In multimodal dialogues, users could also type in some misrecognized words.

In spoken dialogue interpretation, however, it is not realistic to suppose that users might give enough new information to the system to cover 28% of the cases (from 72% to 100%). This is why K.H. Loken-Kim and I have proposed an architecture where a human interpreter, an expert on the system, may be called in to help when communication through the MI system stalls or deadlocks [7].

3. Why and how can interactive disambiguation be multimodal?

Interactive disambiguation may be performed using only text, only speech, only graphics, or any combination of the three. It is hypothesized that higher efficiency will be reached in multimedia contexts, where different techniques and communication channels can be combined.

3.1. Multimodal interactive disambiguation of written documents

We may consider the following possibilities, usable alone or in conjunction:

- *Textual interactive disambiguation* through menus (as in LIDIA-1.0 [13]).
- *Graphical interactive disambiguation* using colors, structures, icons, arrows pointing at possible referents, etc. For example, JETS [1] uses colors to highlight possible dependency word pairs. To solve attachment ambiguities, the user could also manipulate the "best guess" of the system, in something like the outline mode of Word™, either under the constraints given by the set of possible structures (selection only) or in free mode (selection or correction).
- *Spoken interactive disambiguation.* Some questions could be spoken by the system. If speech synthesis is good enough, this would open interesting possibilities [2], e.g. to disambiguate about emphasis. A limited speech recognition capability, such as item spotting[3], could be necessary to make spoken disambiguation dialogues natural and usable, because users would presumably expect to talk back and not to answer by typing, clicking or pointing.

3.2. Multimodal interactive disambiguation of telephone dialogues

Although the telephone seems to be unimodal, it always has a numeric keyboard, and sometime a 1-way text pad. The numeric keyboard is routinely used for automated opinion polls and question-answering surveys. It could typically be used to interrupt the system in order to correct one's previous utterance. In addition, it could be used to control the overall communication process, e.g. by indicating one's desire to abort the translation because the message is already clear, or for any other reason.

The text pad could be used by the system to send a very rough translation, obtained by word-for-word translation of the words spotted so far, so that the user may understand the message even before it is completely analyzed and disambiguated at the other end.

However, it is clear that the bulk of the interactive disambiguation should be done through speech.

---

[2] To take the product may be oversimplified, because the sources of knowledge of speech recognition and linguistic analysis may not be independent. But a recent paper by Oviatt & Cohen (ref. in cmp-lg archive) estimates the error in doing so at bout 10%, so that the figures could "climb" to 50% and 80% in the best possible situation. Our argument still holds.

[3] In commercial applications such as telephone ordering, an item is a term to be chosen from a short list known by the speech recognizer. This is a simpler task than, for example, that of spotting words in continuous speech.

3.3. Multimodal interactive disambiguation of multimodal dialogues

In addition to what has been said for the case of written documents, it is possible to envisage disambiguation techniques based on the use of two modalities at the same time, because a sophisticated speech recognition system will necessarily be available. For instance, it is natural for a user to draw a circle around a spot on a map while pronouncing the name of something lying within the circle, such as the name of a train station, a street, a square or a monument.

Also, the possibilities to offer guidance tools will increase enormously. For example, the speech recognizer could show its progress in a window, by displaying the words spotted so far in a graphic form (lattice or other), and the user could click to select the correct ones, or overwrite to correct the system and force his intended words.

III. A data-base of ambiguities, what for?

According to what has been said above, building an interactive disambiguation subsystem requires us to study in detail:

- the real ambiguities arising in the intended context.
- the methods used to solve them interactively.

1. Study of real ambiguities

That study should be qualitative, quantitative, and relative:

qualitative:	ambiguity should be organized by linguistic criteria, and appropriate examples should be provided, in the three envisaged contexts distinguished in the introduction.
quantitative:	the frequency of occurrence of each type of ambiguity should be researched, again in each envisaged contexts.
relative:	the relative importance of solving each type of ambiguity in each context should be researched, as some relatively rare ambiguities may be more damaging to the task at hand than more frequent ones.

That kind of study should be done by collecting information and examining it. The information should consist in real utterances, together with the analyses produced by one (or more) reasonable parser, or, if none is available, with analyses produced manually as approximations of what a reasonable parser should produce.

The qualitative and quantitative analysis should consist in finding all ambiguities, storing them as objects (see definition above) in the data base, classifying them, and counting them in various ways.

How to perform a relative analysis is not yet very clear. A first idea is simply to have a human expert translator or interpreter assign a "communicative penalty" to each ambiguity. The higher the penalty of an ambiguity, the more crucial it is to disambiguate it in order to avoid communication problems.

For example, ambiguities on number is very frequent in Japanese. The expert would presumably decide that many of them don't need to be solved for good communication to be achieved, and assign them low penalties. Finding out why some ambiguities are more crucial than others will help in designing disambiguation strategies which ask the most important questions first. This is important, because, as we have argued above, it will in many cases not be practical to ask all questions necessary to reach perfect disambiguation.

2. Simulation or experimentation of interactive disambiguation methods

The objective of the project is to study which combinations are adequate in the envisaged contexts, and how to implement them in practice.

Note that the database can only be used to study output-oriented (clarification) methods, because:

- it is not practical to store there the input (in case of speech) and the intermediate stages of recognition and analysis, and
- in any case, the effects of input-oriented methods could not be studied without using a completely implemented system.

3. Help in Wizard of Oz experiments

The study of interactive methods should lead us to propose experiments for the simulator. For example, one could write a simple program which would prompt the "wizard interpreter" to ask a disambiguating question of a certain type (or types), at a given moment, so as to reach a preset goal of frequency of questions and relative frequency of each type of question.

The program would simply keep a count of the number of clarification-oriented utterances and prompt the wizard interpreter for a given type of question, or tell him or her to stop asking questions for a while.

More could actually be done. For example, other wizards might help the wizard interpreter by preparing clarification questions. For this, they would spot an ambiguity, and fill an "ambiguity card" on the fly. This means that they would fill a field with the context of the ambiguity, select its support, and prepare the associated representations.

In the case of lexical ambiguity, clicking on a button would access a dictionary, which would return appropriate definitions or synonyms as representations.

In the case of a structural ambiguity, the helper wizard would quickly insert a few brackets, in the same way this is done by developers of tree banks [6].

In the case of a discourse ambiguity (illocutionary force type, speaker-hearer distinction, etc.), the helper wizard could checked the possible values in a panel, an so on.

IV. Design

A first version, MIDDIM-DB-1.0, has been developed by J. Winship in the framework of her ERASMUS stay at GETA from February to August 1994. This version, described in [12] has been installed and tried. Because of lack of time and underestimation of the difficulty of the task, it was not completely finished when we began to work on it at ATR. We have tried to round up its programming, and thereby studied the complete documentation given in [15]. This has led us to the decision to specify and implement a second version, which would:

- completely overhaul the part of the code and data structure aiming at making the base multilingual, in the sense that all messages and button names appear in one among several menu-selectable dialogue languages.
- normalize the programming techniques employed, by
  - reducing the number of global variables from several dozens to at most one or two;
  - transforming all procedures ("scripts") which act directly on other objects (cards, fields, buttons) so that they only send messages to these objects.
  - improving the interface so that it conforms to the Macintosh guidelines and produces the expected look-and-feel.
- take into account the possibility to integrate sound and video, by adding adequate fields and programming buttons to control them.

It should be stressed that, although the first version was not finished and leaves room for many improvements, it is the result of an enormous amount of work by a very dedicated student.

1. Why HyperCard?

We have chosen HyperCard because:

- it is able to support sound and video in addition to text;
- it has a powerful programming language, HyperTalk;
- it is multiscript, so that Japanese is supported without doing anything special;
- it is open, in the sense that it can call any other application;
- it is extensible, in the sense that "external commands" (XCMD) can be programmed in any available programming language, and added to the repertory of HyperTalk commands;
- it is widely available, very well documented, and cheap.

We have experimented with videotapes produced by EMMI and with audio tapes. It proved quite easy to attach audio and video sequences to HyperCard cards. However, when converting a video tape into a QuickTime file on a Macintosh AV using Fusion Recorder™, the sound was lost. This problem has yet to be fixed.

HyperCard manipulates "stacks" which consist of "cards" on which we can store information. The cards in a stack are linked by a common subject, or theme, and the stack forms a HyperCard document. A card has a named background, and several cards can share a background, thus allowing the easy storage of data in a standard format. A stack can contain cards of several backgrounds.

There are also buttons and fields. Buttons allow actions to be performed according to their scripts, and fields allow the storage of text (fields are actually also "scriptable"). Both can be either part of the background of the card (in which case they appear on all cards with the same background), or of the card itself (in which case they appear only on that particular card).

Scripts can be associated to buttons, fields, cards, backgrounds and stacks themselves. The scripting language, HyperTalk, functions on the principle of "message passing".

Messages can be sent to objects (buttons, fields, cards and stacks), and the message is interpreted by the script of the receiving object. There exists a hierarchy in which messages pass. If the message is not interpreted by the receiving object, it passes in the hierarchy until it is intercepted.

A message goes first of all to buttons and fields, then to cards, then to backgrounds, then to stacks in use, and finally to the Home stack before being passed to HyperCard itself if it's still not intercepted. The system sends messages itself automatically (e.g. the message 'openCard' on the opening of a card), and these messages also pass in the same hierarchy and can be intercepted by user scripts.

2. Architecture

The architecture we propose for the second version is quite close to that explained in [12]. There is a top-level implemented by one stack, and the "domains" relative to the documents, the ambiguities, the ambiguity types, and the disambiguation methods, are implemented in different stacks.

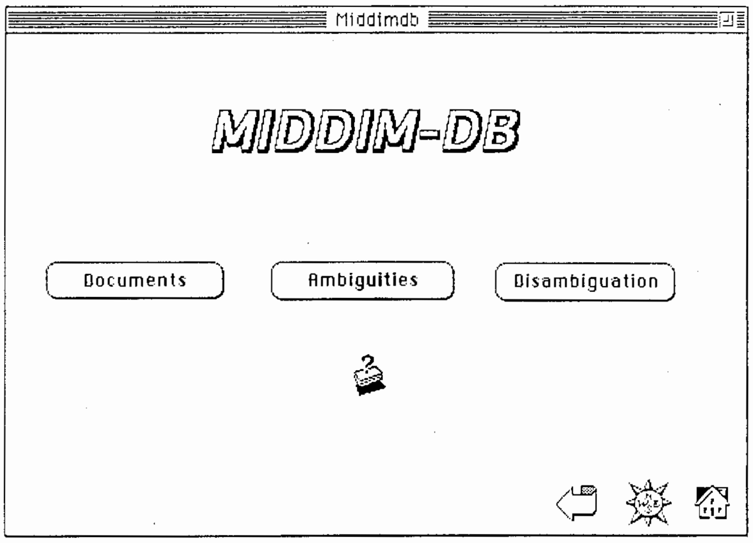
2.1. Top level: the MIDDIM-DB stack

2.1.a Main menu

The top level is implemented in the MIDDIM-DB stack. We show this card only to illustrate what buttons and fields are. The text "MIDDIM-DB" is contained in a field, which we lock to prevent users from editing it. Under it are three named buttons. Here is the script of the button "Documents".

```
on mouseUp
  go to card "Document menu"
end mouseUp
```

Scripts may be as simple as that, or contain many "handlers" and be quite complex. The other 4 buttons have associated icons, and don't show their names (but they could, at a click of the mouse in the scripting panel).

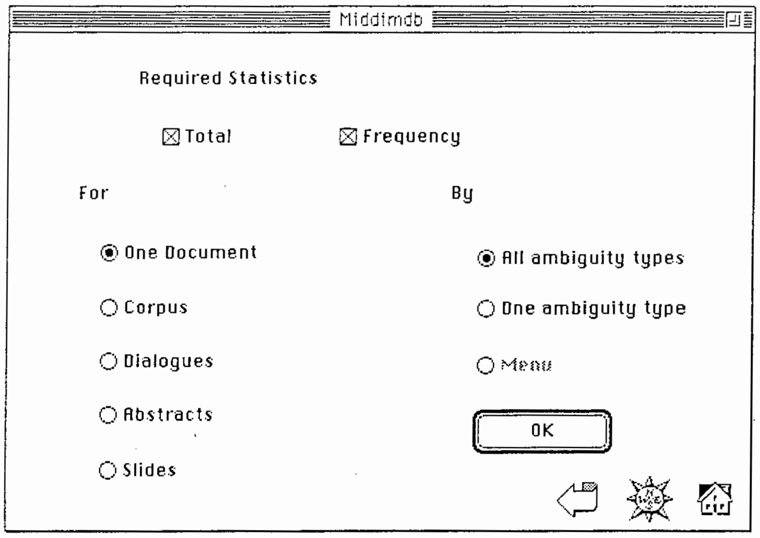


2.1.b Operations with documents

From the "Document menu" card, it is possible to browse through the catalogue of stored documents, to add or delete a document, and to open it in the original form or in ASCII form to inspect its content.

2.1.c Statistics on ambiguity types

From the Statistics card ("Amb Ren Menu"), we can require various sorts of statistics, on one document, on a type of documents, or on the whole collection of documents. Counts and statistics can be obtained for specific ambiguity types, or for all types.



Middimdb

Required Statistics

[✓ Total] [✓ Frequency]

For: [✓ One Document] Corpus Dialogues Abstracts Slides

By: [✓ All ambiguity types] One ambiguity type Menu

2.1.d Operations with disambiguation methods

In the same way, there is a catalogue of disambiguation methods, with types. Programmed methods must be attached as external commands. Methods simulated by other HyperCard stacks can just be referenced by name.

2.2. Documents

For a given document, we should be able to :

- read it.
- enter into the base the ambiguities that are in it, in such a way that we can access them after either by type or by document.
- calculate statistics on the ambiguities that are in it.
- access information on the document (a bibliographical reference, type etc.).

We should also be able to access statistics on the types of documents that are in the base, and a list of their titles.

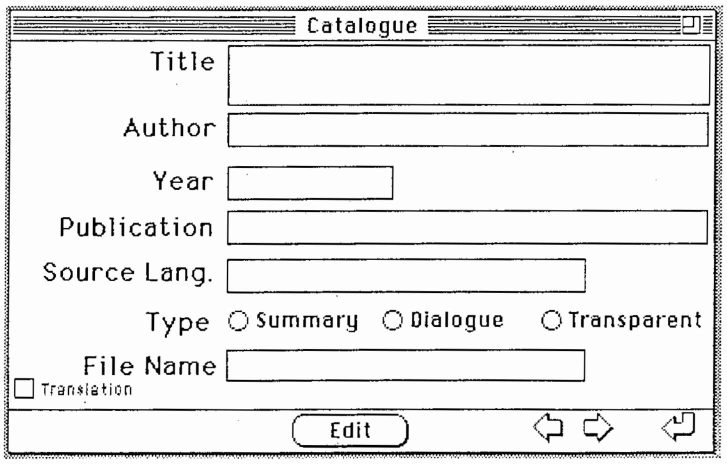
2.2.a Catalogue stack

The catalogue stack contains the information on all the documents (one card per document) and statistics on the number of documents by type (abstract, slide, telephone dialogue, multimodal dialogue), by original application (Word™, PowerPoint™, MacDraw™, WinText™...), etc.

A document card contains :

- the bibliographic reference (title, author, year etc..)
- the source language of the document
- the type of the document
- the name of the file containing the document (in order to link the information with the actual document).
- whether or not it is a translation.

This is a catalogue card for a document :



Catalogue

Title

Author

Year

Publication

Source Lang.

Type: Summary Dialogue Transparent

File Name

Translation

2.2.b Document stacks

When a document named "DDD" is added to the catalogue, a stack named "DDD.ref" is created to contain its image. It contains a card showing the entire document, with the same information as in the catalogue, and then cards associated with "paragraphs" and "sentences". These terms may be understood as "utterance" and "turns" in the case of dialogues.

The following cards correspond to the two levels of segmentation, with one card for each paragraph and one card for each sentence.

The cards provide buttons for modifying the segmentation interactively, exactly as in MIDDIM-DB.1.0.

We feel it is also necessary to provide an external exchange format, with associated import and export functions.

What has to be imported and exported is:

- the text (or transcription) and its 2-level segmentation.
- its representations (in one or several representation systems). We currently have fields for results of analysis by two parsers, one for GETA and one for ATR, and for hand-made representations.

2.2.c Ambiguity stacks

For each document named DDD, another stack, named "DDD.amb", is created to store the ambiguities.

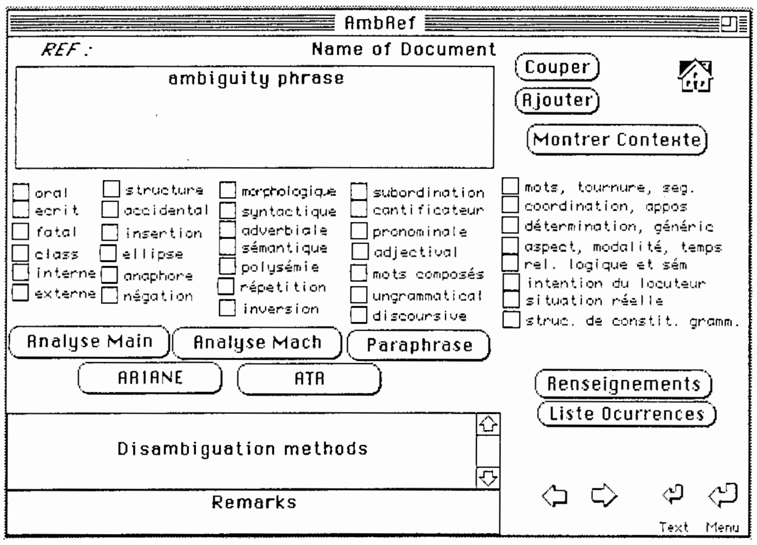
Here, what has to be imported and exported is, for each ambiguity:

- its support (the fragment F in the definition above) and its context (U above). Both may be represented as references to the text or as plain copies.
- the name of the representation system (R).
- the ambiguity type in the linguistic classification [14]
- the main structure (S) and the competing substructures s_i.

2.3. Ambiguity types

2.3.a Explanation of ambiguity types

Ambiguity types are combinations of features which characterize the differences between the competing substructures s_i from a linguistic point of view.



Here is an ambiguity card of MIDDIM-DB.1.0. We are currently working with M. Tomokiyo and researchers from GETA to design a simpler presentation of the classification used here.

Basically, then, an ambiguity type is a combination of binary features, expressing answers to questions such as:

- is the ambiguity accidental or fatal (does it appear in all contexts)?
- does it concern text, speech, or both?
- does it concern the morphosyntactic class?
- does it concern attachment, and in this case is it relative to coordination, prepositional attachment, quantifier scope, negation scope, etc.?

The last item in our list shows that we may have to organize the classification of ambiguities along a 2-level system of features, that is, using features and subfeatures. That would resemble the use of classification hierarchies in other parts of the description of linguistic phenomena. For instance, nouns may be refined into common nouns and proper nouns, proper nouns into person names and place names, place names into country names and others, etc.

2.3.b Guided tour in documents

The stack of ambiguity types was not specified in [12]. We think it maybe useful to introduce it, so that linguist developers, and later helper wizards, can develop a common understanding of the classification used. For this, the stack should provide a kind of "guided tour" of ambiguities, in example stacks annotated with explanations, and in the stacks associated with the documents.

2.3.c Simulation of interactive disambiguation methods on linguistic examples

Finally, this stack should provide examples of disambiguation methods applied to test cases. Again, these methods can be simulated, as was done by H. Blanchon at COLING-92 [3], or actually executed, as done in the LIDIA-1.0 prototype [13]

2.4. Disambiguation methods

In this domain, we have to be able to:

- add a method to the base, and produce links from it to one or more ambiguities
- activate methods from the ambiguities to which they apply
- initiate the disambiguation of a document
- initiate the disambiguation of a certain type of ambiguity in the base
- access information on the methods available in the base

This stack should also contain a "guided tour", with explanations and illustrations of each method.

3. How to make stacks multilingual

The solution used in MIDDIM-DB-1.0 to make the stack multilingual is very complex:

- A global variable must be created for each message.
- The content of each button has to be the list of its names in the possible dialogue languages, which prevents using it in normal ways, e.g. to generate a pop-up menu.
- Because of the desire to permit
  - to change the dialogue language at any time,
  - to edit the messages at any time, and, even more,
  - to add a new dialogue language at any time,

  all main scripts check whether one of these conditions is met, and behave accordingly.

At the same time, it is still incomplete, because button names used in scripts are not changed in the scripts when the dialogue language changes, which prevents the scripts to run as they should. Using the button identifiers instead of their names makes the scripts very difficult to read and debug.

What we propose to do, then, is the following:

- create a special card at the top level to handle everything concerning the management of the dialogue languages and of the messages.
- replace all messages and button names in scripts by calls to a unique handler, MiddimMsg(), the calls being of the form:

```
MiddimMsg (msg-type, msg-number, &1, &2… ,&n)
        -- msg-type is B for button-name and M for message
        -- msg-number is an integer identifying the message
        -- &1, &2… ,&n are parameters to substitute in corresponding
        -- place holders in the message pattern.
```

- physically change the names of the buttons only when a background is entered, and then do it for all the cards sharing this background. To do this, it is enough to
  - add one (hidden) "thisBackgroundDialogueLanguage" field to each background, and use one global variable "currentMiddimDialogueLanguage".
  - add one line to each "openBackground" handler, in order to pass the message to the "changeButtonNames" handler in the MiddimMsg card.

This type of solution has been used extensively in the Ariane-G5 MT shell, and should work as well here. A very big advantage is that it could in principle be applied to *any* existing stack, with extremely systematic and localized modifications.

Conclusion

We hope to implement these specifications before next April, and send first preliminary versions to ATR before January. Working on this data base is however not only a practical endeavour. As we have seen, it has prompted the quest for a precise, formalized definition of ambiguities. In the same vein, we hope to clarify the notion of "type of ambiguity" in the near future, with the help of linguists from GETA and from ATR, and to embody it in the new design.

Acknowledgments

Thanks should go to G. Fafiotte, K.-H. Loken-Kim, M. Seligman, and M. Tomokiyo, who carefully read the first draft and helped remove many errors and clarify several points, as well as to J. Winship, for the dedication and enormous amount of work she has put into MIDDIM-1.0. Thanks also to Dr. Y. Yamazaki, president of ATR-ITL, and Mr. T. Morimoto, head of Department 4, who have raised several questions and contributed to the clarification of several essential points.

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

References

[1] **Maruyama H., Watanabe H. & Ogino S. (1990)** *An Interactive Japanese Parser for Machine Translation.* Proc. COLING-90, Helsinki, 20-25/8/90, H. Karlgren, ed., ACL, vol. 2/3, 257-262.

[2] **Boitet C. (1989)** *Speech Synthesis and Dialogue Based Machine Translation.* Proc. ATR Symp. on Basic Research for Telephone Interpretation, Kyoto, December 1989, 6-5-1:22.

[3] **Blanchon H. & Guilbaud J.-P. (1992)** *LIDIA : the disambiguation process — le processus de désambiguïsation.* Nantes, 23—28/7/92, Pile HyperCard présentée à l'exposition COLING-92.

[4] **Blanchon H. (1993)** *Report on a stay at ATR.* Project report (MIDDIM), GETA & ATR-ITL, July 1993, 30 p.

[5] **Boitet C. (1993)** *Multimodal Interactive Disambiguation: first report on the MIDDIM project.* Technical Report, TR-IT-0014, ATR Interpreting Telecommunications, Aug. 93, 16 p.

[6] **Black E., Garside R. & Leech G. (1993)** *Statistically-Driven Grammars of English: the IBM/Lancaster Approach.* J. Aarts & W. Mejs, ed., Language and Computers: Studies in Practical Linguistics, Rodopi, Amsterdam, 248 p.

[7] **Boitet C. & Loken-Kim K.-H. (1993)** *Human-Machine-Human Interactions in Interpreting Telecommunications.* Proc. International Symposium on Spoken Dialogue, Tokyo, 10–12 November 1993, Waseda University, 4 p.

[8] **ATR-ITL (1994)** *Transcriptions of English Oral Dialogues Collected by ATR-ITL using EMMI (from TR-IT-0029, ATR-ITL).* EMMI report, ATR-ITL, January 1994, 33 p. (file edited at GETA).

[9] **Boitet C. & Axtmeyer M. (1994)** *Documents prepared for inclusion in MIDDIM-DB.* Internal report, GETA, IMAG (UJF & CNRS), June 1994, 3 p.

[10] **Axtmeyer M. (1994)** *Analysis of ambiguities in a written abstract (MIDDIM project).* Internal report, GETA, IMAG (UJF & CNRS), July 1994, 17 p.

[11] **Boitet C. & Blanchon H. (1994)** *Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup.* Machine Translation. (to appear).

[12] **Winship J. (1994)** *Building MIDDIM-DB, a HyperCard data-base of ambiguities and disambiguation methods.* ERASMUS project report, GETA, IMAG (UJF & CNRS) & University of Sussex at Brighton, 22 July 1994, 22 p.

[13] **Blanchon H. (1994)** *Perspectives of DBMT for monolingual authors on the basis of LIDIA-1, an implemented mockup.* Proc. 15th International Conference on Computational Linguistics, COLING-94, Kyoto, Japan, 5-9 Aug. 1994, 5 p. (to appear).

[14] **Tomokiyo M. (1994)** *Ambiguity analysis and MIDDIM-DB.* MIDDIM Report, TR-IT-0064, ATR-ITL & GETA-IMAG, Aug. 94, 74 p.

[15] **Winship J. (1994)** *Documentation of MIDDIM-DB, version 1.0.* Internal report, GETA, IMAG (UJF & CNRS) & University of Sussex at Brighton, 17 August 1994, 100 p.

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

Contents

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