A Customizable Interactive Disambiguation Methodology and Two Implementations to Disambiguate French and English Input

Hervé BLANCHON[†]

GETA-CLIPS
150, rue de la Chimie
BP 53
38041 Grenoble Cedex 9, France

herve.blanchon@imag.fr

Abstract

As natural language is highly ambiguous even in restricted domains, interactive disambiguation is seen as a necessity for achieving more robust and user-friendly interactive systems, face-to-face translation systems and Dialogue-Based Machine Translation systems. We have proposed a methodology which distinguishes between two parts in a disambiguation module: an engine (language- and application-independent) and a lingware (language- and application-dependent). The engine is, thus, to be reused in the design of any disambiguation module. This paper presents the current state of our work, that is: an engine that has been used to design two interactive disambiguation modules, for French, and English.

Keywords

Interactive disambiguation, software for NLP, machine translation, disambiguation dialogue.

Introduction

Spoken, written or multimodal natural language is going to be used more and more as the interaction modality between human users and interactive software. Natural language processing techniques do not, and will not, allow a really robust, fault-tolerant and user-friendly utilization of these modalities.

As far as textual input is concerned, some problems can occur with spelling or syntactic consistency. However, the most important difficulty is the ambiguousness of language. In fact, ambiguities can arise even if the domain of the utterance is well-defined and many knowledge sources are involved in the analysis process.

[†] Part of this work was sponsored by ATR-ITL from July 94 to August 95.

As far as spoken input is concerned, the first problem concerns speech-to-text transformation. Then, difficulties related to textual input arise. In speech-to-text transformation the difficulties stem mostly from two problems: segmentation and variability. Moreover, even when the segmentation has been done without any ambiguity, the problem of homophony has to be solved.

Multimodality combines the problems of speech with other specific problems. Among those one can think of gesture recognition, the unambiguous selection of any hand- (or mouse-) pointed objects, etc.

For the use of natural language to be more robust, fault-tolerant, and user-friendly, we propose to integrate a disambiguation module as a component of every relevant piece of interactive software. The role of such a module is to help the recognizer (for speech) and the analyzer (for text) to produce an unambiguous representation of the user's input corresponding to his intention.

In our opinion, disambiguation has not yet been studied as a core research framework; this is what we would like to advocate. On the other hand, many ambitious projects are using natural language as a communication modality between the system and the user. Thus, we think that the need for a clarification framework will be felt more and more.

As a first step in this direction we would like to propose an interactive disambiguation methodology which is based on a clear distinction between two components: a language-independent one (the engine), and a language- and application-dependent one (the lingware).

As the language independent part, the engine should be used in every designed disambiguation module. For each specific application, a specifically designed lingware should be produced as automatically as possible.

In this paper we will first introduce our basic ideas and proposals. In the second part we will present the current content of the lingware and then the existing implemented engine. The fourth part is dedicated to a brief presentation of our first experiments with the lingware. A first interactive disambiguation module has been designed and implemented, for French input, within a first mockup in the framework of the LIDIA project of Dialogue-Based Machine Translation at the GETA-CLIPS Lab. A second one has been designed and implemented, for English input, at ATR-ITL as a study of interactive disambiguation in the framework of speech-to-speech translation.

I. Basic Ideas & Proposals

I.1. Software architecture

In the architecture we propose, an interactive disambiguation module is made of two components.

– An engine, which is the core of the module and is language-independent. It will be used in all the disambiguation modules to be developed.
– A lingware, which is language-dependent. It constitutes input data for the engine so as to instantiate a particular disambiguation module.

The very long term goal of this research is to build an interactive environment to develop disambiguation modules to be integrated in any given application. We are thus aiming for application independence, with the restriction that the linguistic structures produced by the application can be handled in the proposed framework.

I.2. Lingware

Ideally, we would like to provide the designer of a disambiguation module with a set of tools allowing him/her, at least, to describe[1]:

– the ambiguities to be solved, even though most of then should be discovered automatically,
– the labeling of the questions to be asked to solve these ambiguities,
– the order in which the ambiguities should be solved, if several are present,
* the modalities to be used to solve each ambiguity,
* the modalities to be used to answer the questions about each ambiguity,
* the way questions should be prepared and displayed.

I.3. Engine

The engine is then designed to use the lingware to instantiate an interactive disambiguation process. It should provide[1] :

– an ambiguity description language,
* an augmentation tool to increase automatically or semi-automatically the number of recognized and solved ambiguities,
– an ambiguity recognition mechanism to be used to recognize the ambiguities,
– a set of operators to be used to describe the construction of the labeling of the dialogues,
– a mechanism to realize the ordering of the ambiguity recognition process,
– predefined dialogue classes corresponding to the possible modalities,
– a question presentation mechanism,
* a set of question preparation, display and answering strategies.

Let's move now to the actually implemented engine and lingware. We will first describe the content of the lingware.

II. The lingware

The structure manipulated in our context is a tree structure called an "mmc structure." This stands for "multisolution, multilevel and concrete." The ambiguities are thus described in terms of tree structures.

II.1. Ambiguity description

A type of ambiguity is described as a set of patterns called a beam (Fig. 1). A pattern describes a tree structure, containing variable parts (forests), with constraints on their geometry and labeling.

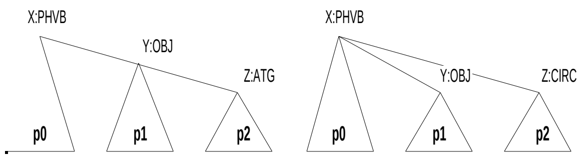


Figure 1. The patterns for a phvb prepositional attachment type 1 ambiguity

[1] The items introduced by "–" have already been investigated, the items introduced by "*" have not been investigated yet.

The previous figure (Fig. 1) shows the graphic description of a beam made of two patterns. In those patterns, the nodes are X, Y, and Z; the forest variables are P1, P2, and P3. This graphical representation makes explicit the geometrical constraints.

The two following figures (Figs 2 & 3) show the internal representation of the patterns shown in the previous figure.

```
(defvar *phvbprepatt-t1-1*
 (make-instance 'pattern
  :pattern-name '*phvbprepatt-t1-1*
  :pattern-value
    '((?is ?x node-prop-equal-p 'CS 'PHVB)
      (?+ ?p0)
      ((?is ?y node-prop-equal-p 'FS 'OBJ)
       (?+ ?p1)
       ((?is ?z node-prop-equal-p 'FS 'ATG)
        (?+ ?p2))))
 :pattern-method #'item-production-method))
```

*Figure 2. The pattern *2phvbadvatt-t1-1**

```
(defvar *phvbprepatt-t1-2*
 (make-instance 'pattern
  :pattern-name '*phvbprepatt-t1-2*
  :pattern-value
    '((?is ?x node-prop-equal-p 'CS 'PHVB)
      (?+ ?p0)
      ((?is ?y node-prop-equal-p 'FS 'OBJ)
       (?+ ?p1))
      ((?is ?z node-prop-equal-p 'FS 'CIRC)
       (?+ ?p2)))
 :pattern-method #'item-production-method))
```

*Figure 3. The pattern *2phvbadvatt-t1-2**

It is shown that a pattern has a name, a value (its description), and a method.

The node variables are ?x, ?y, ?z; they are constrained on the value of one of their fields. In the first pattern (Fig. 2):

– the value of the field CS of the node ?x must be PHVB,
– the value of the field FS of the node ?y must be OBJ, etc.

The forest variables are ?p0, ?p1, ?p2. They are constrained on their length by the symbol ?+ stipulating them not to be empty.

The method item-production-method is described in § II.2.

A class (or family) of ambiguity is described as a set of beams called a stack. For example, the ambiguities of coordination can be described as a class of ambiguity, and of course, there are several types of ambiguity of coordination.

The following figure shows a beam called *phvbprepatt_set_1* made of two patterns: *phvbprepatt-t1-1* (Fig. 2) and *phvbprepatt-t1-1* (Fig. 3).

```
(defvar *phvbprepatt_set_1*
 (make-instance 'pattern-beam
  :beam-name '*phvbprepatt_set_1*
  :beam-value (list *phvbprepatt-t1-1*
                    *phvbprepatt-t1-1*)))
```

*Figure 4. The beam *2phvbadvatt_set_1**

II.2. Ambiguity resolution

A dialogue item production method is associated with each pattern to describe a non-ambiguous rephrasing of the part of the sentence recognized by the pattern. This rephrasing is produced using the forest variables instantiated during a successful match. The variables are manipulated with a set of operators described in section III.1.2.

The following figures (Fig. 5 & 6) show the dialogue item production methods associated with the pattern *phvbprepatt-t1-1* (Fig. 5) and that associated with the pattern *phvbprepatt-t1-2* (Fig. 6).

```
(defmethod item-production-method
  ((pattern-name (eql '*phvbprepatt-t1-1*))
   binding)
 (format nil "~A (~A ~A)."
      (text (cdr (assoc '?p0 binding)))
      (text (cdr (assoc '?p1 binding)))
      (text (cdr (assoc '?p2 binding)))))
```

*Figure 5. Item-production-method ((pattern-name (eql '*phvbprepatt-t1-1*)) binding)*

```
(defmethod item-production-method
  ((pattern-name (eql '*phvbprepatt-t1-2*))
   binding)
 (format nil "~A, ~A ~A."
      (text (cdr (assoc '?p2 binding)))
      (text (cdr (assoc '?p0 binding)))
      (text (cdr (assoc '?p1 binding)))))
```

*Figure 6. Item-production-method ((pattern-name (eql '*phvbprepatt-t1-2*)) binding)*

The first method (Fig. 5) will produce a string made of the text of p0, the text of p1 and the text of p2 with the texts of p1 and p2 bracketed together.

The second method (Fig. 6) will produce a string made of the text of p2, then a comma, then the text of p0 and the text of p1.

II.3. Ambiguity recognition state

The order in which the different ambiguity classes are to be solved has to be defined. An automaton is used to implement this order. In this automaton, there is one state per ambiguity class defined. The states are organized so as to uphold this order.

The skeleton of those states is the same:
– if one ambiguity of the class of ambiguity to be recognized is actually recognized; then a question has to be prepared and other ambiguities have to be found in the concerned sentence,
– if no ambiguity of the class of ambiguity to be recognized is recognized; then there is transition to the following state in the automaton.

II.4. Localized dialogue boxes

In the current implementation the questions to be asked to the user are displayed as dialogue boxes on the screen. These dialogue boxes contain the ambiguous phrase and a set of rephrasing items from which the user will choose the intended one.

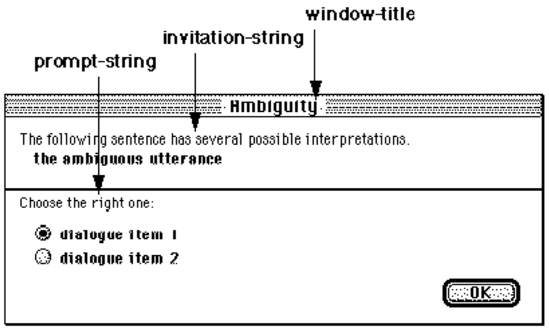
The language used to present the question to the user is the language to be disambiguated. Thus there is a need to specify some generic classes that are provided. This consists in defining the fonts and styles to be used to display the question as well as the meta-text to be used.

```
(defclass english-general-textual-
dialog-class (generic-textual-clarif-
dialog-class)
  ((window-length
    :initform 550)
   (invitation-string
    :initform "The following sentence has
several possible interpretations.")
   (invitation-string-font
    :initform '("helvetica" 16))
   (ambiguous-string-font
    :initform '("helvetica" 16 :bold))
   (prompt-string
    :initform "Choose the right one:")
   (prompt-string-font
    :initform '("helvetica" 16))
   (items-font
    :initform '("helvetica" 16 :bold)))
  (:default-initargs
    :window-title "Ambiguity"))
```

Figure 7. The lingware class english-general-textual-dialog-class

For example Figure 7 shows a specification of the `generic-textual-clarif-dialog-class` textual dialogue box class for English. The invitation-string, the prompt-string, and the window-title are labeled in English. If the question were to be asked in French, the generic-textual-clarif-dialog-class would have to be specified for that.

Figure 8 gives a visual description of the textual items that are specified.



The following sentence has several possible interpretations.
the ambiguous utterance
Choose the right one:

Figure 8. Some dialogue boxes' slots

III. The engine

III.1. Ambiguity recognition & question construction

1.1. Pattern & beam matching

1.1.1. Pattern matching

The patterns are described with a language derived from the one proposed in [17]. Pattern matching also inspired by the proposals in [17] has been implemented.

The result of the pattern matching mechanism is a list whose first element is `t` if matched, `nil` if not, and whose second element is a binding list containing the value of each variable in the pattern.

1.1.2. Beam matching

A sentence **S**, with **s** solutions $\mathbf{Sol_i}$, contains the ambiguity described by the beam **B** made of **b** patterns $\mathbf{P_j}$ if and only if:
– the number of solutions (**s**) is strictly greater than the number of patterns (**b**),
– for each solution $\mathbf{Sol_i}$ there is an unique pattern $\mathbf{P_j}$ that matches that solution,
– each pattern $\mathbf{P_j}$ matches at least one solution $\mathbf{Sol_i}$,
– the distance **fd** between the bindings of the forest variables is null.

That is:

– *1* **b** < **s**
– *2* i, !j / **match-p**(S_i, P_j)=t
– *3* j, i / **match-p**(S_i, P_j)=t

$$
\begin{aligned}
-\ 4 \quad & i, i', \ j, j' \\
& \mathbf{match\text{-}p}(S_i, P_j)=t \\
\text{and } & \mathbf{match\text{-}p}(S_{i'}, P_{j'})=t \\
\Leftrightarrow \ & \mathbf{fd}(\ \mathbf{binding}(S_i, P_j), \\
& \mathbf{binding}(S_{i'}, P_{j'}))=0
\end{aligned}
$$

Figure 9. Beam matching definition

The distance between the forest variables of two bindings, **fd**, is null if and only if:

- the coverage of each forest variable, except the last one, is the same in each binding, and
- for the last forest variable of the patterns, if the coverages are not the same, one coverage has to be a prefix of the others.

The **coverage** of a variable is the projection of the leaves of the subtree this variable represents.

1.2. Operators

Operators are used to describe the dialogue items' construction. They are used by the prepare-question-tree module (cf. Fig. 13) and allowed to perform several operations on the binding of the variables. Three families of operators are defined to perform: selective projection, access to the lexical database, and formatting operations.

1.2.1. Selective projection

Operators of the first family describe some manipulations of subtree structures, basically the selection or the suppression of some part of the trees.

Text	produce the text of the linguistic trees given as parameter.
Subject	produce the text of the subject of the linguistic trees given as parameter.

– note: There is such a function for each syntactic function and syntagmatic class. Example: VerbalGroup, CircComp...

Coord	produce the coordinating occurrence of the linguistic trees given as parameter.
But_Coord	produce the text of the linguistic trees given as parameter without the coordinating occurrence.
But_Sub	produce the text of the linguistic trees given as parameter without the subordinating particle.
But_Det	produce the text of the linguistic trees given as parameter without the determiner.
Project	applied to a leaf of the mmc-structure, project the occurrence and the syntactic class of the occurrence. In the future some other information may also be proposed.

1.2.2. Access to the Multilingual Lexical Data Base

Agreement	produce the form of an adjective according to gender and number constraints.

```
ex :    Agreement("noirs",(("gn" [...] (gnr fem nbr
                  plu)) ([...]))) -> "noires"
```

Substitute	replace an ambiguous preposition by a non-ambiguous one (in the context) according to several properties.

```
ex :    Substitute("de", #Objet_1) -> "à propos de"
```

Definition	produce the definition of the occurrence given as the parameter.

1.2.3. Other operations

Operators of the third family describe some more complex operations of distribution and bracketing of subtrees.

Distribute	distribute an occurrence or a group of occurrences over other groups of occurrences and link the new groups with a preposition of coordination.

– note: The distribution is done making the gender and the number of the adjective agree with the gender and the number of the different substantives

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

Bracket	brackets the text of the arguments.

1.3. Disambiguation question class

The disambiguation process produces a question-tree made of disambiguation questions. Those questions are to be asked to the user according to the modalities to be used.

In the current implementation, the questions are displayed as dialogue boxes on the screen. Each disambiguation question is an instance of the predefined class clarification-question-class whose slots are defined as follows:

- question-language is the language disambiguated and thus the metalanguage to be used to present the question;
- question-type is the type (the class) of ambiguity to be solved with the question;
- question-modality is the modality(ies) to be used to ask the question. It is textual. In the future it may be any combination of textual, spoken, or drawn modalities;

– ambiguous-item is the utterance to be disambiguated with the question;
– question-items-list: is the list of the different choices to be proposed to the user.

There is also a particular class called empty-question that is a subclass of the clarification-question-class. This class of question is used to construct the leaves of the question trees. It is defined with two slots with the following meanings:
– concerned-sol is the number of the chosen solution.
– concerned-tree is the analysis tree associated with the chosen solution.

III.2. Disambiguation automaton

Each state of the automaton is defined as a CLOS method [15]. There are basically three kinds of states:
– an entry point, that is, the first state of the automaton. It is called automaton-scheduler and provided by the engine,
– meta-ambiguity recognition states provided by the engine, and,
– ambiguity class recognition states provided by the lingware.

2.1. Automaton entry point

The method automaton-scheduler is the entry point of the disambiguation automaton. If there is no ambiguity to be solved, this state is also the exit point of the automaton. When there are no more questions to be prepared, an empty question is produced which is a leaf of the question tree .

This method is specified on the language parameter so that the entry point name of each automaton is the same.

2.2. Meta-ambiguity recognition states

A meta-ambiguity recognition state is a predicative state used as a branching state in the automaton. So far we have proposed three ambiguity meta-classes. These classes are called: lexical-ambiguity, geometrical-ambiguity and labeling-ambiguity. They are refined by the designer of the lingware into several designer-defined classes of ambiguity.

2.3. Scheduling

Thus, a disambiguation automaton should be shaped as shown in Figure 10.

The lexical ambiguities are to be solved first, then the geometrical ones, and finally the labeling ones. This strategy is guided by the following principles:
1 first, find the right simple phrases (i.e. solve the lexical ambiguities),
2 second, find the construction of the verbs (i.e. solve the labeling and some of the geometry ambiguities),
3 third, find the structure of the dependents of the verbs (i.e. solve the geometry ambiguities which did not fall in the previous case),
4 last, find the word senses.

Those criteria seem reasonable and natural. Moreover, the order of the kinds of questions will not be changed to improve the usability of the system.

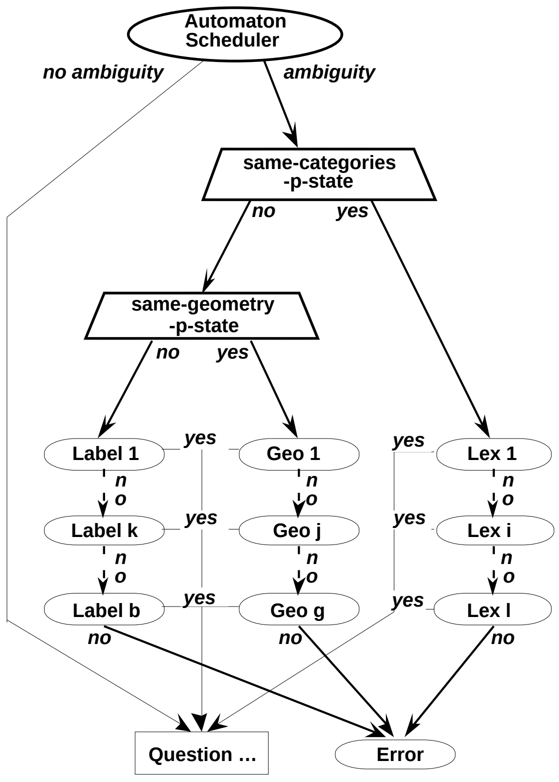


Figure 10. General organization of a disambiguation automaton

III.3. Question construction & presentation

3.1. Question construction

```
(defmethod prepare-question
 ((the_language (eql 'english))
  the_type
  (the_modality (eql 'textual))
  the_sentence
  the_list_of_triplets)
 ;construction the dialogue items
 ;creation of clarification-question to be
 ;inserted in the current question tree
```

Figure 11. The engine method prepare-question

Once an ambiguity has been recognized, a question is produced with the method `prepare-question` defined in Figure 11.

A `triplet` is a list of three elements: the name of the pattern which has been matched, the resulting binding, and a list of the index numbers of the solutions concerned with the matching of the pattern. This triplet is used to construct a dialogue item.

The `produce-item` is implemented as follows:

```
(defun produce-item (self)
 (let ( (pattern (first self))
        (patt_name (pattern-name pattern)
        (binding (second self))
        (concerned_solutions (third self)))
 ;the pattern-method associated with the
 ;pattern is applied to the binding
  (list (apply (pattern-method pattern)
               (list patt-name binding))
        the_concerned_solutions)))
```

Figure 12. The engine function produce-item

`self` is a triplet as described above. The method associated with the pattern is applied to the binding and produces a dialogue item. This dialogue item is associated with the solutions produced by the analyses that are concerned with the item.

3.2. Question tree construction

The construction of the question tree is a loop in a disambiguation automaton organized by the `prepare-question-tree` method (Fig. 13).

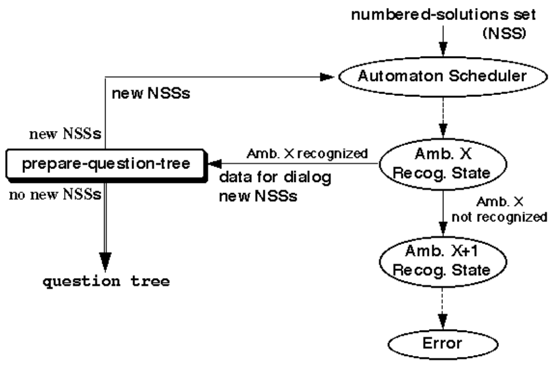


Figure 13. The question-tree construction

When a question tree has been built, it is presented to the user by the `question-tree-presentation` method.

3.3. Question tree presentation

For a given ambiguous utterance, the disambiguation automaton produces a question tree. The question tree is covered by the `question-tree-presentation` function until no more questions are to be asked. The method `ask-question` proposes the question to the user.

IV. Implementation

So far we have developed two disambiguation modules; one for French and one for English. They are briefly described here.

IV.1. A French input disambiguation module

1.1. Context

The module for French has been developed at the GETA laboratory in the framework of the LIDIA project of Dialogue-Based Machine Translation [1, 2].

It is currently made of 13 beams.

1.2. Corpus

For the LIDIA-1 mockup we constructed a corpus of ambiguous sentences. Those sentences were chosen from the literature about ambiguity in French [11, 13, 14, 20]. The sentences were selected according to their underlying linguistic structure. The wording was not considered.

1.3. Classification & example

We have defined the following ambiguities described by means of beams.

<u>Lexical Ambiguity</u>

- *Verbal coordination*

Il atteint la grange et la ferme[2].

<u>Geometrical ambiguity</u>

- *Argument structure of the verb*

Il parle depuis l'école de cuisine[3].

Le capitaine a rapporté un vase de Chine[4].

- *Noun coordination*

On étudie l'écolution de la structure de la bourse et des investissements[5].

---

2 He reaches the barn and the farm *or*, He reaches the barn and closes it.

3 He speaks from the school, about cooking *or*, He speaks from the cooking school.

4 The captain brought a vase from China *or*, The captain brought a Chinese vase.

5 We are studying the evolution of the structure of the investments *or*, We are studying the investments.

- *Adjective coordination*

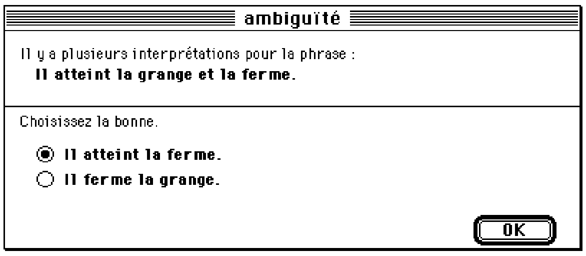
Il prends des cahiers et des classeurs noirs[6].

- *Subordination*

Elle épouse un professeur de droit anglais[7].

1.4. Dialogues

Here is a set of examples of the produced dialogues.

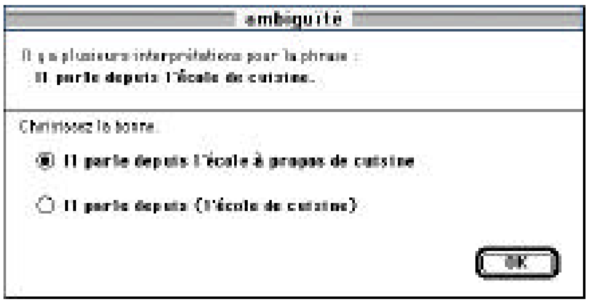

ambiguïté

Il y a plusieurs interprétations pour la phrase :
Il atteint la grange et la ferme.

Choisissez la bonne.

[✓ **Il atteint la ferme.**]
Il ferme la grange.

Figure 14. Dialogue for a verbal coordination ambiguity[8]

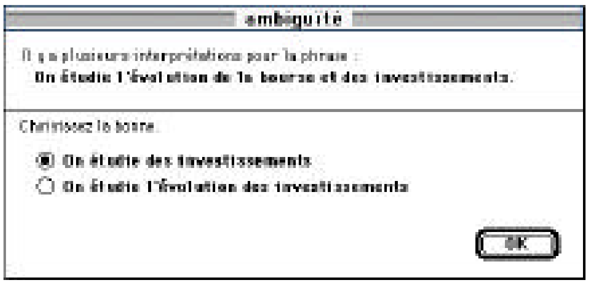

ambiguïté

Il y a plusieurs interprétations pour la phrase :
Il parle depuis l'école de cuisine.

Choisissez la bonne.

[✓ **Il parle depuis l'école à propos de cuisine**]
Il parle depuis (l'école de cuisine)

Figure 15. Dialogue for an ambiguity of argument structure of the verb, type 1 [9]


ambiguïté

Il y a plusieurs interprétations pour la phrase :
On étudie l'évolution de la bourse et des investissements.

Choisissez la bonne.

[✓ **On étudie des investissements**]
On étudie l'évolution des investissements

Figure 16. Dialogue for a noun coordination type 1 ambiguity[10]

6 Black files *or,* Black files and black notebooks.

7 An English professor *or,* A professor of English law.

8 The proposed interpretations are: He reaches the farm, He closes the barn.

9 The proposed interpretations are: He speaks from the school about cooking, He speaks from (the cooking school).

10 The proposed interpretations are: We study the investments, We study the evolution of the investments.

IV.2. An English input disambiguation module

2.1. Context

The module for English has been developed at ATR-ITL in the framework of Speech Translation [3, 4, 8].

It is currently made of 10 beams.

2.2. Corpora

The ambiguities which make up the first corpus upon which the clarification mechanism was based were taken from a data base of spontaneous speech collected at ATR-ITL. The conversations, between native speakers of American English, were recorded during an experiment conducted in the Environment for MultiModal Interaction (EMMI) [16], and took place via both telephone and multimedia communication contexts [10]. The 17 conversations from the experiment, comprising over 12,000 words, were examined by hand, and all detected ambiguities were extracted. Ambiguities due solely to polysemy were disregarded.

A second corpus was built using two other experiments carried out at ATR [18, 19] The analysis trees were also produced. This corpus has not been yet used in the current implementation of the English module.

A description of the ambiguities to be solved is given in [9] in this volume.

2.3. Classification & examples

So far we have defined the following ambiguities described by means of beams.

<u>Lexical Ambiguity</u>

- *Noun adjective*

This is an English speaking agent.

<u>Geometrical ambiguity</u>

- *Prepositional attachment*

Let me pull out my maps to help you.

Where can I catch a taxi from Kyoto station.

You are going to the International Conference Center.

- *Adverbial attachment*

I will show you where you are located right now.

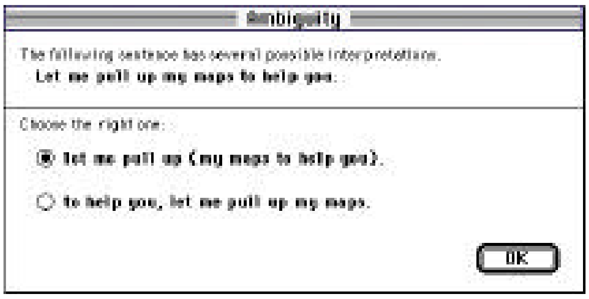
You can pay for it right on the bus

- *Verbal-phrase coordination attachment*

You can tell him that you are going to the international conference center and it should be a twenty minute ride.

2.4. Dialogues

Here is a set of examples of the produced dialogues.



Ambiguity

The following sentence has several possible interpretations.
Let me pull up my maps to help you.

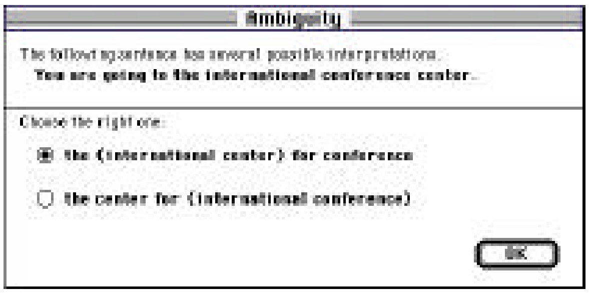
Choose the right one:

[✓ Let me pull up (my maps to help you).]

to help you, let me pull up my maps.

OK

Figure 17. Dialogue for "Let me pull up my maps to help you."



Ambiguity

The following sentence has several possible interpretations.
You are going to the international conference center.

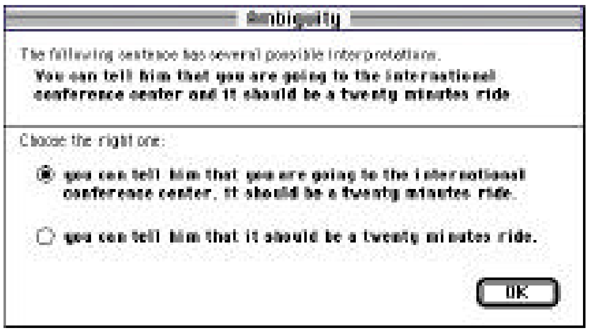
Choose the right one:

[✓ the (international center) for conference]

the center for (international conference)

OK

Figure 18 . Dialogue for "You are going to the international conference center."



Ambiguity

The following sentence has several possible interpretations.
You can tell him that you are going to the international conference center and it should be a twenty minutes ride

Choose the right one:

[✓ you can tell him that you are going to the international conference center, it should be a twenty minutes ride.]

you can tell him that it should be a twenty minutes ride.

OK

Figure 19. Dialogue for "You can tell him that you are going to the international conference center and it should be a twenty minutes ride."

Conclusion & perspectives

On the methodology

The methodology we proposed should allow the development of customized disambiguation modules that can be easily improved incrementally.

The customizability comes from the clear separation between the lingware and the engine. In this framework, different disambiguation modules can be produced for one or several different languages and kinds of input.

The description of the linguistic data can be augmented incrementally as the design and the use of a disambiguation module progress.

Certainly, we do not claim that any given module will cover all the ambiguities found in natural language (written, spoken or multimodal). On the other hand, we claim that a given module for a given application can incrementally reach a broad coverage for the application it has been designed to be integrated into.

The anticipated evolution of the ambiguity recognition coverage is given in the following figure.

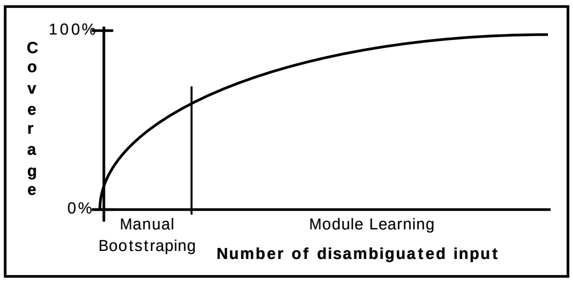


Figure 20. Evolution of the coverage of a given disambiguation module

On the evaluation

We think that, whenever a system uses a natural language analysis module, the evaluation criterion must not be simply the task completion time. More important is the user satisfaction. That is why we feel that it is very important to study the design of the clarification sessions, and moreover, the design of the questions to be asked. We are aware that this kind of study is energy- and time-consuming but it has to be done if we are aiming to build real-scale usable systems.

Thus, we have proposed to run experiments to study, before other questions, the understandability of the proposed disambiguation dialogues. We have already run two experiments on that topic.

– A pilot experiment [5] allowed us to gathered some interesting feedback and to learn how not to design such an experiment.
– A second experiment [7], allowed more precise and measurable results.

The results of these two experiments are also described in this volume [6].

We strongly hope that it will be possible to carry on this work on understandability and assessment.

Technical perspectives

There is of course a lot of work to be carried out to reach our goal. Here are the most challenging ones:

The (semi-) automatic augmentation of the description of the lingware. That is:

- the learning of new patterns, and beams[11],
- the construction of the dialogue item production methods to be associated with the new patterns (maybe with the use of analogy),
- the updating of the disambiguation automaton.

The manipulated data structures may be weighted so that the module can learn from the history of the dialogue (adapt itself to the user), and be tunable.

- Weighted patterns would be reordered inside each beam. The most likely selected item should be selected by default. Also, if likelihood of answer > user customized value then the question may not be asked
- Weighted beams would be reordered to speed up the system. Also, if likelihood of beam < user customized value then the beam may not be tested
- Weighted ambiguities may enable the reordering of the states of the automaton to speed up the system. Also, if likelihood or importance of ambiguity < user customized value then automatic selection

We do also think that it will be necessary to provide an interactive environment to enable the design of a disambiguation module.

The handling and the use of several modalities also have to be studied.

Acknowledgment

I would like to thank Drs. Yamazaki, Morimoto and Loken-Kim who invited me to ATR-ITL for 13 months to work on that topic and let me develop my ideas. I am also grateful to L. Fais for reviewing earlier drafts of this paper. The remaining deficiencies are of course mine.

11 We are already investigating this point [12] .

References

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

[2] **Blanchon, H.**, (1994). *Pattern-based approach to interactive disambiguation: first definition and implementation.* Rap. ATR-Interpreting Telecommunications Research Laboratories. Technical Report. n° TR-IT-0073. Sept. 8, 1994.

[3] **Blanchon, H.**, (1995). *An Interactive Disambiguation Module for English Input: an Engine and the Associated Lingware.* Rap. ATR-ITL. Technical Report. n° TR-IT-`0129. Aug., 1995.

[4] **Blanchon, H.**, (1995). *An Interactive Disambiguation Module for English Natural Language Utterances.* Proc. NLPRS'95. Dec 4-7, 1995. vol. **2/2** : pp. 550-555.

[5] **Blanchon, H. & Fais, L.**, (1995). *Pilot Experiment on the Understandability of Interactive Disambiguation Dialogues.* Rap. ATR-ITL. Technical report. n° TR-IT-0177. Sept., 1995.

[6] **Blanchon, H. & Fais, L.**, (1996). *How to ask Users About What they Mean: Two Experiments & Results.* Proc. MIDDIM'96. 12-14 Août 1996. *To be published.* 22p.

[7] **Blanchon, H. & Fais, L.**, (1996). *A Second Experiment on the Understandability of Interactive Disambiguation Dialogues.* Rap. ATR-ITL. Technical Report. n° TR-IT-0167. April, 1996.

[8] **Blanchon, H. & Loken-Kim, K.-H.**, (1994). *Towards More Robust, Fault-Tolerant and User-Friendly Software Integrating Natural Language Processing Components.* in Bulletin of the Information Processing Society of Japan (94-SLP-4). vol. **94**(109) : pp. 17-24.

[9] **Fais, L. & Blanchon, H.**, (1996). *Ambiguities in Task-oriented Dialogues.* Proc. MIDDIM'96. 12-14 Août 1996. *To be published.* 15 p.

[10] **Fais, L. & Loken-Kim, K.-H.**, (1994). *Effects of communicative mode on spontaneous English speech.* Rap. Institute of Electronics, Information and Communication Engineers. Technical Report. n° NLC94-22. Oct. 94.

[11] **François, D. & François, F.**, (1967). *L'Ambiguïté Linguistique.* in WORD, Journal of the linguistic circle of New York. vol. **23**(1-2-3) : pp. 150-179.

[12] **Frey, V.**, (1995). *De la reconnaissance à la découverte de "patrons d'ambiguïtés" en TAFD.* Rap. Université Josephe Fourier (MATIS - Archamps). Rapport de DEA. 24 juin 1996.

[13] **Fuchs, C.** (ed.), (1985). Aspects de l'ambiguïté et de la paraphrase dans les langues naturelles. Peter Lang. Berne. 215 p.

[14] **Fuchs, C.**, (1987). *L'ambiguïté et la paraphrase en linguistique.* in L'ambiguïté et la paraphrase. Centre de Publication de l'Université de Caen. Caen. pp. 15-20.

[15] **Keene, S. E.**, (1989). *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS.* Addison-Wesley Publishing Company. New York. 266 p.

[16] **Loken-Kim, K.-H., Yato, F., Kurihara, K., Fais, L. & Furukawa, R.**, (1993). *EMMI-ATR environment for multi-modal interactions.* Rap. ATR-ITL. Technical Report. n° TR-IT-0018. Sept. 30, 1993.

[17] **Norvig, P.**, (1992). *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp.* Morgan Kaufmann Publishers. San Mateo, California. 945 p.

[18] **Park, Y.-D. & Loken-Kim, K.-H.**, (1994). *Text Database of Telephone and Multimedia Multimodal Interpretation Experiment.* Rap. ATR-ITL. Technical Report. n° TR-IT-0086. Dec., 1994.

[19] **Park, Y.-D., Loken-Kim, K.-H., Mizunashi, S. & Fais, L.**, (1995). *Transcription of the Collected Dialogue in a Telephone and Multimedia/Multimodal WOZ Experiment.* Rap. ATR-ITL. Technical Report. n° TR-IT-0090. Feb., 1995.

[20] **Vauquois, B. & Nédobejkine, N.**, (1977). *Ambiguïtés de la langue écrite.* Rap. Action Thématique Programmée de Linguistique Genérale, Table ronde organisée au CNRS. 11 janvier 1977.