

# Rule Based Reasoning for Heuristic Dialogue with Small Screen Mobile Devices

Alessandro Mazzetti

Planasia s.a.s. – Piazza S. Maria Nascente, 3 – 20148 Milano - Italy  
[planasia@planasia.it](mailto:planasia@planasia.it) - [www.planasia.it](http://www.planasia.it)

**Abstract.** This paper addresses the problem of Intelligent User Interface, with particular emphasis on problems arising from limited screen devices. Some techniques borrowed from Artificial Intelligence are presented, focusing on taxonomy management for heuristic dialogue driving. The proposed representation formalism is Rule-Based Backward-Reasoning. Two architectures are described: server-side and client-side reasoning. Languages and development Tools are presented, with examples and references to real applications.

## Introduction

Mobile devices are more and more becoming pocket information systems; we might suppose that in a near future mobile devices will be used in the same way as today's desktop web browsers.

In this paper we will consider 2.5 generation mobile phones with particular emphasis on widely deployed devices, whose features are:

- Small screen size
- Limited keyboard
- Restricted pointing system (no mouse)
- Fair graphic capabilities
- Full computational skills (e.g. Java)
- Full data communication operativity (GPRS)
- Unexploited logic capabilities

## Screen size and pointing system may be crucial

Let us consider the home page of whatever web portal, for instance in the field of tourism. Several home pages are oriented to show as many items as possible, in order to let user have all information at a glance.

For example, a 50 items catalogue of hotels would be a pleasant list into a desktop web page (fig. 1), but will be a nightmare into a phone (fig. 2). The absence of a pointing system (mouse) introduces discomfort in the most simple operations: instead of simply moving a mouse, you need to scroll a list by means of several boring clicks.

The limitations imposed by screen size can be overcome by intelligently selecting items. This may be achieved by means of a "heuristic" dialogue with some questions presented in sequential order (fig. 3). Notice that the number of choice-items for each question is crucial.

This is a problem of data segmentation, also known as "data clustering".

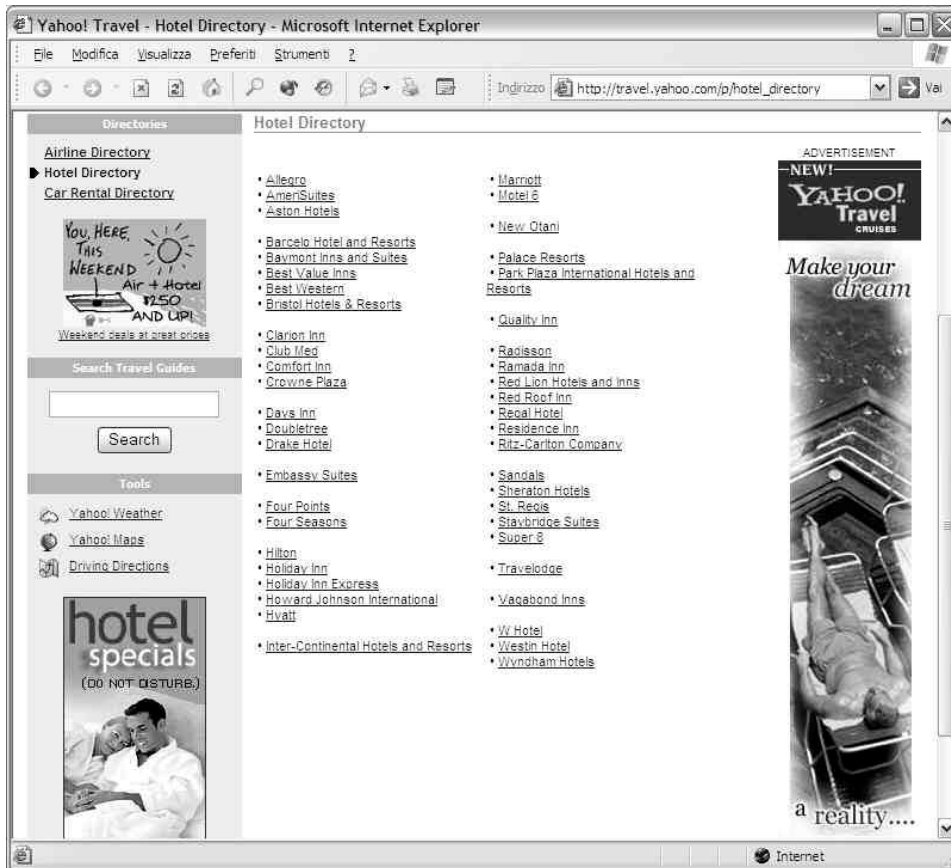


Fig. 1. Home page of a web portal: the motto is “all at a glance”



Fig. 2. A long list into a phone would require a nightmare-hardware.



Fig. 3. Few questions lead to selected items.

## Taxonomy and Heuristics

Data clustering is based on the hierarchical categorisation of items (fig. 4). The resulting structure is a taxonomy. The taxonomy should be built considering specific domain knowledge and this is the base for heuristic search, i.e. a quick path to the most plausible solution. A good taxonomy should guarantee that useless questions are never asked.

Let us consider for instance the case in which the user is looking for a downtown hotel: it does not makes sense to ask him about panoramic position or bathing facilities. Therefore each branch in the taxonomy tree should be followed only by pertinent paths.

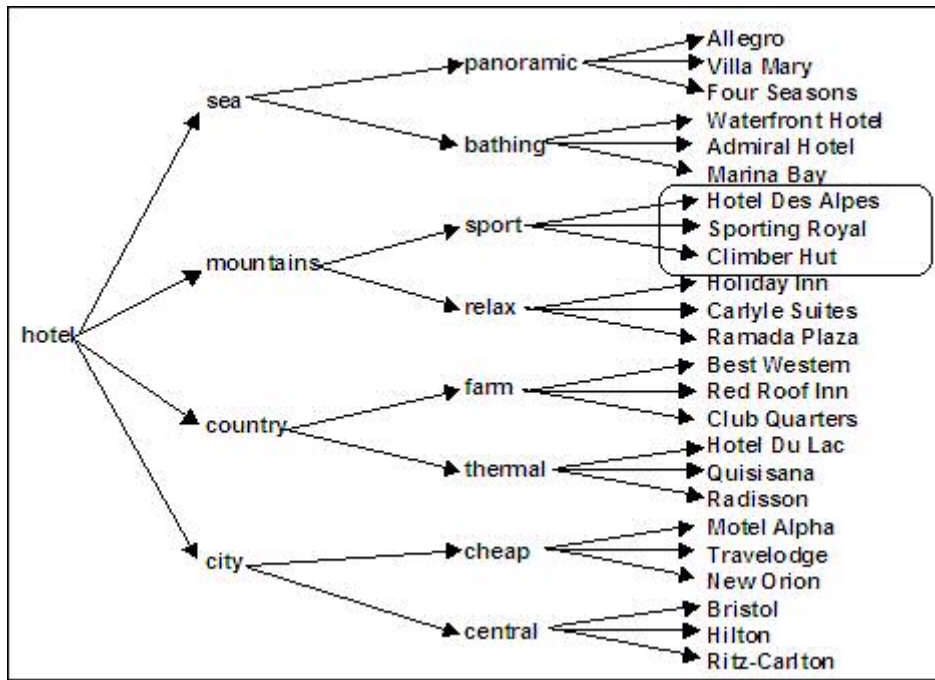


Fig. 4. A taxonomy for data clustering: each cluster contains few items

One important question is: how many taxonomy branches should I develop? It strongly depends on the domain knowledge, because some paths might be deeper than other. Roughly speaking, the number of levels in an equilibrated tree is

$$Q = \frac{\log(N)}{\log(P)}$$

where Q is the number of levels (i.e. the number of questions asked on the screen); N is the total number of items, P is the cluster cardinality (i.e. the number prompted choices). As an example, if we have a list of 100 items and we want to put on the screen questions with 4 choices, we need to ask 3.32 questions.

## Rule Based Backward Reasoning

One of the simplest formalism to represent knowledge for taxonomies is by means of Rules. An "If-Then" rule can easily represent a path from the taxonomy root to a leaf. As an example:

```

IF location = sea AND position = panoramic
    THEN hotel = "Allegro" OR "Villa Mary" OR "Four Seasons"

IF location = sea AND position = bathing
    THEN hotel = "Waterfront Hotel" OR "Admiral" OR "Marina Bay"

IF location = mountain AND activity = sport
    THEN hotel = "Hotel Des Alpes" OR "Sporting Royal" OR "Climber Hut"

...
  
```

A backward reasoning inference engine will start from the goal “hotel” and scan the rules with the aim of verifying conditions. Whenever a data is unknown (i.e. not derived by any rule in the knowledge base) it is asked. Considering the above example, the first question will be on the variable “location” (with options “sea”, “mountain”,...); if the user answers “sea”, the following question will be on “position”, otherwise it will be on “activity”.

In the above example there is no conceptual difference between a classification tree and a set of rules. However, rules are more expressive and powerful. Rules can go along with expert twisted mentality (achieved through non monotonic reasoning).

## Architectures

The typical architecture for a Backward-Reasoning Rule-Based System includes four modules: a knowledge base, an inference engine a user interface and a data access system (fig 5).

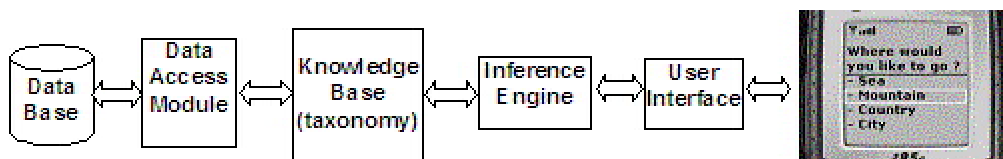


Fig. 5. Software architecture of a knowledge driven application

It is worthy underline the difference between Knowledge-base and Data-base. The knowledge-base contains mainly links between data, while the database contains full data. As an example, the knowledge base may contain hotel features (just the one useful for dialogue), while the database would contain full data with addresses, photos, prices, and all those information that are not implied in dialogue.

Two architectural possibilities exist: server-side reasoning (fig 6) and client-side reasoning (fig 7). The choice depends on languages and development tools.

In the server-side case, the application is configured as a Servlet. The inferential engine should support multi-task reasoning and the knowledge base may be updated frequently. The client device does not need any specific software; a generic Wap Browser would be sufficient. The communication between server and client is performed (and paid) for each question.

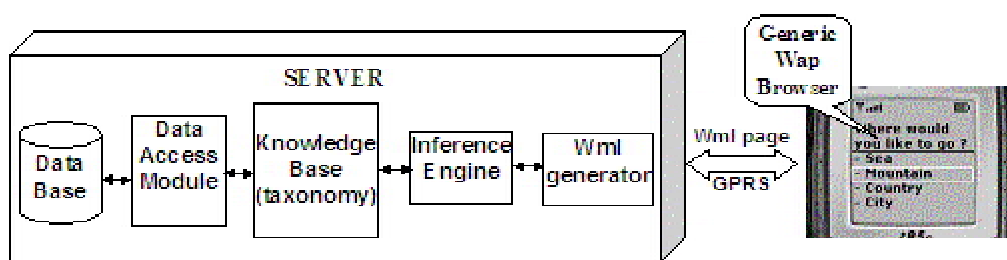


Fig. 6. Server side reasoning

The main advantages of server-side approach are:

- No limitations in knowledge-base size
- Easy maintenance of knowledge-base
- Security (knowledge-base is not deployed)

Drawbacks are:

- Expensive consultations
- Slowness (data transmission between questions)
- Connectivity required

In the client-side case the application is configured as a MIDlet and should be downloaded by OTA (Over The Air) provisioning, therefore knowledge base updates should be pulled by the user. The inferential engine should be particularly slim and knowledge base should be efficiently compiled. The communication between server and client is

reduced to the minimum necessary and performed only for getting specific data. The majority of dialogue could be carried on without any connection with server. Data transfer is reduced to the minimum necessary.

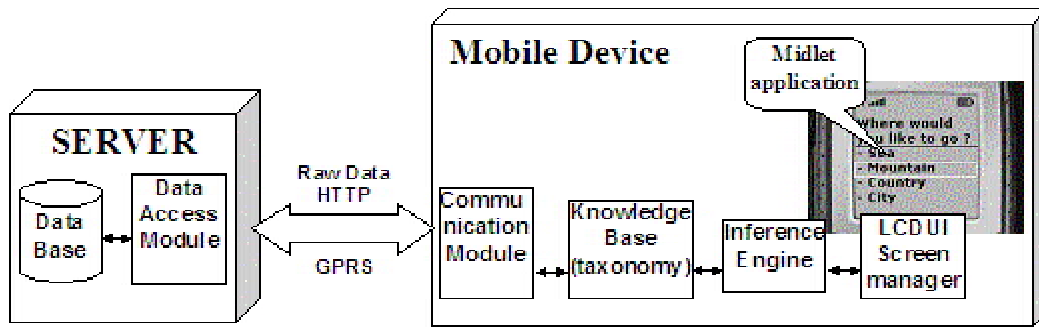


Fig. 7. Client side reasoning

The client-side approach is the most original issue of this paper. The widespread opinion about heaviness of inferential engines must be dispelled. Thin inference engines are easy to develop and some commercial products already exist. The main advantages of client-side approach are:

- Speed of dialogue (no connection needed between questions)
- Connectivity not required (or required only at end of consultation)
- Economic (download once, consult whenever you want)

Drawbacks are:

- Limited knowledge base size
- Difficulty in knowledge-base maintenance
- Disclosure of knowledge-base

It is desirable that big projects benefit of the possibility to implement both approaches, sharing the knowledge-base and the inference engine (just the GUI component must be different). This solution requires flexible tools and languages.

## Languages and Tools

The most suitable language for the development of such kind of application is Java. The Java technology supports both the server-side reasoning (J2EE) and client-side one (J2ME).

A powerful tool for the development of Knowledge Base and Inference Engine is Plexpert (<http://www.planasia.it/English/DemoPlexEng.html>). Plexpert is available both in Micro-edition (for client-side applications) and in Enterprise-edition (for server-side applications). Plexpert allows a remarkable compression of knowledge base (100 rules requires 4.7 Kbytes) and the inference engine requires 12 Kbytes.

Plexpert supports forward and backward rules chaining, with evocative capabilities (i.e. partially verified rules are considered at low priority) and heuristic search (i.e. data are evaluated only when necessary). Rule conditions can contain numeric expressions (greater-than, less-than,...). Reasoning may be non-monotone (i.e. achieved goals may be re-evaluated when premises change). Reasoning is based on predicate logic, with existential and universal quantifiers (i.e. IF\_EXISTS, IF\_ALL and FOR\_EACH). Data structure is based on hierarchical classes and multi-instances variables. One of the most valuable feature of Plexpert's Inferential Engine is the high integration between inferential and procedural processing: the knowledge base mixes Rules and Java-methods with a bi-directional interaction (a rule can call a method; a method can call a rule, launching a backward session).

Plexpert includes several predefined user-interfaces: Applet, Console, Servlet, Midlet, Emulated-Midlet, Graphical. This allows the development of a unique Knowledge Base and the deployment in several architectures.

In its Micro-edition, Plexpert includes tools for Wireless HTTP Connection (possibly through GPRS) and for Permanent Data Storage into mobile device (RMS – Record Management Storage).

## Applications and examples

A complete application, developed with Plexpert, is “Ricettexpert”, a cookery adviser that offers recipes and facilitates supermarket purchases. This application contains a wide taxonomy for menu composition and guarantees the consistency and adequacy of dishes. The knowledge-base contains specific heuristic about cookery subject: as an example, if the user has little time for cooking, an investigation is done about the possibility to prepare some dishes the day before. Reasoning flow aims at finding quickly a satisfactory solution from the user point of view.



**Fig. 8.** “Ricettexpert” is a cookery adviser that offers recipes and facilitates supermarket purchases. At the moment it is available in Italian language only.

Ricettexpert is a typical Client-side application: the Rule-base is totally stored into the mobile device, therefore the most of dialogue may be carried out without any connectivity. The connection is activated only for Database access, specifically for cooking recipes download. The application allows the storage of recipes into local phone memory. The most interesting aspect of this application is the possibility to send a purchase order directly to a supermarket, with the guarantee to obtain all necessary ingredients. This kind of automation can be very business prone.

A demo version of Ricettexpert is available on: <http://www.planasia.it/Prototipi/Ricette/ProvaRicette2.html>

## References

- [1] Diego Bisci, Massimo Gallanti, Alessandro Mazzetti, e-Maintenance and Business Intelligence for High Voltage Power Network Components, KES-2002, Int. Conference on Knowledge Engineering Systems, Crema (MI) Italy, sept. 2002
- [2] Alessandro Mazzetti, Luca Bonini, A Knowledge-Based Web Site For Agriculture, KES-2002, Int. Conference on Knowledge Engineering Systems, Crema (MI) Italy, sept. 2002
- [3] Planasia, Plexpert User Manual
- [4] Alessandro Mazzetti, Applicazioni dei Sistemi Esperti, Muzzio Publ 1987.
- [5] Hayes-Roth, Waterman, Lenat, Building Expert Systems. Addison Wesley Ed. 1983.
- [6] J.F.Glimore, Expert System Tool Evaluation. 6th Internat. Workshop on Expert System and their applications, Avignon, 1986.
- [7] Alessandro Mazzetti, Interpretazione Interattiva dei Dati in ambito Internet/Intranet. Workshop AI\*IA on Organizational Tools and Intelligent Access for Heterogeneous Information - Padova, Sept 1998.
- [8] Alessandro Mazzetti, Development Tools for Knowledge Based Systems. Giornata di studio ANIPLA su processi produttivi - Milano, Oct 1996.
- [9] Alessandro Mazzetti, Sistemi di ausilio all'evoluzione e manutenzione della conoscenza. 5th Symposium of Italian Association for AI (AI\*IA). Napoli, Italy, Sept 1996.