A Multilingual Query Processing System using Software Agents

Journal of Digital Information Management

Sumam Mary Idicula
Department of Computer Science
Cochin University of Science & Technlogy
Cochin 682022, Kerala, India
sumam@cusat.ac.in

David Peter, S
Department of Computer Science
Cochin University of Science & Technlogy
Cochin 682022, Kerala, India
davidpeter@cusat.ac.in

ABSTRACT: *The goal of this work was developing a query processing system using software agents. Open Agent Architecture framework is used for system development. The system supports queries in both Hindi and Malayalam; two prominent regional languages of India. Natural language processing techniques are used for meaning extraction from the plain query and information from database is given back to the user in his native language. The system architecture is designed in a structured way that it can be adapted to other regional languages of India. . This system can be effectively used in application areas like e-governance, agriculture, rural health, education, national resource planning, disaster management, information kiosks etc where people from all walks of life are involved.*

Categories and Subject Descriptors
H.2.3 [Languages]; Query languages: ID.2.7 [Natural Language Processing]; H.5.2 [User Interface]; Natural language

General Terms
Query processing, Software agent, Multilingual data processing

Keywords: Query processing, database design, Software agents, Interfaces

Received 18 August 2006; Revised 11 Feb. 2007 and accepted 11 May 2007

1. Introduction

This work was aimed at developing an intelligent agent based system for database query processing. Database holds huge quantities of data. There are several artificial languages for manipulating the data in the database. But their usage needs knowledge about the database structure, language syntax etc. Several Natural language front-ends have been developed as a result of rigorous works done in this field of artificial intelligence. But majority of them uses English as the natural language. India being a multi-lingual country and only 5% of the population can boast about of education up to matriculation level, their use is limited. In this context, information retrieval in regional languages from database has tremendous impact. It is of very great use in application areas like e-governance, agriculture, rural health, education, national resource planning, disaster management, information kiosks etc. Even research organizations can make use of such systems to bring their findings to common man.

Most of the natural language query processing systems available are constructed to function intelligently with a single locus of control. Examples can be seen in[1],[2],[3]. But we have developed the system as a multi-agent system. To the best of our knowledge no system has been developed for Indian languages based on agent technologies so far [16]. Sybase Inc has developed a natural language front end for English using agent based technologies [15]. The system we have developed is a multi-agent open system where agents are capable of flexible autonomous action in order to meet their design objectives. Each agent exhibits pro-activity, sociability and learning capability. Pro-activeness is attributed to the goal directed behavior of each agent. Sociability is referred to the interaction with other agents to get its objective done and learning ability attributed to the capability to adapt to user's language preferences and new languages.

The database considered in this study was the National Resource Information (NRI) database. The user can give his query to the system as if he delegates a human being for information gathering. There is no rigid syntax for giving the query. The system uses NLP techniques to extract meaning of the query and retrieves information from data stores and presents the information to the user in his own native language.

2. System Architecture

Architecturally the system used distributed agent technologies based on Open Agent Architecture (OAA) for interoperation, information brokering and distribution. An agent based architecture was chosen to support this application because it offers easy connection to legacy applications and the ability to run the same set of support components in a variety of hardware configurations ranging from standalone PC to distributed operations across numerous workstation and PCs. Additionally the architecture can be extended to support mobility by running lighter weight agents like user interface agent on handheld while the more computationally intensive processing can be migrated elsewhere on the network. The agents were written in the programming language Java and they communicated via Inter-agent Communication Language (ICL).

The architecture of the system is illustrated in Figure 1. The agents collaborating in the system are user-interface agent, analyzer agent, parser agent, SQL agent and facilitator agent. A brief description of each agent follows.

The collaborating agents were distributed over a network of machines. Multiple users could use the system at the same time. In our experimental set up, we have used four machines. The Facilitator agent and the Database were run on one machine. The Analyser agent, Parser agent and SQL agent ran on another machine while two client machines were loaded with User-interface agents. All agents were programmed in Java. For running the system all the machines should have J2DK 1.4.0 and OAA software package [14].

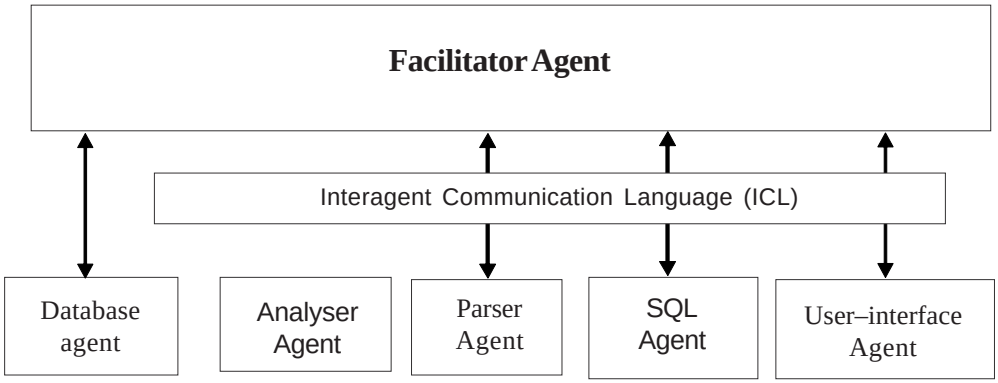

Figure 1. System architecture

2.1 User Interface Agent

It accepts typed natural language queries (Hindi / Malayalam) from the user and presents response to the queries in Hindi / Malayalam. Query can be input either through the keyboard or through the onscreen keypad with the help of touch pen. Onscreen keypads were made for both Hindi & Malayalam. The sample view of Hindi text control and onscreen Keypad is given in figure 2

grouping and attaching semantic properties to each word group. In this work the analysis of words is conducted in a domain (database) specific way rather than in a language specific way. This will make tokenizing the incoming user query easier.

The query is first separated into individual words. An individual word or a group of words that are "meaningful units" in database context are identified with the help of a lexicon. These

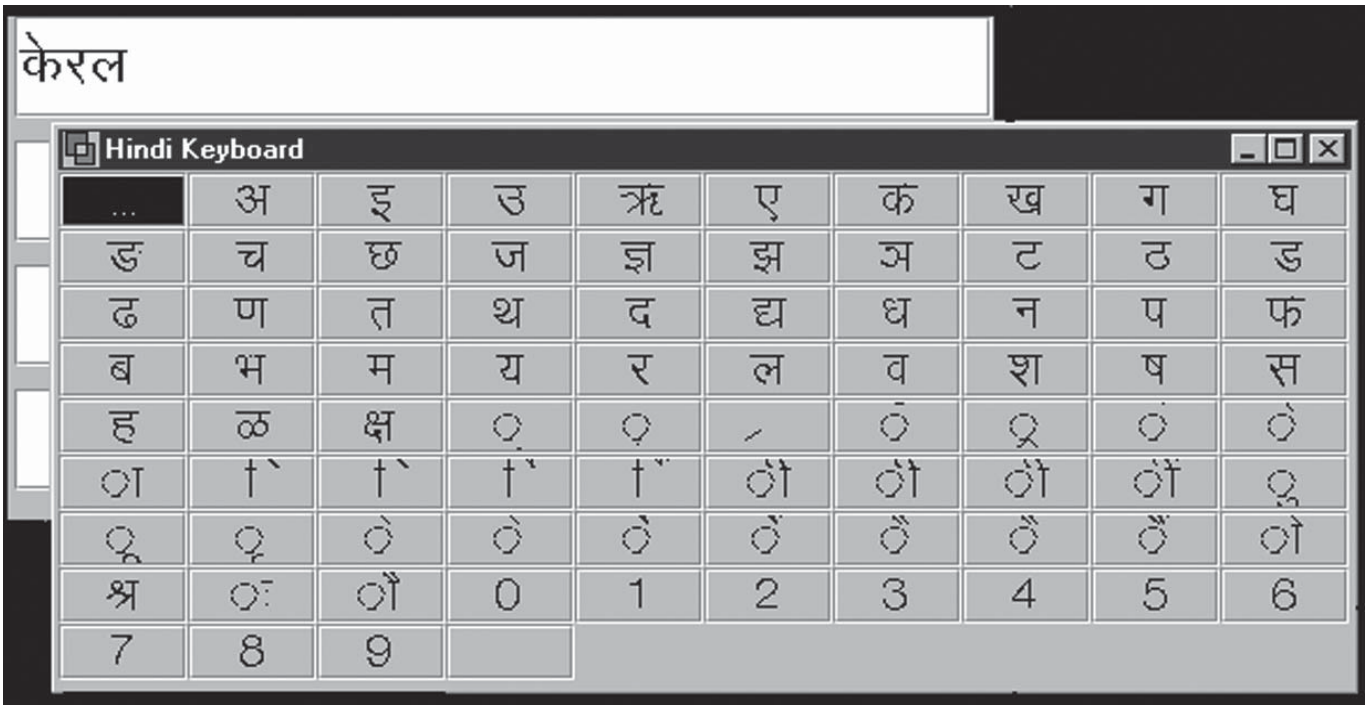

Figure 2. Onscreen Keypad for Hindi

The standard used for information exchange in Indian language is ISCII. But popular programming languages like Java uses two byte Unicode characters. So if we want to render Indian letters in Java GUI controls, we have to map ISCII to Unicode. As it was found difficult to get fonts that satisfied both ISCII & Unicode we did a font specific mapping The Indian fonts used for Hindi and Malayalm were DVBW-TTYogeshEN and MLB-TTIndulekha respectively developed by C-DAC.

2.2 Analyzer Agent

This agent performs activities like splitting the query into individual words, spelling correction, domain dependent word meaningful units are called Tokens. Knowledge content of Tokens are stored in frames. The output of the analyzer will be a table that contains token frames.

Each token frame has the following slots: -

- Value
- Tag
- Id
- Meaning

Tag is a string that implies the database specific category of the token. A list of most significant tags & interpretations are given below in Table 1.

Token category (Tag)	Interpretation
<CFLD>	Character attribute of a database table
<NFLD>	Numeric attribute of a database table
<CVAL>	Character value
<NVAL>	Numeric value
<FUNC>	An SQL function
<AND>	The logical operator AND
<OR>	The logical operator OR
<COND>	Relational operator
<DONT>	Don't care terms

Table 1. Token Categories

The tag labels are selected to imply database specific meaning. Id is an integer that along with tag will uniquely identifies a token in a token frame table, since the token frame table may contain more than one token having the same tag type. The value field contains the actual value of the token. The meaning field contains the corresponding database field name or operator name.

Consider the following query कितने गॉव मे जनसंख्या 2500 से अधिक है

The word grouping will result in following set of tokens: -

[कितने] [गॉव] [मे] [जनसंख्या] [2500] [से अधिक] [है]

The tagging is done as follows: -

Token	कितने	गॉव	मे	जनसंख्या	2500	से अधिक	है
Tag	<FUNC>	<CFLD>	<DONT>	<NFLD>	<NVAL>	<COND>	<DONT>

Table 2 depicts the token frame table produced as a result of Analysis.

	Tag	Id	Meaning	Value
1	<FUNC>	1	COUNT	कितने
2	<CFLD>	1	Village.vname	गॉव
3	<DONT>	1	---	मे
4	<NFLD>	1	Demography.totp	जनसंख्या
5	<NVAL>	1	2500	2500
6	<COND>	1	>	से अधिक
7	<DONT>	2	---	है

Table 2. Token Frame Table

Parsing is then done on the tag sequence: -

<FUNC><CFLD><DONT><NFLD><NVAL><COND><DONT>

2.3 Parser Agent

This agent checks whether the sentence is grammatically correct. Here pattern driven parsing is used [3]. Each pattern has got specific database mapping meaning. These patterns are then mapped to columns & conditions in a database.

The parsing is done with the help of a set of production rules. The production rules contain natural language pattern as antecedent and category as consequence. The incoming query is split to form a tree, which contain patterns found in the natural language query. Since our main aim is to find the conditions and columns for generating SQL, we are having only two category items. They are <CONDITION> and <COLUMN>. If any patterns is not following any of the antecedent of a production rule then it will be treated as a <FILLER>. The Table 3 gives some of the production rules used: -

Pattern (Antecedent)	Category (Consequence)
<NFLD><NVAL><COND>	<CONDITION>
<FUNC><NFLD>	<COLUMN>
<CVAL><CFLD>	<CONDITION>
<CFLD>	<COLUMN>
<FUNC><CFLD>	<COLUMN>
<CVAL>	<CONDITION>
<NFLD><NVAL><NVAL><COND>	<CONDITION>
<FUNC><NFLD>	<COLUMN>

Table 3. Antecedents & Consequents of production rules

After parsing the tokens that matches with the antecedent of a production rule, they are grouped together to form a node. The node contains information about the pattern, category and participating tokens, which is the final output of the overall understanding process. These nodes are passed to SQL agent for SQL generation.

Table 4 gives the output of the parser which is to be used by the SQL agent.

Pattern	Category	Tokens
<FUNC><CFLD>	<COLUMN>	Tokens 1 & 2
<MEM>	<FILLER>	Token 3
<NFLD><NVAL>	<COND><CONDITION>	Token 4,5 & 6
<HY>	<FILLER>	Token 7

Table 4. Set of patterns identified

2.4 SQL Agent

This agent generates SQL equivalent of the natural language query entered by the user. The output from the parser agent gives clear indications of required columns & conditions in the final SQL. The SQL is generated based on the underlying database structure and set of expert rules for query building. Interpretation of the natural language patterns that we received as a result of parsing is required for generating SQL.

For example: -

```
IF (pattern = "<FUNC><NFLD>")
   {
    ADD '<FUNC>(<NFLD>)' TO COLUMNS;
    ADD TABLE(<NFLD>) TO TABLES;
   }
```

The rule-base can be created as an XML document, so that the processing can be done easily with the help of already available XML parsers.

2.5 Database Agent

Database agent interacts with the National Resource Information Database stored in MySQL 1.4. This database contains information like electricity, land use, common index,

agriculture index, development index, demography, occupation, education, health, transport etc for every state in India up to panchayat level.

2.6 Facilitator Agent

The facilitator agent is a blackboard server agent that is responsible for coordinating agent communication and control and for providing a global data store to its client agents. It maintains a registry of agent service and data declarations [6],[14]. All communication between client agents must pass through the black board. An extension of Prolog has been chosen as the Interagent Communication Language (ICL)to take advantage of unification and backtracking when processing queries. The primary job of the facilitator is to decompose ICL expressions and route them to agents who have indicated a capability of resolving them. Thus agents can communicate in an undirected fashion, with the facilitator acting as broker

Every agent participating in the system defines and publishes a set of capabilities expressed in the ICL, describing the services that it provides [14]. These establish a high-level interface to the agent, which is used by a facilitator in communicating with the agent, and in delegating service requests to agents. The capabilities are referred as "solvables". In this work parsing a token string is a solvable of the parser agent. Displaying the output on the screen is a solvable of the user-interface agent. While performing a task, an agent can request the service provided by other agents through the procedure "Solve( )"

3. An Example Scenario

The following is an example of an operational demonstration scenario that illustrates inter-agent communication. The user tells the interface agent through the onscreen keyboard the following query. The interface agent translates this query into an ICL "solvable" expression and posts this query to the facilitator. The facilitator determines that analyzer agent is capable of performing the requested service and the request will be routed to the analyzer agent. The do_event() callback of the analyzer will be executed and the output will be posted as a ICL solvable request to the facilitator. Then the parser agent, SQL agent and Database agent will perform their services in sequence and the result will be given to the interface agent, which will render the result to the user.

4. Results

Some sample screen shots of the multilingual information retrieval system are given below. Since we have only developed a prototype of the system, we haven't populated the database considerably. Hence only few records are seen in the output. The scope of the query can be restricted to village, district or state level. In the GUI, there is a panel for entering the query. The output message from every agent will be displayed in the panel given next to this. The information retrieved from the database is shown in a separate window.

Figures 3 & 4 represent the screen shots displaying the results of queries given in Hindi and Malayalam respectively. The query entered by the user and the corresponding SQL statement generated by the SQL Agent and the result obtained can be seen in the screen shots.

5. Evaluation

We have evaluated the system using 42 questions created by two native Hindi speakers and two Malayalam speakers. The results indicated that the system works acceptably good in spite of being a prototype.

5.1 Creating Question and Answers

We explained to the evaluators about the information stored in the NRI database and the possible information they can get from it. This approach helped the users to formulate questions, which can be answered with the data stored in the database. The type of questions was balanced with character and numeric outputs. The system was tested with 42 queries. These queries were entered in both Malayalam and Hindi. In addition to simple queries, the question type included those containing built-in functions like sum, avg, max, count etc, sub queries, grouping and multiple table queries.

5.2 Examples and Analysis of Results

Fig 3 & 4 shows examples of queries entered together with the SQL statements generated by the system and also their output. We ran the system for 42 queries and evaluated the result. For 28 queries we got the corresponding desired SQL and got the correct output. But for the remaining 14, the execution didn't complete and error messages were received. The analysis of errors showed that the main cause for error was the absence of words used in the Natural language query in the language lexicon. Synonyms of all common nouns and phrases were not included in it. This was the case with both languages and the analysis agent reported these errors. The second cause was that the patterns stored in the rule base of parser agents were not complete. Some query patterns used by the native users were not adequately taken care in the rule base of the parser.

Anaphoric and elliptical queries also created problems since these aspects were not considered in depth. But all the queries in both languages that fell in the purview of the stored lexicon and rule base of parser were correctly processed.

6. Conclusion & Future work

The advantage of this system is that the user can query the data store in his own native language without knowing the complexity of the database structure and location of the database. The agents involved in the system are adaptive to the user language and the usage style. It is the facilitator agent who is planning and coordinating the sequence of tasks

The query given in Hindi was गुजरात संस्तान के किन किन गॉव में जनसंख्या 10000 से अधिक ओर स्वास्थ्य केन्द्र 1 के समान है while the query given in Malayalam was കേരളത്തിൽ 1500ൽ കൂടുൽ ജനസംഖ്യ ഉള്ളതും മാതൃ ശിശു സംരക്ഷണ കേന്ദ്രം 1ൽ കൂടുൽ ഉള്ളതുമായ വില്ലേജുകൾ ഏവ

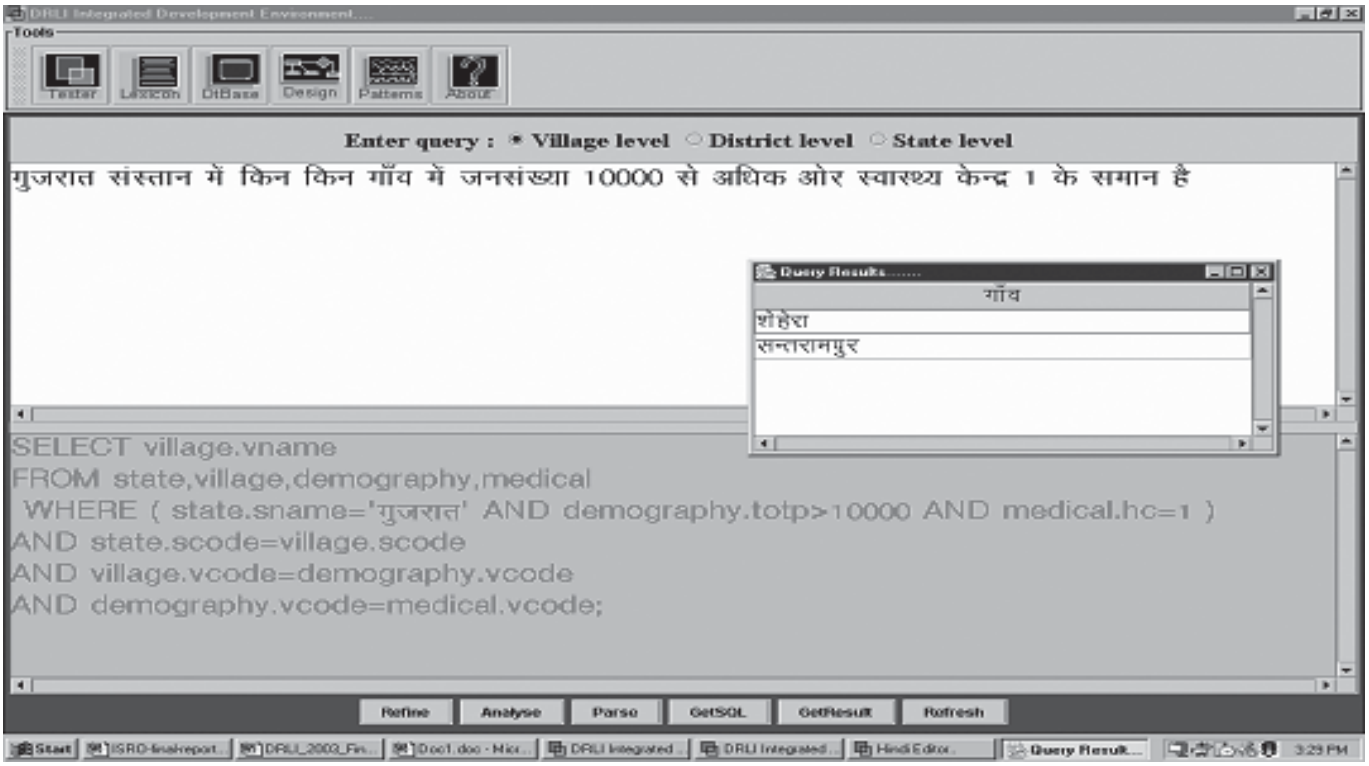

Figure 3. Screen shot for query in Hindi

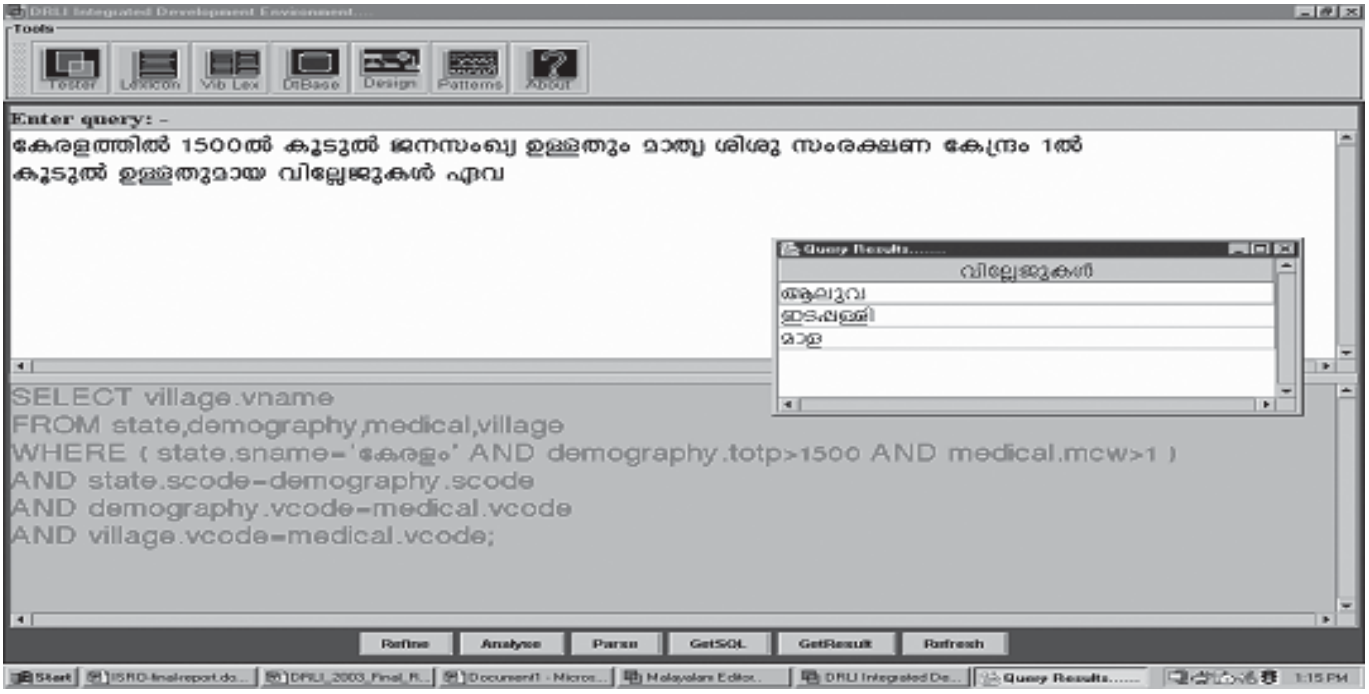

Figure 4. Screen shot for query in Malayalam

involved in query processing. The Open Agent Architecture is useful for building complex systems in which there are many heterogeneous components and in which flexibility and extensibility is important.

This work can be extended to other languages because each language family exhibits homogeneity in structure. For example other members of the Dravidian family of languages can use the language resources and processing techniques used for Malayalam. The system's performance can be improved by considering anaphoric & elliptical queries. The user interface can be made more user friendly by adding agents capable of processing multi-modal inputs like speech and gesture. Now the output is only in text form. It can be extended to include spatial information by integrating this system with spatial database. It adds more value to the result and can be effectively used by government bodies for resource planning.

Acknowledgement

Indian Space Research Organization sponsored this research work. We thank I.C Matieda, Head, Division of Informatics, Space Application Center Ahmedabad for providing many analyses and suggestions that helped an efficient implementation. We also thank the programming support given by G.Jayababu.

References

[1] Adam N.R., Gangopadhyaya, A (1997). A Form-based Natural Language Front End to a CIM Database, *IEEE Trans. On Knowledge and Data Engineering,* 9 (2) 238-250.

[2] Androutsopoulos I., Ritchie G., Tanisch, P(1993).

[3] Bharathi, Akshar., Sanyal, Rajeev (1993).

[4] Bahrathi, A., Chitanya, V., Sangal, R (1996).

[5] Bradshaw, J.M (1997).Software Agents, AAAI Press.

6] Cohen, P.R., Cheyar, A (1994). An Open agent Architecture, In: Proceedings of the AAAI Spring Symposium on Software Agents, California, 21-28.

[7] Weiss, Gerhard (1999). Multiagent Approach to Distributed Artificial Intelligence, MIT Press.

[8] Idicula, S.M., Peter, D.S(2000). Frame Based System for Understanding Malayalam, VIVEK, 13 (2) 25-33.

[9] Jennings, N.R., Woodridge M.J(1999). Agent Technology-Foundation, Application and Markets, Springer.

[10] Jurafsky, D.,Martin, J.H(2000). Speech and Language Processing, Prentice-Hall.

[11] Manning, C.D., Schutze, H (2002). Foundation of Statistical Natural Language Processing, MIT press.

[12] Michael, N.H., Munindar P.S., (1997). Readings in Agents, Morgan Kaufmann.

[13] Sugumaran, M., Narayanasamy, P (2002).

[14] The OAA Home page at http://www.ai.sri.com/~oaa

[15] http://www.ianywhere.com

[16] Idicula, S.M., Peter, D.S (2005). Intelligent Agent-based Multilingual Information Retrieval System, In: Proceedings of the International CALIBER-2005 on Multilingual Computing, Cochin, India, 8-21.