An Interactive Disambiguation Module for English Natural Language Utterances

Hervé Blanchon[†‡], Kyung-Ho Loken-Kim[†], Tsuyoshi Morimoto[†]

[†]ATR-ITL
2-2 Hikaridai
Seika-cho, Soraku-gun
Kyoto 619-02 Japan
{blanchon, kyungho, morimoto}@itl.atr.co.jp

[‡]GETA-CLIPS
150 rue de la Chimie
B.P. 53
38041 Grenoble Cedex 09 France
herve.blanchon@imag.fr

Abstract

An interactive disambiguation methodology has been proposed and implemented at the GETA lab. in the framework of Dialogue-Based Machine Translation. This methodology has been generalized and re-engineered at ATR-ITL in the framework of spoken language transtlation and the MIDDIM project, a joint research between ATR-ITL and the GETA-CNRS aimed to study Multimodal Interactive Disambiguation.

The disambiguation methodology is based on the manipulation of tree structures. A kind of ambiguity is described with a set of patterns called a beam. A pattern contains variables and descibes a tree structure with constraints on its geometry and labelling. Once a beam has been recognized, a question is prepared. The question items are produced through the manipulation, with a set of basic operators, of the values given to the variables instanciated during the recognition of the beam.

A particular disambiguation module is described with a lingware which is language and analyzer dependant. This lingware is then used as input data to an interactive disambiguation engine so as to describe a particular instance of a running disambiguation module. In this paper we are going to describe the engine and the lingware we have developed at ATR-ITL for English input.

Keywords

Interactive disambiguation, pattern-matching, beam-matching, ambiguity resolution, disambiguation framework

Introduction

Natural language (spoken or written) is seen as an attractive modality for interactive computer systems. Recent applications using a natural language interface include multi-modal drawing tools (Caelen 1994 ; Hiyoshi & Shimazu 1994 ; Nishimoto, *et al.* 1994), on-line information retrieval (Haddock 1992 ; Zue, *et al.* 1993 ; Goddeau, *et al.* 1994), oral control systems, and, finally, face to face translation systems (Morimoto, *et al.* 1992 ; Kay, *et al.* 1994).

As natural language is highly ambiguous even in restricted domains, interactive disambiguation is seen as a necessity to achieve more robust and user-friendly interactive systems, face to face translation, systems and Dialogue-Based Machine Translation systems (Boitet & Blanchon 1993). In those contexts, an interactive disambiguation module is to plan interactive sessions for the user to choose, among the solutions produced by an analizer, the one corresponding to the intended "meaning".

To build such disambiguation modules, we have defined a framework in which two disambiguation modules have been produced, one for French, at the GETA lab (France), in the context of the LIDIA project (Blanchon 1994 ; Boitet & Blanchon 1995) of dialogue-based machine translation, and one for English, at ATR-ITL (Japan), in the context of interpreting telecommunications (Blanchon & Loken-Kim 1994).

In this paper we will concentrate on the English disambiguation module. We will first give an overview of the methodology involved. We will, then, describe the disambiguation engine, which is the language-independent part of a disambiguation module. We will next describe the lingware which make up the English disambiguation module. We will finally give several examples of the produced disambiguation dialogues.

1. Overview

1.1. Framework

In the framework we propose the analyzer is to produce multiple analysis trees for every ambiguous sentence. An analysis is called a "solution" (fig. 1).

When an ambiguity is recognized, a question is prepared. A question is made of items among which the user will have to choose. Those items are constructed by rephrasing each solution. To rephrase the solutions we do not use any complex generation process but a collection of operators so as to ensure that unambiguous items are proposed to the user[1].

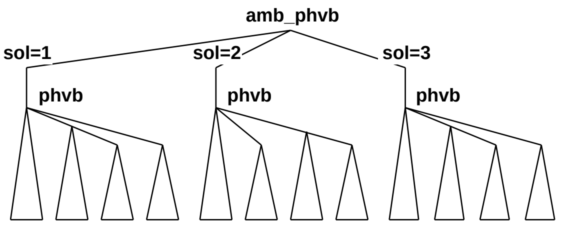


Figure 1: A multiple analysis made of 3 solutions

The most complex mechanism involved in the recognition of an ambiguity is the matching of a set made of several linked patterns (a beam) over the produced solutions. The patterns are said to be linked because they are sharing some variables that should have the same value when the matching process is over.

A disambiguation module is made of two parts, an *engine* that is language-independent and a *lingware* that is language and parser dependent. The *engine* combines a pattern matching module, a beam matching module, the set of basic operators, and a question presentation module. The *lingware* is made of beams, dialogue item production techniques and a disambiguation scheduler in charge of defining the order in which the beams are matched against the analysis structure.

1.2. Ambiguity Classification

We have defined a set of three meta-classes of ambiguities to be used and refined in each instance of a particular disambiguation module. Thoses meta-classes — lexical ambiguity, geometrical ambiguity, labelling ambiguity — are defined as follows.

– There is a lexical ambiguity when the analizer is unable to operate an unique segmentation into words or terms ([right here] *vs* [right] [here]), or unable to choose a word among homophones (to *vs* too *vs* two), or unable to choose a syntactic class among homographs (conduct noun *vs* verb).

– There is a geometrical ambiguity when the analyzer produces several solutions with different geometries without a lexical ambiguity.

– There is a labelling ambiguity when the analyser produces serveral solutions with the same geometry without a lexical ambiguity.

Here are some examples of the kind of ambiguities solved with the English disambiguation module:

Lexical Ambiguity
Noun-Adjective
*ex:This is an **English** speaking agent.*
Noun-Verb
*ex:You can travel by subway, or **taxi**.*
Phrasal-Verb
*ex: It is difficult to **get out** of Kyoto station.*
Geometrical Ambiguity
Prepositional Attachment
*ex:Where can I catch a taxi **from** Kyoto station?*
Adverbial Attachment
*ex:You can pay for it **right** on the bus.*
Conjunction
*ex:Can I ask you to type in your name **and** the telephone number?*
Labelling Ambiguity
*ex: I want a reservation **for** the hotel.*

1.3. Implementation

The current implementation is realized in Common Lisp Object System (CLOS) in the Macintosh Common Lisp environment. The only platform-specific module is the dialogue presentation module. Thus, most of the code is portable to any CLOS implemenation.

2. The engine

The disambiguation engine, which is language independent and is to be reused by each disambiguation module, consists of: a pattern matching mechanism (2.1.), a beam matching mechanism (2.2.), a presentation module (2.3.), and, a set of basic operators (2.4.).

2.1. The pattern matching mechanism

The patterns are described with a language derived from the one proposed in (Norvig 1992). A pattern (fig. 2) describes a family of trees, with constraints on their geometry and labelling.

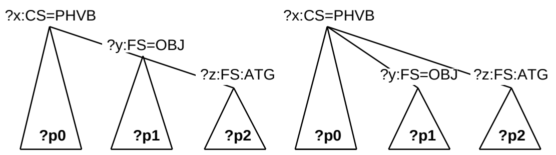


Figure 2: 2 patterns forming a beam

A pattern contains two kinds of variable: node variables (?x, ?y, ?z) describing constrains on nodes (CS=PHVB), and forest variables that can be sets of trees (?p0, ?p1, ?p2).

The pattern matching mechanism is also inspired by (Norvig 1992)'s proposal.

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] Indeed, it is impossible to ensure onself that the output produced by a generation proces — producing a textual output from a nore or less abstract representation — is not ambiguous.

2.2. The beam matching mechanism

A family of ambiguities can be defined with several sets of patterns also called beams (fig. 2 & 9).

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 pattern (**b**),

– for each solution $\mathbf{Sol_i}$ there is an unique pattern $\mathbf{P_j}$ that match that solution,

– each pattern $\mathbf{P_j}$ match at least one solution $\mathbf{Sol_i}$,

– the distance **d** between the bindings of each forest variables is null .

The distance between two bindings is null if and only if the coverage of each forest variable, except the last one, is the same in each binding. For the last variable of the patterns, if the coverage is not the same, one coverage has to be a prefix of the other.

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

In practice, the beam matching is realised by the method **match-beam** (fig. 3).

The inputs for this method are a pattern-beam and a numbered-solutions set. A numbered-solutions set is a list of couples: ((number solutions)$^{+}$). The disambiguation allows the user to select the number of the solution he meant.

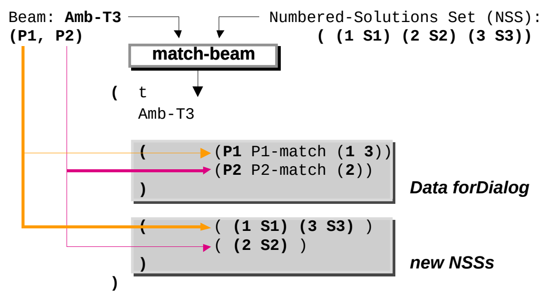


Figure 3: match-beam *input and output*

The output is a list of four data:

– t or nil, if the beam has matched of not. In the latter case the other data are irrelevant,

– the name of the matched beam,

– a list of triplets which will enable the engine to construct the dialogue items (cf. fig. 9) to be used to solve the ambiguity,

– a list of new numbered-solutions sets in which, if necessary, other ambiguity will be searched to produce other disambiguation questions (cf. fig. 5).

2.3. The presentation module

For a given disambiguation module, the disambiguation automaton produces a question tree.

A disambiguation question is made of:

– a question-language: language used (here English),

– a question-type: ambiguity to be solved,

– a question-modality: modality(ies) to be used to present the question to the user,

– an ambiguous-item: utterance to be disambiguated,

– a question-items-list: dialogue items to be proposed to the user.

The question tree is covered until no more question is to be asked. A method, ask-question — specialized on the question-language, the question-type, and the question-modality —, proposes the question to the user.

2.4. The operators

The operators are used to describe the construction dialogue items. They allow to perform several operations on the binding of the variables. Three families of operators are defined:

1) the operators of the first family describe some manipulations of subtree structures, basically the selection or the suppression of some part of the trees; for example:

`Text`	produces the text of the subtree given as parameter.
`Coord`	produces the coordinating occurrence of the subtree given as parameter.

2) the operators of the second family describe the replacement of a occurence by an other; as:

`Substitute` replaces an ambiguous preposition with a non-ambiguous one according to several properties.

3) the operators of the third family describe some more complex operations of distribution and bracketing of subtrees.

3. The lingware

The lingware is used to describe a particular instance of a disambiguation module for a given language and a given analyzer. The lingware consists of patterns and beams (3.1.), a disambiguation automaton (3.2.), dialogue item production methods (3.3.), and dialogue classes (3.4.).

3.1. The patterns & beams

A pattern is made of a pattern-name (the name of the pattern), a pattern-value (the definition of the pattern) and a pattern-method (the method to be applied to the binding in order to produce a dialogue item).

A beam is made of a beam-name (the name of the beam), and a beam-value which is a list of patterns composing the beam. The order in which the patterns appear in the list determines the order of the question's items; for a recognized beam, pattern$_i$ gives rise to the production of item$_i$ of the dialogue.

Beams describing the same kind of ambiguity are then grouped into a beam-stack. A beam-stack is associated with each one of the relevant ambiguity recognition states of the disambiguation automaton. When trying to recognize a class of ambiguity, each beam is matched one after another until a beam has been matched or until every beam has been tried without success.

3.2. The disambiguation automaton

The disambiguation automaton is made of three kinds of states: an automaton-scheduler, meta-class recognition states, and ambiguity-class recognition states.

The automaton-sheduler (fig. 4) is the entry point of the disambiguation scheduler of every disambiguation module. It is defined as a method with one parameter specialized on the_language.

The ambiguity class recognition states (fig. 4) are described by methods sharing a common skeleton:

– when an ambiguity is recognized, two kinds of data are produced: data to produce the dialogue aimed to solve the regognized ambiguity and, if necessary, new numbered-solution sets (**new NSSs**) used to produce the following questions if any ambiguity is still remaining.

– when an ambiguity is not recognized, the next ambiguity class recognition state is triggered.

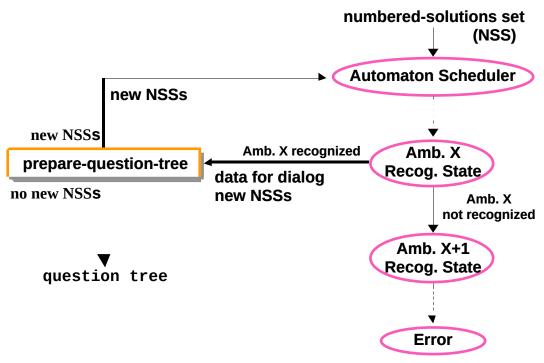


Figure 4: Preparation of a question tree

With the current method prepare-question-tree, the question tree is iteratively constructed and a question tree is prepared as shown fig. 5.

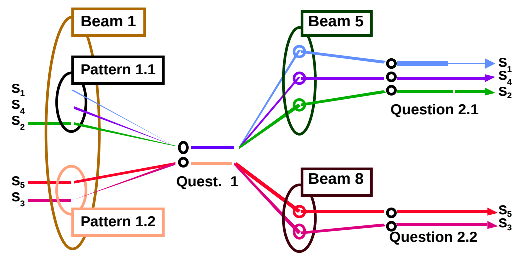


Figure 5: A question tree to discriminate 5 solutions

With the current strategy all the questions are first prepared before the first one is answered. An other strategy would prepare a question only when necessary. This would be made thanks to a specialisation of the method prepare-question-tree on a parameter called strategy (all-at-first, one-by-one).

3.3. The dialogue item production methods

The dialogue item production methods are described with the method **item-production-method** (fig. 6 & 7) specialized on its pattern-name argument.

Each method produces a string of characters which is an arrangement or a manipulation, with the operators defined in § 2.4, of the binding of some of the variables defined in the pattern the method is associated with.

The method associated with the left pattern in figure 2) produces the following string:

text(?p0) **(** text(?p1) text(?p2) **)**

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

Figure 6: The method item-production-method

The method associated with the right pattern in figure 2 produces the following string:

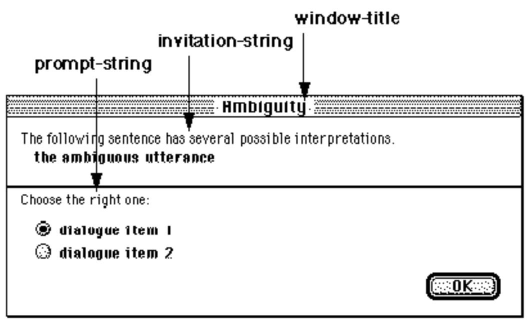
text(?p2)**,** text(?p0) text(?p1)

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

Figure 7: The method item-production-method

3.4. The dialogue classes

Generic dialogue classes are specialized with new dialogue classes according to the disambiguated language. This concerns, the fonts, styles and the meta-language used to ask a question. The following figure shows what has been done for English.



The following sentence has several possible interpretations.
the ambiguous utterance

Choose the right one:

● **dialogue item 1**
○ **dialogue item 2**

Figure 8: Some dialogues' slots

4. Some examples

In this last section, we will present some examples of the produced dialogues.

4.1. Syntactic class ambiguity

In the sentence "This is an English speaking agent." there is ambiguity for the syntactic class of the word "English" which can interpreted as a noun (an agent who speaks English), or as an adjective (an agent who is English).

This ambiguity is described as follows.

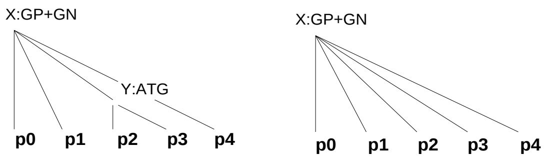
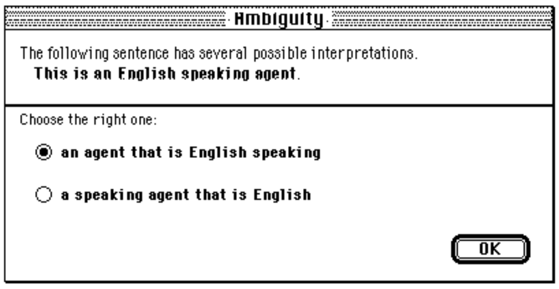


Figure 9: A beam describing an zmbiguity of syntactic class

The produced dialogue is shown below.



The following sentence has several possible interpretations.
This is an English speaking agent.

Choose the right one:

● **an agent that is English speaking**

○ **a speaking agent that is English**

Figure 10: The resolution of a syntactic class

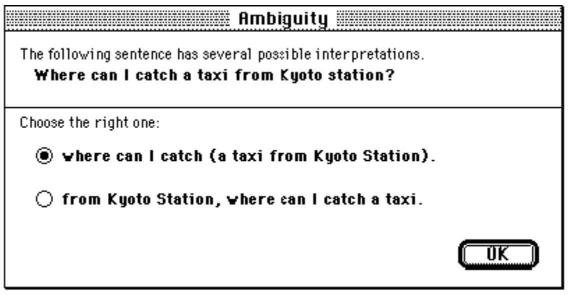
4.2. Prepositional attachment involving the verb

In the sentence "Where can I catch a taxi from Kyoto station?", there is an ambiguity called phvb-prep-att (prepositional attachment involving the verb).

The ambiguity is detected by the beam shown fig. 2. With the following binding:

– ?p0 = "where can I catch"
– ?p1 = "a taxi"
– ?p2 = "from Kyoto station"

The dialogue produced is shown below.



The following sentence has several possible interpretations.
Where can I catch a taxi from Kyoto station?

Choose the right one:

● **where can I catch (a taxi from Kyoto Station).**

○ **from Kyoto Station, where can I catch a taxi.**

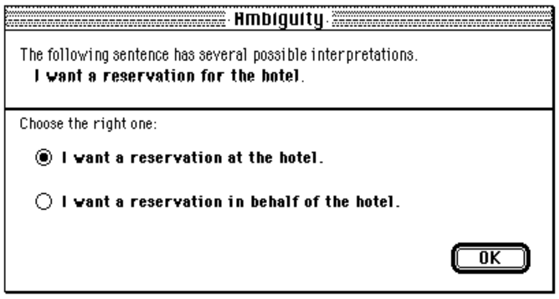
Figure 11: The resolution of a prepositional attachment ambiguity

The first dialogue item is produced with the method described in figure 6, and the second item with the method described in figure 7.

4.3. Labelling ambiguity

This kind of ambiguity is not recognized through the use of a beam matching mechanism, but rather by the recognition of property over lists.

For the following sentence, "I want a reservation for the hotel." The preposition "for" is ambiguous, it can be replaced by "on behalf of" (***I*** is making the reservation on behalf of the hotel) or by "at" (***I*** is making a reservation for someone to sleep at the hotel).



The following sentence has several possible interpretations.
I want a reservation for the hotel.

Choose the right one:

● **I want a reservation at the hotel.**

○ **I want a reservation in behalf of the hotel.**

Figure 12: The resolution of a labelling ambiguity

Conclusion

The methodology we described has in our opinion two main advantages: it can be customized and it can be improved incrementally. The ability to customize comes from the clear separation of the linguistic data from the kernel. In this framework, a number of different disambiguation modules can be produced for different languages and kinds of input. The description of the linguistic data can be improved incrementally as the design and the use of a disambiguation module progress.

The coverage of the first version of an interactive disambiguation module described here is currently being evaluated. For this evaluation we are constructing a new corpus from the data collected in the latest experiments conducted in the EMMI framework of interpreting multimedia /multimodal and telephone communications, (Park & Loken-Kim 1994) and (Park, *et al.* 1995).

Most of the ambiguities we have found in the evaluation and improvement corpus are already covered by the current module. The most important difference lies in the fact that there are a greater number of ambiguities of coordination. After we evaluate the first version of the disambiguation module by testing its coverage on the new data, we will improve the module by extending it to include the ambiguities it was not able to handle in the test data. Certainly, we will not be able to claim that the improved module will cover all the ambiguities found in spontaneous English, but it will have broad coverage for application to these particular domains.

Similarly, if new ambiguities are located in future data, the module can be incrementally improved to cope with the new ambiguities.

We are also currently investigating in two directions:

- The fisrt direction is the use of weights to let the module learn from the history of the dialogue. If one particular interpretation of an often recurring ambiguity is always chosen, the module will more heavily weight that interpretation, either automatically, or after querying the user. The module will then be tunable.
- The second direction is the automatic construction of new patterns. When an ambiguity is not recognized by the module, it should prepare the patterns to be used to recognize it. The module will the learn to recognize next ambiguities. Of course, the dialogue items production methods to be associated with the new pattern will have to be prepared by hand.

In future work, we intend to integrate the disambiguation module into the EMMI context to investigate its usability in a multimedia environment. There is also a need to design experiments with naive users to determine the optimal design of disambiguation dialogues and interactive sessions. Finally, we hope to use these results and continue to improve then back in the framework of DBMT and perhaps face to face translation in France at the GETA-CLIPS lab.

Acknowledgement

I would like to express my sincere gratitude to Dr. Yamazaki and ATR-ITL for having supported this work, Tsuyoshi Morimoto and Kyung-Ho Loken-Kim for their strong challenges to my proposals, and Laurel Fais for her kind cooperation and smart proposals. I would not forget Prof. Boitet and the GETA lab. for their constant support.

References

Blanchon H. (1994). *Perspectives of DBMT for monolingual authors on the basis of LIDIA-1, an implemented mock-up.* Proc. Coling-94. Kyoto, Japan. August 5-9, 1994, vol. **1/2** : pp. 115-119.

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.

Boitet C. & Blanchon H. (1993). *Dialogue-based MT for monolingual authors and the LIDIA project.* Proc. NLPRS'93. Fukuoka, Japon. December 6-7, 1993, vol. **1/1** : pp. 208-222.

Boitet C. & Blanchon H. (1995). *Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup.* in Machine Translation. vol. **9**(2) : pp. 99-132.

Caelen J. (1994). *Multimodal Human-Computer Interaction.* in Fundamentals of Speech Synthesis and Speech Recognition. Keller, E. (ed.). John Wiley & Sons. New York. pp. 339-373.

Goddeau D., Brill E., Glass J., Pao C., Philips M., Polifroni J., Seneff S. & Zue V. (1994). *GALAXY: a Human-Language Interface to On-Line Travel Information.* Proc. ICSLP 94. Yokohama, Japan. September 18-22, 1994, vol. **2/4** : pp. 707-710.

Haddock N. J. (1992). *Multimodal Database Query.* Proc. Coling-92. Nantes, France. 23-28 juillet 1992, vol. **4/4** : pp. 1274-1278.

Hiyoshi M. & Shimazu H. (1994). *Drawing Pictures with Natural Language and Direct Manipulation.* Proc. Coling-94. Kyoto, Japan. August 5-9, 1994, vol. **2/2** : pp. 722-726.

Kay M., Gawron J. M. & Norvig P. (1994). *Verbmobil: A Translation System for Face-to-Face Dialog. CSLI lecture note no 33.* Center for the Study of Language and Information, Stanford, CA. 235 p.

Morimoto T., Suzuki M., Takezawa T., Kikui G., Nagata M. & Tomokio M. (1992). *A Spoken Language Translation System: SL-TRANS2.* Proc. Coling-92. Nantes, France. 23-28 juillet 1992, vol. **3/4** : pp. 1048-1052.

Nishimoto T., Shida N., Kobayashi T. & Shirai K. (1994). *Multimodal Drawing Tool Using Speech, Mouse and Key-Board.* Proc. ICSLP 94. Yokohama, Japan. September 18-22, 1994, vol. **3/4** : pp. 1287-1290.

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

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-86. Dec., 1994. 161 p.

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-90. Feb., 1995. 123 p.

Zue V., Seneff S., Polifroni J. & Phillips M. (1993). *PEGASUS: a Spoken Dialogue Interface for On-Line Air Travel Planing.* Proc. ISSD-93 — New Directions in Human and Man-Machine Communication. International Conference Center, Waseda University, Tokyo, Japan. November 10-12, 1993, vol. **1/1** : pp. 157-160.