An Agent System Reasoning about the Web and the User

Giovambattista Ianni
Department of Mathematics
Università della Calabria
Via Pietro Bucci, 30B
87036 Rende (CS), Italy
ianni@mat.unical.it

Francesco Ricca
Department of Mathematics
Università della Calabria
Via Pietro Bucci, 30B
87036 Rende (CS), Italy
ricca@mat.unical.it

Francesco Calimeri
Department of Mathematics
Università della Calabria
Via Pietro Bucci, 30B
87036 Rende (CS), Italy
calimeri@mat.unical.it

Vincenzino Lio
Department of Mathematics
Università della Calabria
Via Pietro Bucci, 30B
87036 Rende (CS), Italy
lio@mat.unical.it

Stefania Galizia
Department of Mathematics
Università della Calabria
Via Pietro Bucci, 30B
87036 Rende (CS), Italy
galizia@mat.unical.it

ABSTRACT

The paper describes some innovations related to the ongoing work on the *GSA* prototype, an integrated information retrieval agent. In order to improve the original system effectiveness, we propose the GSA_2 system, introducing a new internal architecture based on a message-passing framework and on an ontology description formalism (WOLF, Web ontology Framework). GSA_2 is conceived in order to describe and easily perform reasoning on "facts about the web and the user". The most innovative aspect of the project is its customizable and flexible reasoning system, based on Answer Set Programming; it plays the role of the central decision making module, and allows the Agent to take proactive decisions. The introduction of a logic language allows one to describe, program and plan behaviors of the Agent easily and quickly, and to experiment with a large variety of Information Retrieval strategies. Both the System Architecture and WOLF are general and reusable, and the result constitutes a good example of real implementation of agents based on logics.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Algorithms

Keywords

Agents, Logic Programming, Answer Set Programming, Information Retrieval

Comments

This work has been partially funded by the EU projects WASP (IST-2001-37004) and INFOMIX (IST-2001-33570). A full version of this paper is available from `http://gsa.gibbi.com`.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

Introduction

The Global Search Agent, illustrated in [2, 1], is a prototype conceived in order to improve web search quality on both the recall and the precision sides, and to assist the user through user profiling techniques. It is, in a sense, a aggressive document collector. Many of the web retrieval techniques are exploited and combined in a single framework:

Meta-Searching: in order to enhance coverage, a lot of search engines are queried in parallel;

Anticipated exploration: linked documents are explored and filtered in advance, independently of user's intervention;

User profiling: the user is allowed to express implicit and explicit preference on selected documents;

Document ontology design: the user can classify his search in a hierarchical taxonomy. Such "concept tree" information is employed to tailor the anticipated exploration to a fine-tuned search space.

The version of *GSA* herein presented (GSA_2 or the Agent, in the following), keeps the spirit of the previous version, and aims at better integrating such techniques, introducing a new flexible, modular, easily reprogrammable architecture. The main aim of the new architecture is to allow the Agent designer to experiment, combine and evaluate a variety of information retrieval, meta-searching, peer-to-peer techniques in a quick, clear and declarative way. To this end, we have decided to:

1. split the internal structure of GSA_2 in a set of cooperating agents; each agent is a specialized module plugged within the system giving the Agent its own capabilities; new functionalities might be easily added introducing new kinds of agents;

2. introduce a new Peer to Peer module, in order to allow the Agent to exchange knowledge throughout a network of peer GSA_2 instances;

3. introduce a Web Ontology Framework (WOLF in the following), containing logic primitives intended to model the Web, the internal document ontology, the user(s) profile(s), the web information sources (like traditional search-engines), the document contents, and the interaction and knowledge exchange between peers;

4. equip the system with an internal (logic) reasoning engine (embedded inside the agent *KatheRinE*, which aims at reasoning on a knowledge base built upon WOLF primitives). This allows the Agent to take its own decisions about the way to retrieve in-

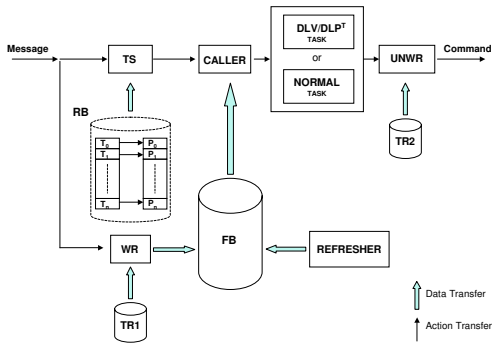

Figure 1: Architecture of the *KatheRinE* agent

formation, answer to explicit and implicit user requests, classify documents, and interact with peers.

The architecture of *GSA*$_2$ is conceived as a multi-agent system, where many (neither predefined nor fixed) types of agents interact through a message passing strategy. The platform is conceived in compliance with the FIPA specifications.

Each kind of agent has its own vocabulary for types of notifications and commands it can receive or send. The notification and commands dictionary has been conceived taking in account the FIPA message format; the final version of *GSA*$_2$ will be able to work using the JADE multi-agent platform, in order to exploit an already well-known and tested environment.

1. THE KATHERINE AGENT

KatheRinE (Figure 1) is the reasoning module of the *GSA*$_2$ system. The role played by *KatheRinE* is central, since it basically *implements* the behavior of the system.

This module acts as intelligent supervisor-colleague for all other internal agents. *KatheRinE* knows almost all about what happens inside of the Agent (i.e., which documents have been found, which documents have been scored, etc.) and outside of it (i.e., that the user has performed some activity, the P2P network has been queried, etc.), and *a)* chooses what has to be remembered, *b)* decides if some reaction has to be performed.

KatheRinE adopts the Answer Set Programming (ASP, in the following) as knowledge representation and reasoning framework.

1.1 The WOLF framework

The WOLF (Web Ontology Framework) consists of a complete set of predicates intended to describe the Web and the user in a logic model. It is designed in order to comprehend as much as possible anything the GSA programmer may want to deal with within the reasoning engine.Predicates are divided in two main categories: fluents and actions. Fluents models the Web and user status in the form of assertions coming from external agents, whereas actions are employed in order to indicate actions *GSA*$_2$ should perform, in response to a given event.

Fluents are intended in order to describe:

1. Knowledge exchange: trust links between agents, physical reliability and knowledge of each agent.

2. Document ontologies: topic taxonomies, relationships between concepts, documents and keywords.

3. Meta-search modelling: reliability and responsiveness of each search engine. Quality of the search engine with respect to any topic.

4. User profiling: explicit and implicit feedback from the user; documents visited, time user has spent on them; submitted queries, favourite search engines etc.

The main objects involved in the presented ontology are: peers (remote *GSA*$_2$ instances), queries (set of keywords), documents (annotated lists of words and hyperlinks), bunch of documents (set of documents of generic use), topics/concepts (nested categories of documents), words. It is assumed each object is associated with an unique object identifier.

For instance, WOLF includes predicates like `authority(p,o,c)`, stating that the authority (competence) of the peer p on the object o is given by the integer value c; or `up(p,t)`, which indicates the peer p was reported alive at time t. As for the web structure modeling, the framework includes primitives like `link(u1,u2)`, which is true if GSA knows that the document $u1$ links $u2$. The existence of a query q is modelled by the fact `query(q)`, and `concept(c)` means that there exists the category of documents c. `contains(c,d)` states that the topic d is contained in the topic c.

Most of the action predicates are in the form `askforobject(o)`, where o is some kind of object (e.g. a document), or in the form `askforrelationship(o,o')`.

1.2 Example of the Katherine lifecycle

Assume a message m, of the kind *QueryRequest* enters *KatheRinE*. This kind of message may come either from the user interface (whenever the user submits a query in a suitable text field), or from some *GSA*$_2$ peer around the *GSA*$_2$ network, asking for answers to a text query. Let m contain the query $q = \{$ "search","engine" $\}$. m is wrapped to a set of logical facts, like:

```
query(qx0000,t01,"local.louise").
belongs("search",qx0000,t01,1,"local.louise").
belongs("engine",qx0000,t01,2,"local.louise").
```

The above facts tell *KatheRinE* that q exists since time $t01$, and that this information has been asserted by the local agent *loUIse*. A logic program associated to the *QueryRequest* message type is then invoked. Such program describes which action to take in this case. For instance, we may want to answer by consulting search-engines, or taking advantage of the *GSA*$_2$ network. Such a program may look like the following:

```
1. t(Q,_) :- trigger(Q,_,0), query(Q).
2. suggest_object(U,"local.louise") :-
     t(Q,_), document_score(U,Q,S,_,_),
     scoreThreshold(Threshold), S > Threshold.
3. askforobject(Q,"local.meri") :-
     t(Q,"local.louise").
4. askforobject(Q,"local.pippi") :-
     t(Q,"local.louise").
5. askforobject(U2,"local.iris") :-
     t(Q,"local.louise"),
     suggest_object(U1,_), link(U1,U2,_),
     not document_score(U2,Q,_,_,_).
```

The purpose of the above program is to take advantage of any resource in order to answer the user request.

This way, it is possible to quickly program search strategies taking full advantage of the logic programming declarativity.

2. REFERENCES

[1] A. Castellucci, G. Ianni, D. Vasile, and S. Costa. Surfing and searching the web using a semi-adaptive meta-engine. In *International Conference on Information Technology: Coding and Computing (ITCC). Las Vegas, Nevada (USA)*, pages 416–420, 2001.

[2] G. Ianni. Intelligent anticipated exploration of web sites. *AI Communications*, 14(4):197–214, 2001.