An Architecture for Natural Language Dialog Applications in Data Exploration and Presentation Domain

Algridas Laukaitis, Olegas Vasilecas, Raimondas Berniunas,

and Eimontas Augilius

Vilnius Gedimino Technical University
algridas@isl.vtu.lt

Abstract. In this paper we present architecture and its implementation for the natural language dialog (NLD) applications in data exploration and presentation domain. Presented architecture can be integrated as a part of corporate information delivery web portal to bring new modalities for user interfaces. The architecture is based on software agent's paradigm and supports mobile as well as stationary agents. On this NLD architecture we implemented open source project for data exploration when the system user explores corporate data in the terms of machine human dialog. The following well know toolboxes has been integrated in this project: GATE - general architecture for natural language processing, IBM natural language understanding (NLU) toolbox, JOONE - neural network toolbox, Aglets - mobile agents framework.

1 Introduction

Data environments are becoming more and more complex as the amount of information a company manages continues to grow. Information delivery web portals have emerged as the preferred way to bring together information resources. Using information delivery web portal, your organization's employees, customers, suppliers, business partners, and other interested parties can have a customized, integrated, personalized, and secure view of all information with which they need to interact. But one of the big challenges remains for organization: it is how to teach employees or customers to use and understand complex database environment without involving experts and IT resources which are costly and time consuming. One of the solutions it to use natural language database interfaces.

From the early 80's and 90's there was many efforts involved in the research of natural language use for information extraction from data base management systems (DBMS). Natural language database interfaces (NLDBIS) are systems that allow users to access information stored in a database by formulating requests in natural language. For example a NLDBI would typically be able to answer questions like the following.

"Show me the latest prices of IBM shares"

The system that supports (NLDBIS) functionality automatically would translate user sentences to adequate SQL script, query some DBMS and return results to the user. NLDBIS have received particular attention within the natural language processing community (see [2] for reviews of the field), and they constitute one of the first areas of natural language technology that have given rise to commercial applications. Some successes have been achieved and some commercial applications emerged but the NLP techniques have not become a popular approach for DBMS interfaces. As was mentioned by researchers in [2, 3, 7, 8, 22, 28, 33] this is due to:

1. Graphical and menu driven interfaces achieved the level of sophistication that many data analyst can do analysis without deep knowledge of some data queering language (e.g. SQL). On the other side NLP techniques has not been able to deliver interfaces of adequate sophistication.
2. Most research and achieved results reports on the possibility to generate only one data queering script (in most cases this was one SQL sentence) generated from one natural language sentence. They do not support complex dialog, which is the most usual case in real life when we want interactively to build adequate request.
3. Most systems are commercial products [2] and because they are close systems there is difficulties in extending such systems. And we think that only open source projects can bring more attention from researchers to NLDBIS field.

In most available systems only system administrator are able to parameterizes the system. There are no available systems in which learning process will be integrated in user's daily work life. We think that resent advances is building personal assistants in such fields like an adaptive information research from internet [5] or personalized learning knowledge maps [24] will renew researches interest in (NLDBIS) field.

Our approach in this paper was the use of dialog instead of one sentence and on the other hand we do not look at NLP techniques as the one exclusive solution to query databases, instead we look at it as supplementary technique and as a part of multi-modal interface. To tackle mentioned problems we propose our system JminingDialog, which is constituent part of our open source information delivery web portal JMining. We have no intention to describe JMining architecture and information delivery portals in details and for details about it implementation as open source project we refer to [15] and [16].

Instead, we describe architecture of natural language dialog and natural language understanding (NLU) modules for building small web databases queering applications using only natural language. For those modules we propose an architecture that is based on stationary and mobile intelligent agents. Stationary agents are used when amount needed to support communication between distributed agents are small and interchange of short messages is enough. The pluses are that message passing provides platform- and language-independence as well as separation of transport and content information. The use of mobile agents in architecture is reasoned by the approach, which argues that knowledge consists largely of a personal, stored locally data files. Mobile agents can travel to various hosts where local knowledge is stored and gather necessary information that meets user request.

The paradigm of agents is a very promising approach to overcome some of the problems connected with heterogeneity on the side of the data sources as well as on the side of the users. As agents should operate autonomously and can be loosely coupled, they are well suited for the integration of distributed heterogeneous data sources, building unifying wrappers around them. This becomes especially beneficial, if agents can learn to extract information from an information source automatically (see for example [10] and [25]). On the side of the users, the paradigm of personal information agents offers a way to encapsulate the interests, the knowledge as well as the preferences of individual users. Personal agents can take the role of mediators between users and information sources, as well as between users among each other (see also [10] and [30]). Furthermore we present an agent architecture consisting of a set of asynchronously operating agents. This architecture enables us to perform sophisticated data and interaction analysis, without loosing the property of short respond times essential for interactive work in real-time. Based on the paradigm of mobile agents, we present a model for expressing knowledge that has been acquired continuously by individuals and groups of users and for using this as a means for semantic identification of various elements to build necessary web applications.

In our architectural implementation we used several toolboxes that are well established between academic and industry institutions. For natural language understanding we used IBM NLU toolbox [11, 12] as an example of an agent, which represents some kind of black box, i.e. we give input for an agent and get the answer without knowing algorithms and other implementation methods. As supplement to the NLU agent based on IBM NLU toolbox we build second type of NLU agents that are based completely on open source projects: general natural language architecture GATE [6] and JOONE neural networks toolbox [14]. Both technologies combined implements hybrid neural network NLU agents.

The contribution of this paper is threefold: Firstly, we introduce architecture of NL dialog for information delivery web portals. Proposed architecture is characterized by its flexibility to extend and a possibility to build complex information delivery web portals communicating with machine using natural human language. Secondly, we investigate two types of agents for distributed NL dialog systems: stationary agents that communicate by sending messages and mobile agents that move their code and data to remote hosts and locally solves adequate tasks and returns to their master host with the solutions. Thirdly, all presented concepts are implemented as Java open source project. We present discussion about open source projects and importance for support for such projects from academic environment. Our research shows that until now there was no open source project in natural language interfaces with information delivery portals and we think that our project can fulfill such gap.

2 General architecture

The architecture supports coordinated distribution of natural language dialog management and understanding agents and their integration with information delivery web portal components. Figure 1 shows basic components of this architecture. Below follows description of those components and their interconnection.

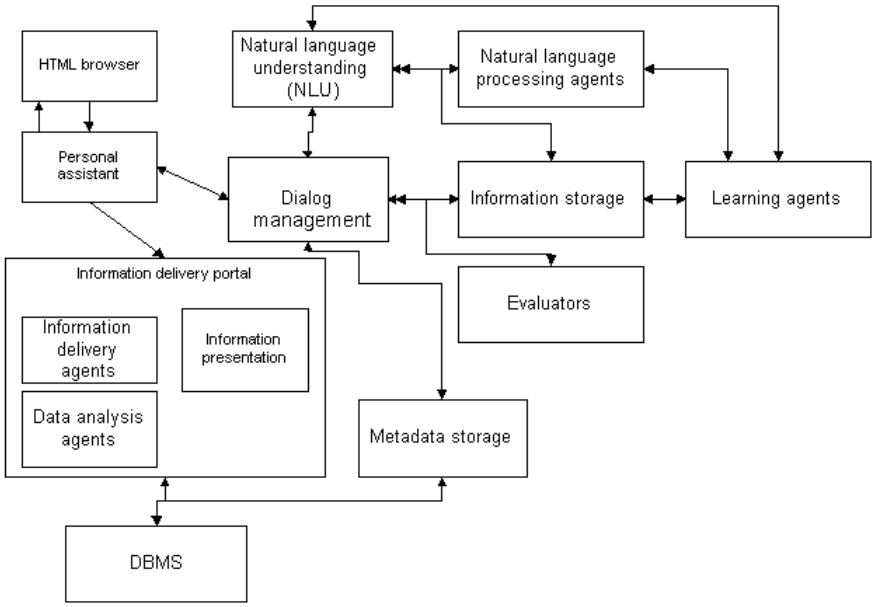

Fig. 1. General architecture of integration between information delivery web portal and natural language agents.

Personal assistant - it is an agent that hides all infrastructures behind information delivery portal and it's NLP components and uses multi-modal interface to communicate with the user. At presents architectural implementation it is possible to use HTML input forms with active hyperlinks and in addition forms with standard natural language dialog interface. At its present implementation personal assistant is far away from passing Turing test but we see it evolution in the future as becoming more intelligent and with ability to communicate with the user in more like human-expert way. Currently the most research in personal assistants has been done to help users search and gather information from unstructured data sources [5, 24] i.e. Internet, papers collections etc. We think that in the future personal assistant will integrate possibility to research structured data sources (e.g. databases) with unstructured data sources like Internet. In current personal assistant edition there are no speech to text converter (keyboard and mouse are only available options) or speech synthesizer but we think that such modalities are very important to imitate truly intelligent behavior and we will consider them in future implementations.

Dialog management – represents two sets of agents: state space dialog management agents and form based dialog management agents. The *state space* dialogue strategy is a mapping from a set of states (which summarize the entire dialogue) to a set of actions (such as identification of tables and database queries). The *state space* is defined by the collection of all variables that characterize the state of the dialogue system at a certain point in time. To avoid combinatorial explosion the designer of the

system must consider how on the one hand to limit the number of variables and the number of values assigned to variables and on the other hand how to use enough variables so that to cover particular domain with various dialog flow possible paths. The *set of actions* describes what the system can do, i.e. the set of functions the system can invoke at any time (e.g. play a certain prompt, query a database, identify the set of entities, etc.). The *strategy* is a mapping between the state space and the action set. For any possible state the strategy prescribes what is the next action to perform. As a result of the action and its interaction with the external environment (e.g. user, database, etc.) the system gets some new observations (e.g. database entities, attributes, etc.). The new observations are registered and modify the state of the system. This process continues until a final state is reached (e.g. the state with legitimate SQL, XML script) [20]. The frame-based systems use templates, i.e., collections of information as a basis for dialogue management. The purpose of the dialogue is to fill necessary information slots, i.e., to find values for the required variables and then perform a query or similar operation on the basis of the frame. We use frame-based approach when we identify entities and we want the user to fill entities attributes. The dialog manager communicates with two other modules from the system: natural language understanding agents to get semantic representation of user utterance (e.g. identify entities, attributes, relationships between entities i.e. to cover all elements from entities relationships diagram) and with metadata module where databases metadata and the information delivery portal knowledge base are stored.

Natural language understanding (NLU) agents – Agent receives text input entered by the user and produces the set of possible actions (e.g. identified entities) with weights that represents the probability of correct (by means of the user understanding) entity identification. We identify two types of agents by their entities identification possibilities: one type of agents uses only current text input without using dialog history another one uses all information of current dialog state i.e. it uses all history of current dialog. In our current implementation first type of agents is IBM NLU toolbox and the second one is hybrid neural network NLU agent. More on the mentioned agents implementations see below.

Information delivery portal – is the Internet/Intranet based system for queering corporate databases, analysing retrieved data and presenting results to the user in graphical and textual templates. Information delivery web portal can be used without NLP techniques but in this paper we concentrate on natural language user interface modalities and their integration with IDP. In our system NLU components are able to map user utterance to semantic concepts that represents three types of scripts: SQL script for queering relational databases, simple script to modify HTML document and script to modify XML document generated by IDP. More on the details see in the section 3.

Information storage – is a black board for storing various information units that are used later by other system modules. It is used as the communication media between agents. In our implementation we used a hash-map as the container to store all objects by various agents.

Natural language processing agents – implements various elements from natural language processing area: named entities recognition, co-reference resolution, tokenisation, sentence splitting, gazetteer lookup, etc.

Learning agents – ensures that the system learns from data presented for learning as well as from dialogs with users.

Evaluators - are used for the particular type of agents. This means that different evaluators evaluate different aspects of agents from different viewpoints. For example, an evaluator may use the dialogue history to determine which dialogue strategy should be used (i.e. which kind of dialogue agent should be selected), while another evaluator may establish which agents is more suited to bring the answer for the user. Like in [31] our evaluators give scores for agents using a scale between [0,1].

3 Problem Domain - information delivery portal

There are many commercially successful information delivery web portal products that are available in the market. Figure 2 presents architecture of IDP implemented by our previous project JMining [15, 16] and many IDP providers implement similar three tier architecture. We have no intention to describe this architecture in details and for details about used in this paper IDP JMining we refer to [16] or SAS [29], Oracle [26], Microsoft [23], Information builders [13], etc. for details of some commercial implementations. Instead, in this section we describe architecture of middle tier that is based on atomic applications container and it's interconnections with agents of natural language dialog management.

As mentioned above one of the biggest problems with NL dialog systems is the number of states. Reduction of this number is one of the key problems in any dialog-based systems. And it is why we used JMining IDP and it's fundamental idea of an atomic applications container.

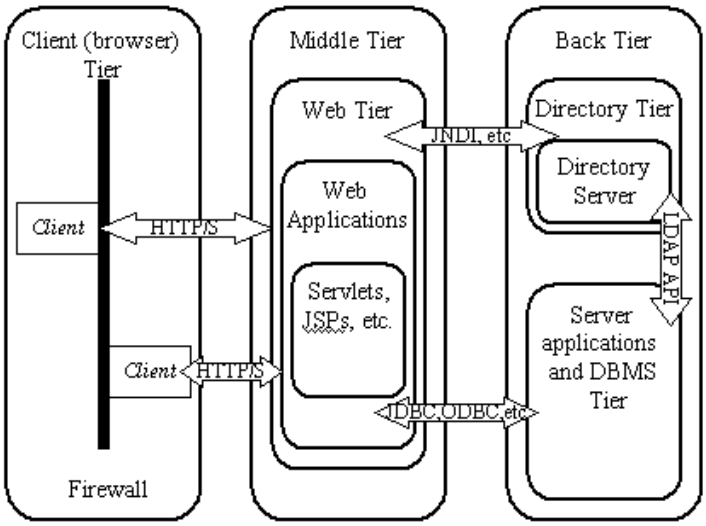

Fig. 2. Information delivery portal three-tier architecture.

IDP JMining is implemented as database and platform independent. Data base systems are accessed by one of the following protocols: ODBC, JDBC or XML. The JMining is server-based application written completely in Sun's Java programming language. Because the JMining modules are written in Java, they can run on any server platform that supports a Java Virtual Machine. Data used by the portal: account credentials, access controls, demographics, personalization parameters, and configuration information can be stored within an X500 directory services database accessible through LDAP (Light-weight Directory Access Protocol). All those data set can be stored into metadata storage of our dialog management system and then accessed and manipulated by other system of the dialog management. By such approach we achieve that such users as system administrators can manipulate (retrieve, modify or create new) some objects stored within LDAP server during NL conversation with the system. Next we describe mentioned fundamental idea of used IDP, which is call atomic applications container.

By atomic application we understand the small web application, which contains following components: database script, user interface HTML page, data representation script (XML, XSL, etc.) and documentation page (additionally there is connection to DBMS parameters, name of the application, and parent name of the application to organise all atomic applications in one single directory structure). Atomic application structure in some way resemblance to well knows web applications developing technologies like Servelets, JavaServer Pages (JSP) and Active Server Pages (ASP). With such technologies like JSP you can have the full power of general programming language like Java. But on the other hand it is unlikely that nonprogrammer or person without Java knowledge can hand such technology. On the other hand by putting more constraint on the web applications structure we achieved that nonprogrammer can successfully develop web applications. Surely that doesn't mean that no IT skills required. The user of this IDP software actually is the user who previously used such products like Microsoft Access to develop some local based database applications. Such user mostly has a good understanding of a database model as well as some basic SQL knowledge (sure most often that is no need for the user to write SQL sentences, instead it is done by interactive software wizards).

Atomic application represents one of the basic classes. Object derived form the class (like a brick in the house) is used to build an enterprise information delivery web solution. As mentioned above the set of such atomic applications can bring full portal solution to some business subject. We think that the small number of components that can be manipulated to build reliable small web application is attractive feature for the systems number of control variables is a big constrain. Below we describe in details these components that can be manipulated by our dialog management system.

SQL - set of SQL statements that are send to DBMS. There unlimited number of SQL statements that can be send to SQL server within one request but the last one must be SELECT type SQL statement. These statements are then executed in the selected database management system to retrieve information and to display it to the user through selected reporting template, which can have graphical or textual formats. Also the users have the choice of modifying these SQL statements as well as reporting templates to create their own applications.

HTML page - HTML document used to set user request parameters which can be used later to form dynamic SQL statements. Even if the primary intention of this parameter was to support dynamic SQL statements, it can be used as an independent HTML page for other web portal need. User has choice to keep parameter values permanently to the end of Internet session or just to the end of request implementation by web server.

Type of visualization object - used to choose selected data representation object from web server (e.g., graphic, bar char, some form of text (XML, HTML, TXT) layout, etc.).

XML (XSL) - Extensible Markup Language (XML) [34, 35] offers its users many advantages, including: simplicity, extensibility, and openness. XML as the atomic application component is used as some script for data visualisation (e.g., it can say which column forms x or y axis in a graphic or which field represents grouping, total variables and how they must be presented in the HTML document, etc.). From DBMS selected data are parsed with statements that are extracted from XML document. If the data comes from XML document (it is common situation in organizations that some data now can be received from XML documents instead traditional of DBMS) document can be used to transform data to HTML format.

The proposed structure of atomic application is optimal in the following way: it contains the minimum number of components that are required for building complex web portal. This IDP architecture is robust to some faults done by non-professional programmers (bugs can effect only one atomic application but the whole system is unaffected).

4 Dialogue supporting agents

The agent architecture approach to dialogue management makes it possible to use different dialogue control models, such as state-machines and forms inside the same system. The combination of different control models is useful when sub-dialogues are implemented in different ways. For example, most database retrieval tasks can be modelled efficiently by using forms, while more open-ended dialogues, such as entities identification in corporate databases may be implemented more efficiently using state-machines.

Below in the Table 1 we describe state variables and variables values in our dialog management system for data retrieval, analysis and presentations tasks. Because the system is user centric orientated the values of some state space variables are nor fixed as in [21] but has some range of flexibility.

Table 1. The state space variables.

agenda	System after the greeting of the user presents agenda. Each item of the agenda is associated with some number (e.g. 0 – no agenda item selected, 1 – select already build atomic application, 2 - manipulate Jmining parameters, 3 – get info

	from metadata storage, 4 – write SQL script, 5- manipulate LDAP objects, etc.).
objects	0- no object under current dialog state, 1- user is trying to identify corporate database to which he want establish connection, 2- user name, 3- user password, 4- system is trying to identify SQL-tables which will be used to query database, 5 – attributes for SQL script data filtering logical sentence (where), 6 – HTML page attributes (color, title, layout of input fields and for the future we plan enrich the set of values), 7 - XML document attributes (data presentation attributes, data grouping attributes, layout of presented data fields, template to use).
objects_confidence	1 – if the object under current dialog management has been established, 0 -if not.
appobject	0- if no atomic application objects, 1-SQL, 2-HTML, 3-XML, 4- visualization template. This variable is redundant but we find that it helps control dialog flow.
confidence	Like in [21] represents the confidence that the dialog management system has after obtaining a value for an attribute. The values 0, 1, and 2 represent the lowest, middle and highest confidence values. The values 3 and 4 are set when system receives "yes" or "no" after a confirmation question.
value_track	Tracks whether the system has obtained a value for the attribute (no=0, yes=1).
number_of_times	Tracks the number of times that the dialog manager has asked the user about the attribute.

Both types of dialog management agents can use all presented variables. Agents that uses state space representation method uses variable to trigger next action and move to the next state. Strategies for moving can be established from learning data. We established 94 dialogs and used reinforcement learning (RL) [21] algorithm to learn strategies for actions triggering. Form based approach uses variables to query user for specific variable values. In our current form based dialog management agent we used VoicXML [32] standard to describe simple control dialog flow based on variables described above.

Next we present simple dialog between human and our system example end shortly discuss how the system responds.

Human-Machine multi-modal dialog example

Dialog	**Description of actions generated by dialog manager**
C: Hello my name is JiMi. I am an expert in the following areas(the content of the metadata is provided in the form of hyperlinks). The last time we used Basel 2 project data	At the first you must establish an agenda. User's assistant provides a multi-framed HTML page where user can do database querying without using dialog or he can use two frames where on the one frame user puts his

area. What you want to do now? *H: Assessment type.*	request of the server in the HTML text field and submits it to the personal assistant which resides on the remote host. In the second frame personal assistant brings all answers by all agents participated at the established session in the form of formatted HTML page. Returned page contains direct answer from dialog manager (it can be retrieved data or request for some information from the user). In addition returned answer in the form of HTML page contains all relevant associations with the system objects and metadata items, in the form of HTML hyperlinks and sorted by their relevance.
C: I have 8 items associated with Assessment type. Can you chose from the list.(the list is presented in separate HTML frame). *H: Show me clients with the assessment type Operational Risk Assessment.*	System identifies the answer with the biggest confident variable value (i.e. user wants to query assessment types classification table) and shows table content in separate frame. In addition system provides the list of hyperlinks of other possible actions.
C: You want request from tables "Involved Party", "Assessment". Filtering will be on table "Assessment" column "Assessment Type" = "Operational Risk Assessment"? Please say "yes" to confirm you request. *H: yes.*	NLU agents returns semantic objects: tables - "Involved Party" and "Assessment", columns - "Assessment Type", filtering values - "Operational Risk Assessment", required object – " SQL script" with confidents level "low". Representation agent builds question for the system and tries to ask confirmation. After the confirmation the system retrieves request results to the separate frame.
C: What do you want do next. *H: Change the color to the red.*	NLU agents return semantic objects: HTML page, action "color", value – "red". Representation agent changes the colour of HTML page.
C: What do you want do next. *H: Save it.*	NLU agents return semantic objects: action – "save atomic application". Representation agent sends the message to information delivery portal to save atomic application. User gets the application name.

5 JMiningDialog architecture

In this section we present more details on our dialog management that we implemented on the general architecture described above. Figure 3 shows basic structure of the system.

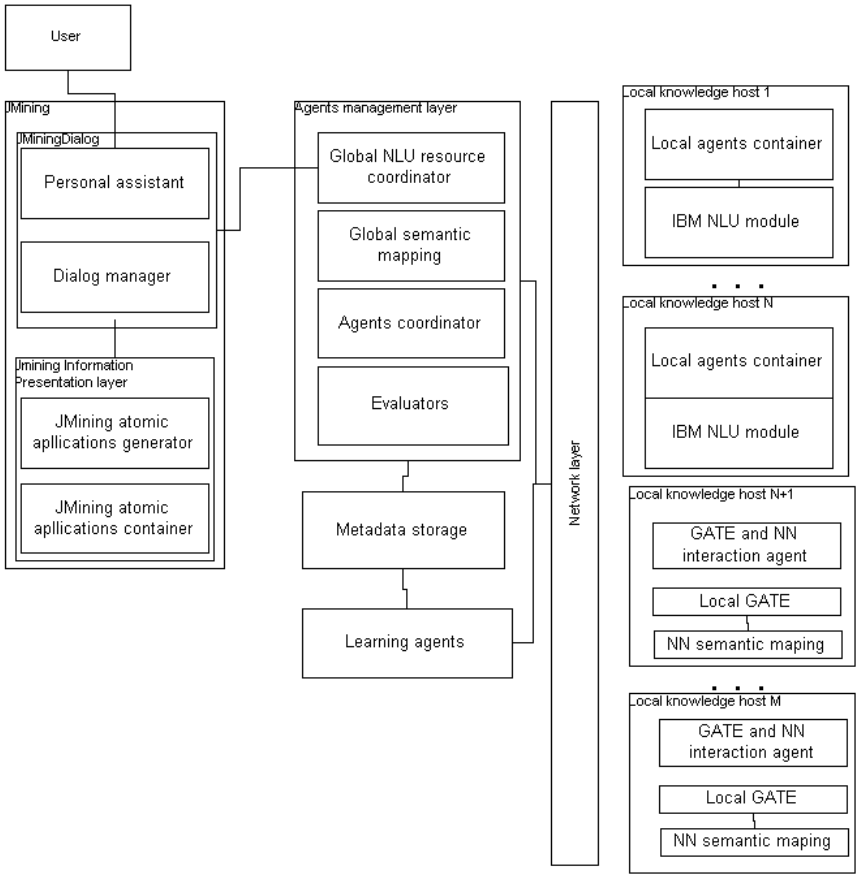

Fig. 3. Architecture of implemented integration between natural language agents and information delivery web portal.

In the rest of this section we will concentrate mainly on the natural language understanding layer. As mentioned above we implemented two types of agents. The first set of agents utilizes technologies proposed by IBM corporation: Aglets – a frame-

work for building mobile (and stationary but we used only mobile concept) agents and IBM NLU toolbox for natural language applications. The system works as follows. The master aglet sends mobile agents to remote hosts where mobile agents gather information stored locally in IBM NLU toolbox internal storage. Each agent then returns to master agent and store returned results. Results comprises of the list of action and level of confidence for each action. Each IBM toolbox is presented as a black box where you put you request and get the answer. Putting in the special IBM NLU toolbox sentences with associated actions does the learning process. The methods of IBM NLU statistical processing are not known.

5.2 Mobile agents role

At this part of the section we present our motivation of using mobile agents approach. Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming 'back home' having performed the duties set by its user. Mobile agents may cooperate or communicate by one agent making the location of some of its internal objects and methods known to other agents. By doing this, an agent exchanges data or information with other agents without necessarily giving all its information away [1].

The mobile agents need *not* be stationary; indeed, the idea is that there are significant benefits to be accrued, in certain applications, by putting away static agents in favour of their mobile counterparts. These benefits are largely *non-functional*, i.e. we could do without mobile agents, and only have static ones but the costs of such a move are high. For example, in our case consider the scenario when mobile agent is requested to find some knowledge structures related to the words *arrangement and accounts* from several users computers.

A static single-agent program would need to request for all files residing on the remote knowledge sharing host, which may total to several gigabytes. Each of these actions involves sifting through plenty of extraneous information which could/would clog up the network.

And consider the alternative. JMiningDialog NLU module encapsulates, user sentences to the entire program within an agent which consumes may be only several kilobytes which roams the other hosts included in the knowledge sharing network, arrive safely and queries these hosts locally, and returns ultimately to the home computer. This alternative obviates the high communications costs of shifting, possibly, gigabytes of information to user local computer. Hence, mobile agents provide a number of *practical*, though *non-functional*, advantages, which escape their static counterparts. So their motivation include the following anticipated benefits [1].

1. Reduced communication costs: there may be a lot of raw information that need to be examined to determine their relevance.
2. Limited local resources: the processing power and storage on the local machine may be very limited (only perhaps for processing and storing the results of a search), thereby necessitating the use of mobile agents.

3. Easier coordination: it may be simpler to coordinate a number of remote and independent requests and only collate all the results locally.
4. Asynchronous computing: you can 'set off' your mobile agents and do something else and the results will be back in your mailbox, say, at some later time. They may operate when you are not even connected.
5. A flexible distributed computing architecture: mobile agents provide a unique distributed computing architecture which functions differently from the static set-ups. It provides for an innovative way of doing distributed computation.

We have used aglets mobile agents framework in our implementation. Aglets are Java objects that can move from one host on the network to another and have all features mentioned above. More on this techniques can be found in [18].

As the second type of NLU agents we used stationary hybrid neural networks NLU agents that we build on JOONE neural network toolbox [14] and GATE general natural language processing toolbox. Gate has been used as NLP pre-processor and the results converted into binary string have been presented to the neural network. More on this techniques can be found in [17].

6 Conclusions

We presented agent based natural language dialog and understanding architecture for data querying from database management systems and presenting it to the user. We presented reasons why it is important to have in the future, solutions based on mobile agent approach even if now our data amount can be solved by stationary agents approach. Our experience showed that even if we have a limited amount of the data for teaching process, the right strategies for brief dialogs in a narrow domain can be found. We believe that integration between agents that extract information from Internet and others unstructured information sources and information delivery software brings an optimal solution for companies data analysts.

Our research shows that distributed knowledge architecture is more flexible and adaptable for such tasks then centralised solutions.

Finally we like to say several remarks concerning an open source projects. In the past ten years, open source software has become one of the most discussed topics among software users and practitioners. The increasing interest in open source software has been motivated by several factors [9]: 1. The success of products such as Linux (operating systems),Apache (http servers, etc.) , MySQL (DBMS) , GATE (NL processing), Weka (machine learning), etc.2. The uneasiness about the Microsoft or Oracle monopoly in the software industry 3. The increasingly strong opinion that "classical" approaches to software development are failing to provide a satisfactory answer to the increasing demand for effective and reliable software applications. At the initial stage of our project we understood that the code of our project must be the open source if we want to be successive in promoting our ideas. On the other hand the success of our project has been determined by the fact that we used three open sources projects in various areas: GATE in NLP, JOONE in neural networks, Aglets

in mobile agent processing. We hope that our paper will stimulate new research in this software area.

References

1. Aglets Specification (1997) http://www.trl.ibm.com/aglets/spec10.htm
2. Androutsopoulos, I., Ritchie, G. D., Thanisch, P.: Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering (1995) 1(1):29-81
3. Androutsopoulos, I., Ritchie, G. D., Thanisch, P.: Experience Using TSQL2 in a Natural Language Interface. In J. Clifford and A. Tuzhilin, editors, Recent Advances in Tem- poral Databases - Proceedings of the International Workshop on Temporal Databases, Zurich, Switzerland, Workshops in Computing, Springer-Verlag, Berlin (1995) 113-132
4. Atzeni, P., Mecca, G., Merialdo, P.: Design and Maintenance of Data-Intensive Web Sites, Proc. EDBT'98 (1998)
5. Bottraud , J. C., Bisson, G., Bruandet, M. F.: An Adaptive Information Research Personal Assistant. White paper. http://www.dimi.uniud.it/workshop/ai2ia/cameraready/bottraud.pdf
6. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Wilks, Y.: Experience of using GATE for NLP R/D. In Proceedings of the Workshop on Using Toolsets References 200 and Architectures To Build NLP Systems at COLING-2000, Luxembourg (2000) http://gate.ac.uk
7. ELF Software Co. http://www.elf-software.com
8. Esposito, D.: Talk to Your Data. White paper (1999) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnenq/html/mseq75.asp
9. Fuggetta, A.: Open source software - an evaluation. Journal of Systems and Software (2003) 66(1): 77-90
10. Huhns, M. N., Stephens, L. M.: Intelligent Agents, in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. G. Weiss (Ed.), MIT Press, Cambridge, MA (1999)
11. IBM. An Introduction to IBM Natural Language Understanding. An IBM White Paper (2003)
12. IBM Voice Toolkit V5.1 for WebSphere® Studio (2004) http://www-306.ibm.com/software/pervasive/voice_toolkit
13. Information Builders. Leveraging Your Data Architecture for Enterprise Business Intelligence. White Paper (2004) http://www.informationbuilders.com
14. Joone - Java Object Oriented Neural Engine. http://www.jooneworld.com
15. Levin, E., Pieraccini, R., Eckert, W.: Using Markov Decision Process for Learning Dialogue Strategies. Proc. ICASSP 98, Seattle, WA (1998)
16. Levin, E., Pieraccini, R., Eckert, W., DiFabbrizio, G., Narayanan, S.: Spoken language dialogue: From theory to practice," IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado (1999)
17. Litman, D. J., Kearns, M. S., Walker, M. A.: Automatic Optimization of Dialogue Management. White paper (1998)
18. Microsoft corporation. SQL Server and English Query (2003) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/architec/8_ar_ad_ 0hyx.asp
19. Microsoft corporation. Building a Corporate Portal using Microsoft Office XP and Microsoft SharePoint Portal Server. White Paper (2001)

20. Novak, J., Wurst, M., Fleischmann, M., Strauss, W.: Discovering, Visualizing, and Sharing Knowledge through Personalized Learning Knowledge Maps. White paper (2002)
21. Nwana, H. S.: The Potential Benefits of Software Agent Technology to BT. Internal Technical Report, Project NOMADS, Intelligent Systems Research, AA&T, BT Labs, UK (1996)
22. Oracle corporation. Oracle9iAS Portal 3.0.9.8.2 Architecture and Scalability. White Paper (2002)
23. Pieraccini, R., E. Levin, E., Eckert, W.: AMICA, the AT&T Mixed Initiative Conversational Architecure, Proc. of EUROSPEECH 97, Rhodes, Greece (1997)
24. Ruwanpura, S.: SQ-HAL: Natural Language to SQL Translator. Monash University (2000) http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura
25. SAS corporation. SAS Information Delivery Portal. White paper (2000)
26. Takeuchi, N. I.: The Knowledge-Creating Company. Oxford University Press (1995)
27. Turunen, M., Hakulinen, J.: Jaspis - A Framework for Multilingual Adaptive Speech Applications". Proceedings of 6th International Conference of Spoken Language Processing (ICSLP 2000) (2000)
28. VoiceXML Development Guide http://www.vxml.org
29. Watson, M.: Practical Artificial Intelligence Programming in Java (2002) http://www.markwatson.com
30. World Wide Web Consortium, Extensible Markup Language http://www.w3.org/XML
31. World Wide Web Consortium, Extensible Stylesheet Language http://www.w3.org/Style/XSL