This is the published version (version of record) of:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30009536

©2002 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright : 2002, IEEE
AN AGENT-BASED INTELLIGENT SYSTEM FOR INFORMATION GATHERING FROM WORLD WIDE WEB ENVIRONMENT

CHUN-SHENG LI (a), CHENG-QI ZHANG (a), ZI-LI ZHANG (b)

(a) Faculty of Information Technology, University of Technology, Sydney
PO Box 123, BROADWAY, NSW 2007, AUSTRALIA
(b) School of Computing and Mathematics, Deakin University, Geelong
Victoria 3217, AUSTRALIA
E-MAIL: (csl, chengqi) @it.uts.edu.au, zhang@deakin.edu.au

Abstract:
To use the vast amount of information efficiently and effectively from websites is very important for making informed decisions. There are, however, still many problems that need to be overcome in the information gathering research arena to enable the delivery of relevant information required by users. In this paper, an information gathering system is developed by means of multiple agents to solve those problems. We employed some ideas of Gaia methodology and Open Agent Architecture to analyze and design the system. The system consists of query preprocessing agent, information retrieval agent, information filtering agent, and information management agent. The filtering agent is trained with categorized documents and can provide users with the necessary information. The experimental results show that all agents in the system can work cooperatively to retrieve relevant information from the World Wide Web environment.

Keywords:
Software agent; Information retrieval; Information filtering; Categorized training; WWW

1 Introduction

The astonishing growth of the World Wide Web (WWW) is providing vast amount of information. To use the vast amount of information efficiently and effectively from websites is very important for making informed decisions. There are, however, still many problems that need to be overcome in the information gathering research arena to enable the delivery of relevant information required by users [4]. Firstly, it is difficult for us to gather right information from the World Wide Web because the information is vast, unorganized, multi-modal, and distributed on collections all over the world. Secondly, the number and variety of collections and services are constantly changing. Thirdly, information is uncertain (ambiguous and possibly erroneous). Fourthly, the search engine will provide users with overload documents when they are employed with certain keywords or expressions. An information gathering system is developed by means of multiple agents to solve the problems in this paper.

Intelligent agents get commitments from their users and act on the users’ behalf to gather the required information.

For implementing the system with software agent architecture, we employed some ideas of Gaia methodology [5] and Open Agent Architecture (OAA) [6] to contribute a framework. The framework consists of three levels, namely, role model level, agent type level, and agent instance level. In role model level, we describe a model with four roles. In agent type level, four agent types are designed. In agent instance level, the agents based on categorized training are implemented.

The remaining sections of this paper are organized as follows. In Section 2, the related work about this research is cursorily described. Section 3 presents methodology of constructing agent-oriented systems and the model with four roles for information gathering from the WWW. Section 4 discusses four agent types for designing the model and gives the complete framework. Section 5 describes the training and filtering algorithms based on categorized expression weights. At the same time, the architecture of each key agent for filtering is introduced. Section 6 concludes the paper.

2 Review of related work

The information retrieval (IR) and information filtering (IF) are key stages of the information gathering. IR deals with the representation, storage, organization of, and access to information items. IF is to filter out the irrelevant documents [4]. There are three classic models, namely, Boolean, vector, and probabilistic, in IR [8].

Much of the foundational work on agent technology has focused on methodologies to build agent-oriented systems, inter-agent communication protocols, and agent-based application systems [9]. Gaia is a methodology, which was specifically tailored to the analysis and design of agent-based systems [10]. But Gaia, as it presently stands, is not a general methodology for all kinds of multi-agent systems. It is not suitable for modeling open systems and controlling the behavior of self-interested agents [11]. The OAA made it possible for software services to be provided through the
cooperative efforts of distributed collections of autonomous agents. Communication and cooperation between agents were brokered by one or more facilitators, which were responsible for matching requests, from users and agents, with descriptions of capabilities of other agents.

Many agent-based software systems evolved some notion of our framework have been developed. Metacrawler is an agent that operates at a higher abstraction level by utilizing eight existing WWW index and search engines. An integrated environment acting as a software agent for discovering correlative attributes of data objects from multiple heterogeneous resources was presented in [1]. AgentRAIDER [8] was a comprehensive architecture for an intelligent information retrieval system with distributed heterogeneous data sources. It focused on specific aspects of the distributed heterogeneous problem such as database queries or information filtering. KQML is based on speech act theory and allows cognitive agents to cooperate. Although such an endeavor is certainly of interest, the KQML project display loopholes.

3 Methodology

The information gathering system consists of four components, namely, query preprocessing, information retrieval, information filtering, and information management, according to the goal of the system. Each component makes use of significant computational resources and is of independent feature. The organization structure of the system is static because inter-agent relationships do not change at run-time. So it is suitable for applying Gaia to analyze the system.

3.1 The analysis with Gaia methodology

Gaia is intended to allow us to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. In applying Gaia, we move from abstract to increasingly concrete concepts. The main Gaian concepts can be divided into two categories: abstract and concrete. Abstract entities are those used during analysis to conceptualize the system, but which do not necessarily have any direct realization within the system. Concrete entities, in contrast, are used within the design process, and will typically have direct counterparts in the run-time system.

The objective of the analysis stage is to develop an understanding of the system and its structure. In this stage, the system consists of roles, which role is defined by four attributes: responsibilities, permissions, activities, and protocols. Responsibilities determine functionality and, as such, are perhaps the key attribute associated with a role. Responsibilities are divided into two types: liveness properties and safety properties. Liveness properties describe those states of affairs that an agent must bring about, given certain environmental conditions. In contrast, safety properties are invariants. An acceptable state of affairs is maintained across all states of execution. The atomic components of a liveness expression are either activities or protocols. An activity is somewhat like a method in object-oriented terms, or a procedure in a PASCAL-like language. It corresponds to a unit of action that the agent may perform, which does not involve interaction with any other agent. Protocols, on the other hand, are activities that do require interaction with other agents. Permissions are the "rights" associated with a role. The permissions of a role thus identify the resources that are available to that role in order to realize its responsibilities. The activities of a role are computations associated with the role that may be carried out by the agent without interacting with other agents. Protocols define the way that it can interact with other roles. The roles model identifies the key roles in the system. A role will have associated with it certain permissions, relating to the type and the amount of resources that can be exploited when carrying out the role.

In information gathering system, one-to-one correspondence between the components and roles is adopted. The role model comprises four roles: namely, query preprocessing (QP), information retrieval (IR), information filtering (IF), and information management (IM).

QP role involves inputting user's query statement and training documents, getting two kinds of information retrieval expressions (one for browsing Webs, another for information filtering) by analyzing user's query statement and training documents, displaying the results to user, and operating information in file server by means of interaction with IM role. Expression here consists of some sequential words with logic operators. The protocols and activities are as follows:


Responsibilities:
QP=(A1)'(A2)'(A3)'(A4)
A1=(InputQuery)'(GetBrowsingExpressions)'InteractWithIR
A2=(InputQuery)'(GetTrainingExpressions)'(A5)'InteractWithIF
A3=(InputTrainingDocument)'(InteractWithIM
A4=(InteractWithIF)'(DisplayResults)'A5=(InteractWithIM)'(GetTrainingExpressions)'(A5)'InteractWithIF

Here: '.' means follow by; '.' means 'OR'; 'x' means x occurs 0 or more times; 'x' means x occurs 1 or more times; 'x' means x occurs infinitely. We write activity names in a sans serif font, and use a similar font, underlined, for protocol names. The following descriptions about role submit to this comment.

IR role involves browsing the relevant documents by employing search engines with browsing keywords or expressions, and saving the results to file server by means of interaction with IM role. Because the results may include
some irrelevant documents, we call the results as rough documents. The protocols and activities are as follows:

\[
\text{InputBrowsingExpressions, SelectSearchEngine, InteractWithSearchEngine, InteractWithIF, InteractWithIM.}
\]

Responsibilities:
IR= (B1) (B2)
B1 = (InputBrowsingExpressions),
(SelectSearchEngine, InteractWithSearchEngine)
B2 = (InteractWithSearchEngine),
InteractWithIM, InteractWithIF.

The first task of IF role is to train the expression weights based on information categories. The information categories are defined by users according to their information gathering purposes. A description, some relevant supporting documents, and some irrelevant supporting documents about one category are given for training the expression weights. The second task of IF role is to filter the rough documents. The category or categories of user's query must be determined by means of analyzing the query statement. And then the rough documents are filtered based on the expression weights of the category or categories. The protocols and activities are as follows:

\[
\text{InteractWithOP, InteractWithIM, InteractWithIR, TrainExpressionWeights, DecideQueryCategory, FilterDocuments.}
\]

Responsibilities:
IF= (C1) (C2)
C1 = (InteractWithOP), (InteractWithIM, TrainExpressionWeights),
C2 = (InteractWithOP), DecideQueryCategory,
(InteractWithIF),
(InteractWithIM, FilterDocuments).

InteractWithOP
IM role involves operating data files that are stored in file server. The protocols and activities are as follows:

\[
\text{InteractWithOP, InteractWithIR, InteractWithIF, OperateFileServer.}
\]

Responsibilities:
IM= (D1) (D2) (D3)
D1 = (InteractWithOP), OperateFileServer.
InteractWithOP
D2 = (InteractWithIR), OperateFileServer
D3 = (InteractWithIF), OperateFileServer.
InteractWithIF

3.2 Structure in the role model level

There are inevitably dependencies and relationships between the four roles in information gathering system. Indeed, such interplay is central to the way in which the system functions. The relationships between the roles are shown in Figure 1. The links between roles are presented in the interaction model (I Model). In fact, 'I Model' consists of a set of protocol definitions, one for each type of interrole interaction. InputBrowsingExpressions comprise 'I Model', but InputQuery, InputTrainingDocument, and DisplayResults interact with user. The protocols between agent and user consist of the interactive interfaces and disciplines for user to use the system. 'OperateFileServer' includes creating, reading, updating, or deleting information files stored in file server (FS). InteractWithSearchEngine informs selected search engines (SE1—SEm) to browse the rough documents according to the browsing keywords or expressions, and saves those documents to file server by means of interaction with IM role.

4 Framework for information gathering

The Gaia design process involves generating three models, namely, agent model, service model, and acquaintance model. The agent model identifies the agent types that will make up the system, and the agent instances that will be instantiated from these types. The service model identifies the main services that are required to realize the agent role. Finally, the acquaintance model documents the lines of communication between the different agents [7]. According to the analysis about the information gathering system in Section 3, four agent types are suitable for expressing the four roles. They are QP agent type, IR agent type, IF agent type, and IM agent type as shown in Figure 2.
In the framework, IM agent type wraps the information file server as an agent-based platform. From OAA point of view, IM agent type and the file server comprise facilitator agent. The facilitator is a specialized server agent that is responsible for coordinating agent communications and cooperative problem solving. In our system, the facilitator is used to provide a global data manipulation for its client agents, which allows them to adopt a blackboard style of interaction. The interactions between the agent types follow the rules defined in acquaintance model. Acquaintance model includes all communication function of InteractWithQP, InteractWithIR, InteractWithIF, InteractWithIM, and InputBrowsingExpressions. Acquaintance model is the concrete implementation of 'I model'. We employed Knowledge Query and Manipulation Language (KQML) for implementation of this model.

5 Architecture of the filtering agents

All agent types are one-to-one correspondence between roles and agent types in our system. So far each agent type only includes one agent instance. Query preprocessing (QP) agent and information filtering (IF) agent complete the filter of rough documents.

5.1 Architecture of the QP agent

The QP agent consists of Interface Module (INTERFACE), Input Module, Preprocessing Module, Display Module, Library Access Module, KQML Module, Temple File Storage (TF), and Control (C) Module. The architecture is shown in Figure 3.

![Figure 3. Architecture of the QP agent](image)

Fig.3. Architecture of the QP agent

Because QP agent must interact with users, for example, inputting query statement by keyboard, inputting training documents by transmission, controlling to execute specialized function, or displaying results or states of the system, Interface Module deals with interaction between user and the agent to achieve above tasks. Input Module deals with inputs of user's query statement, category description, and training documents (relevant or irrelevant). Preprocessing Module completes two tasks. The first task is to process category description and training documents stored in TF. Training documents, which are categorized, are analyzed one by one. The goal is to draw out the key relevant expressions and irrelevant expressions based on basic expression library (BE Library) and common word library (CW Library). The basic expression library manages all expressions or keywords the agent uses. The common word library manages the words that cannot be used as a keyword or an expression, such as, 'the', 'is', 'that', 'I', and so on. The second task is to process user's query statement and rough documents stored in TF for filtering rough documents. User's query statement, which is described by natural language, and rough documents are analyzed. The goal is to draw out the relevant expressions and irrelevant expressions based on relevant expression library (RE Library) and irrelevant expression library (NE Library). Library Access Module deals with the operations of all those libraries. Display Module deals with showing the results to user. KQML Module deals with the interactions with other agents. Control Module, which is the control center of the agent, makes other modules work in order.

5.2 Algorithms for information filtering

The model of information filtering for each category consists of two stages, namely, expression weight training and rough document filtering.

In training stage, we applied the vector model based on clustering techniques to train the expression weights. For determining the weight of each expression, calculate two factors, namely, intra-clustering similarity and inter-clustering dissimilarity. Term frequency (tf) is used for intra-clustering similarity and inverse document frequency (idf) is used for inter-clustering dissimilarity. Each expression weight is given by:

\[ W_{ij} = tf_{ij} \times idf_{ij} \]

Where \( i \) indicates th expression and \( j \) indicates th category.

Step 1: Calculate the expression weights of category description. Each expression weight of category description is given by

\[ W_{ij} = 10.5 + (0.5 \times freq_{ij})/ \max_{k} freq_{kj} \times \text{para} \times \log(NRDN_j / n_{ij})/ \log(NRDN_j) \]

Where \( freq_{ij} \) is the frequency of expression \( i \) appeared in category description \( j \); \( \max_{k} freq_{kj} \) is the maximal frequency among all expressions appeared in category description \( j \); \( NRDN_j \) is the total number of all irrelevant documents of category \( j \); \( n_{ij} \) is the frequency of expression \( i \) appeared in all irrelevant documents of category \( j \); \( \text{para} \) is a parameter (0<para<=1.0) for adjusting the weightiness of category description. The parameter can be trained for selecting a good set of expressions in Step 3.

Step 2: Calculate the expression weights of the relevant documents of category \( j \). Each expression weight of relevant document is given by

\[ W_{ij} = tf_{ij} \times idf_{ij} \]

\[ tf_{ij} = (1/RDN_j) \sum_{k=1}^{n_{ij}} (freq_{ij}) \]

\[ idf_{ij} = \log(NRDN_j / n_{ij}) / \log(NRDN_j) \]
Where $RDN_j$ is the total number of all relevant documents of category $j$; $freq_{kj}$ is the frequency of expression $i$ appeared in relevant document $k$ of category $j$; $totalfreq_{kj}$ is the total frequency of all expressions appeared in relevant document $k$ of category $j$; $NRDN_j$ is the total number of all irrelevant documents of category $j$; $mj$ is the frequency of expression $i$, appeared in all irrelevant documents of category $j$.

Step 3: Sort all expressions by weight value and save the first 120 expressions' with their weights to the relevant expression (RE) library as trained expressions of category $j$.

Step 4: Calculate the expression weights of all irrelevant documents of category $j$. Each weight is given by

$$W_{ij} = (1/NRDN_j) \sum_{k=1}^{NRDN_j} (freq_{kj}/ totalfreq_{kj})$$

Where $num_{ij}$ is the count value of all irrelevant documents in which expression $i$ appeared; $freq_{kj}$ is the frequency of expression $i$, appeared in irrelevant document $k$ of category $j$; $totalfreq_{kj}$ is the total frequency of all expressions appeared in irrelevant document $k$ of category $j$.

Step 5: Sort all irrelevant expressions by weight value and save the first 20 expressions, which cannot be included in the 120 expressions of category $j$ in the RE library, as irrelevant expressions of category $j$.

Step 6: Repeat Step 1 to Step 5 until all categories are trained.

In filtering stage, two thresholds are given to select relevant documents from rough documents. $RT$ is the threshold of relevant factor. $NRT$ is the threshold of irrelevant factor. $Tf_{ij}$ is the intra-clustering similarity that document $i$ belongs to category $j$, $Idf_{ij}$ is inter-clustering dissimilarity that document $i$ belongs to category $j$.

Step 1: Determine the category $j$ user's query belongs to. Calculate $Tf_{ij}$ by

$$Tf_{ij} = \max\{Tf_{1j}, Tf_{2j}, ..., Tf_{mj}\}$$

where $Tf_{ij} = (1/QN) \sum_{k=1}^{QN} (W_{kj}*counter_k)$

$QN = \sum_{k=1}^{m} (counter_k)$

Where $QN$ is the number of all expressions appeared in user's query and included in the relevant 120 expressions of category $j$ in the RE library; $counter_k$ is the frequency of expression $i$ appeared in user's query; $m$ is the total number of all categories.

Step 2: Calculate $Tf_{ij}$ and $Idf_{ij}$.

$$Tf_{ij} = (1/EN_j) \sum_{k=1}^{EN_j} (W_{kj}*counter_k)$$

$$EN_j = \sum_{k=1}^{m} (counter_k)$$

Where $EN_j$ is the number of all expressions appeared in rough document $i$ and included in the relevant 120 expressions of category $j$ in the RE library; $counter_k$ is the frequency of expression $i$ appeared in rough document $i$.

The $Idf_{ij}$ uses the same formula with $Tf_{ij}$ but $W_{kj}$ is from the NE library.

Step 3: If $Tf_{ij} > RT$ and $Idf_{ij} < NRT$, document $i$ belongs to category $j$.

Step 4: Repeat Step 2 to Step 3 for all rough documents.

Fig.4. Architecture of the IF agent

The main task of IF agent is to implement above algorithms. IF agent consists of KQML Module, Library Access (L.A.) Module, Training Module, Filtering Module, Control Module, Common Words (CW) Library, Basic Expressions (BE) Library, Relevant Expressions (RE) Library, and Irrelevant Expressions (NE) Library. The architecture is shown in Figure 4. Training model is the complementation of training algorithm and filtering model is the complementation of filtering algorithm.

6. Conclusions

A prototype of the proposed system is implemented using C and Socket technique. We are interesting in locating some information about 'what information is available on petroleum exploration in the South Atlantic' that is stored on 4 websites, which contain more than 500 documents on various aspects of petroleum. Certainly, the system has been trained by categorized petroleum information before the query. The precision and recall rate of the filtering algorithm are about 97% and 91%, respectively. However, the results very depend on the training documents. The system shows that information gathering from the WWW can be implemented by means of agents.

References


