Artificially Intelligent Grid Assistant (A.I.G.A)

Roshan Sumbaly*, Abhishek Kumar, Gaurav Paruthi, Shubham Malhotra
Birla Institute of Technology & Science, Pilani
Goa, India – 403726.

Abstract – We present a Grid based application which works in collaboration with natural language processing (NLP), to act as a virtual assistant. The application can answer queries in a conversational manner and is capable of being deployed in various scenarios.

Given the prevalence of large data sources in natural language engineering and the need for raw computational power in analysis of such data, the Grid Computing paradigm provides efficiencies and scalability otherwise unavailable to researchers. In our work we explore the integration of Grid with NLP, to mine relevant answers from these distributed resources. Our system receives queries from various interfaces and then uses NLP to understand the domain of the question. The Grid then routes the queries to the correct knowledge farm, depending on the domain found. Knowledge farms are distributed components which have large annotated domain specific datasets. We propose a novel method which involves the working of the Grid and NLP in concert to mine relevant information quickly. We have also created a working model of our system deploying various existing frameworks as our sub-systems.

I. INTRODUCTION

Decades ago knowledge was restricted to certain locations across the globe and this resulted in sporadic knowledge. Now in the 21st century we face a crisis of *information overload* and hence separating the wheat from chaff is essential today.

The internet era has seen the birth of various search engines, a tool which serves as a central location for this information overload. But even though Search technology has become ubiquitous among Internet users, there is still a lookout for a tool which could go beyond the current functionality of just *displaying* the results. While this Search technology has come a long way in recent years, it's still very much in its infancy. Much work remains to enable users to gain effortless access to the *exact* information they seek.

A student-teacher relationship can serve as an ideal user-scenario wherein *exact information* would be more acknowledged than just *presented* data. Hence surely the current search engines cannot take the place of teachers. We would require an assistant / tool which would be capable of answering the queries of the student in a more interactive fashion.

In this poster, we present A.I.G.A – Artificially Intelligent Grid Assistant – a self-learning interactive assistant with conversational answers. This assistant works mutually with its base Grid Environment whose distributed power it uses to retrieve answer quickly and without overloading the system. The *knowledge farms*, which are the primary source of information, are components distributed across the Grid in a systematic fashion. They are sub-grids which contain large number of annotated corpuses.

This poster has been divided into two main sections viz. Architecture and Implementation. The first section deals with the basic architectural approach which we propose. The latter half contains information regarding the actual implementation of a model system which was setup in our campus. We have also included a concluding section which addresses some of the issues still to be dealt with.

II. ARCHITECTURE

In continuation of our earlier user-scenario, let us draw a similarity between our architecture components and a typical student teacher interaction. A student may ask a question through any of the *modes of communication* available, like e-mail, forum or during a classroom discussion. The teacher should then understand the *background* and *urgency* of the student and accordingly *elucidate with appropriate examples.*

On comparison with our architecture, the first tool should serve as a mode of communication. Hence it is required to build an interface to our assistant sitting on the Grid. The interface can be made accessible by using Web services. Web services are a collection of commonly used protocols and standards for exchanging data between applications or systems over a network. So, in order to enlarge our user base and take advantage of Web services, there is a need to connect our Grid framework with the communication platform provided by Web services.

The next important subsystem of our architecture should be able to understand the *background of the question*. It should receive queries from the various interfaces and understand the importance of *urgent nature* of these inquiries. This subsystem can be realized using the integrated power of the Grid along with natural language processing.

Natural language processing (NLP) is a subfield of artificial intelligence and linguistics. It deals with the problems of understanding and interpreting natural human languages. Our NLP module should be able to analyze the question and crudely classify it into predetermined categories. These categories or *domains* have been defined depending on the type of answer expected. For example, if the query asked is "*How large is the earth?"*, the module must search in the domain MEASURE. The domain taxonomy has been explained in the next section.

The NLP module then uses these domains to route information to the correct knowledge farm. The routing tables used should be flexible so as to enable easy updation as and when new domains are formed.

* rsumbaly@gmail.com

The Grid is the most important subsystem of the architecture. The Grid can be viewed as an aggregation of multiple machines abstracted to behave as *one virtual machine*. We have adapted the hierarchical tree structure for our Grid, as used in the Alchemi framework, a .NET Based software toolkit for implementing Grids. This hierarchic structure allows easy segregation of nodes into *knowledge farms*. We define knowledge farms as a set of nodes in the Grid containing information / data regarding the same domain. Fine details of the actual implementation of knowledge farms have been explained in the next section.

The Grid architecture is well suited for NLP applications due to various reasons. Firstly, it helps play the role of load balancer as it may queue up requests before routing, to prevent overloading of nodes. It also helps in achieving parallelism by accepting queries from multitude users at once. We aim to exploit this robust feature of the Grid to give quick answers to *urgent* queries.

The growth in scale of NLP tasks and datasets is outpacing the growth in the processing power and network bandwidths available to researchers. Thus papers like [2] suggest the use of Grid environment on the basis of two observations. Firstly, many large corpora are already held at most sites where research is being conducted and secondly most data-intensive takes place in the developmental phase and not at application runtime.

Even though the use of Grids is prevalent in computational sciences, the adoption of such an approach as a means to gain computational efficiency in NLP is still at an early stage. Previous work done in [3,4,5] have all adapted used Grid Computing to satisfy the need for computational economy arising from analysis of large data sets.

The last component of the architecture is the information / knowledge from where A.I.G.A acquires its answers. This information is generally distributed and stored as corpuses in knowledge farms.

Figure 1 Legend Explanation

No.	*Information*
1	End user submits his query to one of our multitude interfaces. The query is generally in the form of a question string.
2	The interfaces communicate with a common web service which is responsible for redirecting the query, using the internet to our A.I.G.A assistant. The internet may also be replaced by a local intranet. (as in our model system)
3	The first module which deals with the query is the NLP module. Returns *domain* information along with original query to routing information table
4	The location of correct *knowledge farm* and original query is passed to the Grid Managers
5	Grid Managers schedule a thread with the query string to the appropriate *knowledge farm.*
6	If new subcategory of domain formed *Revise Routing Information* The answer to the query is send back through the network to the web service
7	The interface displays the result to the user.

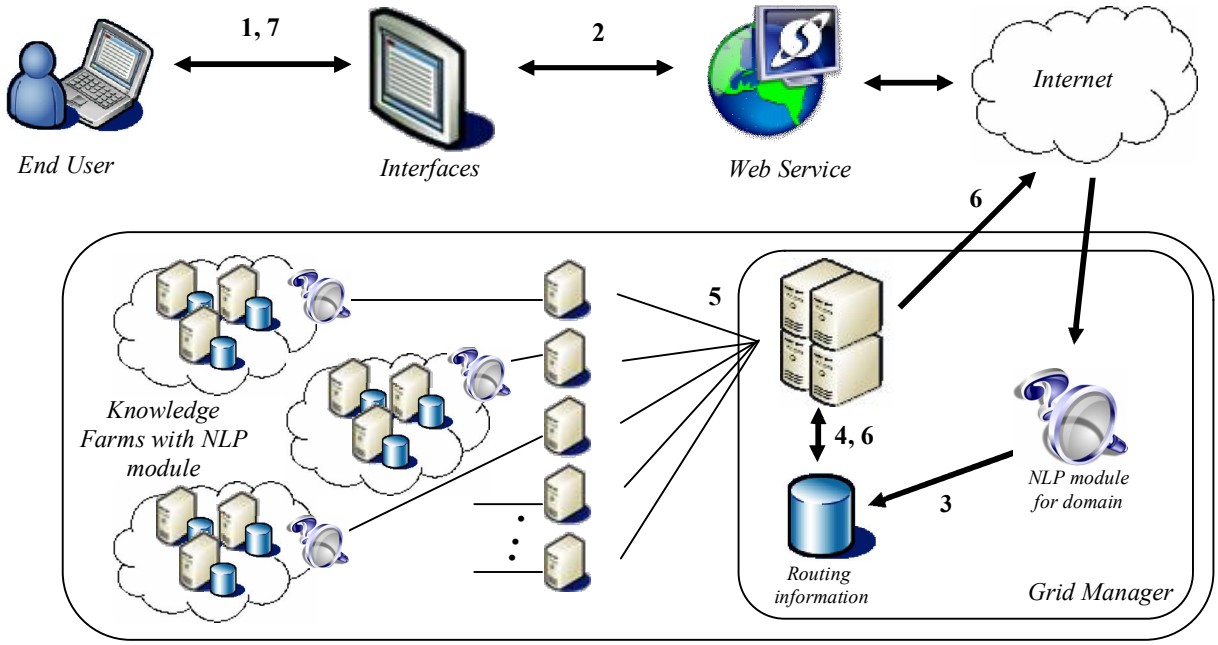

Figure 1

III. IMPLEMENTATION

In this section we explain the actual implementation of A.I.G.A. The working model was deployed in our campus i.e. BITS Pilani Goa Campus, and was made available to students for testing purposes.

We have provided various interfaces so as to enable users to have easy access to A.I.G.A. The various interfaces are as follows : (a) a Client Application build using Visual Studio (b) Integrated with MSN Messenger so as to enable users to add A.I.G.A "*as a friend*" and chat with it (c) Integrated with Microsoft Office to get *"on the fly"* responses to doubts regarding certain features of Office. (d) Integrated with ConferenceXP, a collaborative tool developed by Microsoft Research. (e) Website (f) Windows Vista Sidebar gadget. All these interfaces connect to our web service [6] (implemented using WSDL), which acts as a middle man. This middle man is also responsible for authentication of users.

There are two levels of NLP modules deployed in our architecture. The first module is based on the research by 'MITRE Corporation.'[1]. It is responsible for extracting the type of answer expected from the question. We define these answer types as *domains*. Special pre-defined patterns are used to analyze the questions based on particular words and part-of-speech tags. We build our *initial system* using the domains found in Figure 2. Let us elucidate this with an example query say, *"Who founded Microsoft?"* .Our model would match the query with the pattern **who*** which would indicate ENTITY domain. To further narrow down the search, the verb is picked up and COMPANY and PERSON domains returned. The NLP module always maintain a fresh domain list incase of any new changes.

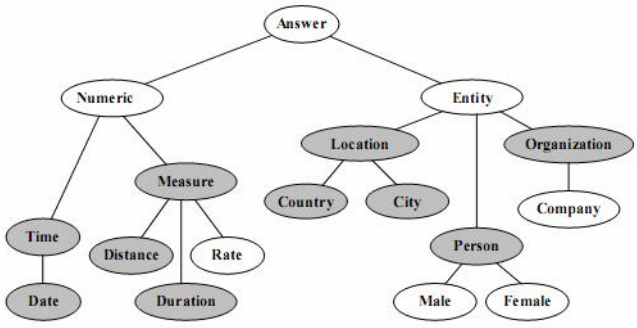

Figure 2

The second NLP module works at the knowledge farm level and is responsible for primarily responding back to domain specific questions. This module has 2 sub-modules, out of which the first one runs at runtime while the other one at developmental time. The former sub-module works in collaboration with AIML Interpreter (explained later) to respond to live queries of questions using *Case-based Reasoning*. The latter sub-module can be run at periodic sessions and is used for tagging large corpuses and store them as AIML files.

We have adapted the Alchemi framework [7], an open source framework that *allows us to painlessly aggregate the computing power of networked machines into a virtual supercomputer*. The Alchemi Grid consists of 2 major applications: (a) Manager and (b) Executor. The Executors are to be installed on all the machines and connect to a central manager. We can also have managers connecting to managers so as to form a larger Grid (Figure 3) [7]

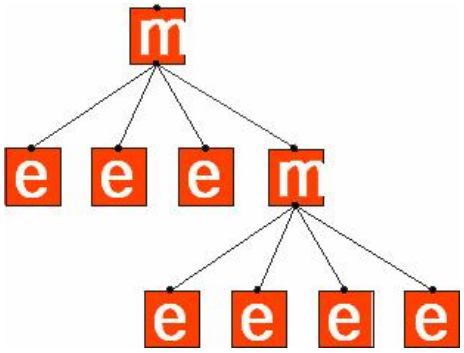

Figure 3

We use the same hierarchical architecture of Alchemi by having *Grid Managers* (refer Figure 1) at the top. Now for every *knowledge farm* (same domain) we have one manager connected to its various executors. It may happen that a domain have sub-domains (For example, MALE sub-domain under PERSON domain in Figure 2), in which case we would have another manager at the next level, similar to Figure 3. Finally at the lowest level of every *knowledge farm* i.e. at the executors, we store the actual data.

The conversational nature of our system is due to AIML (*Artificial Intelligence Markup Language*) [8], an XML dialect for creating natural languages software agents. There are various open source AIML interpreters available online. We adapted *AIMLBot* interpreter, one of the interpreters coded in C#. AIML allows us to set certain patterns, like "*What is * ?"* where * can indicate any entity. AIML also does automatic detection of new patterns for which it has no answer. This feature is called *targeting*. This adaptive nature of AIML makes it a ideal reservoir of data. This is an example of AIML patterns:

```
<category>
<pattern> ARE YOU * ? </pattern>
   <template>
   <think>
         <set name="topic">Me</set>
   </think>
         I am an intelligent
         computer program.
   </template>
</category>
```

Finally, we explain the manner in which communication takes place in the Grid. For the same, we exploit the functionality of the primary programming model supported by Alchemi i.e. *Grid threads*. They are very similar to the normal threads found in Operating

Systems and hence we can imagine a scenario where we can achieve parallelism by running multiple Grid threads concurrently. The steps followed when the query reaches the *Grid Manager* are:

- The Grid Manager produces a 'X' number of Grid threads if 'X' domains have been matched by NLP. This means the Manager will have to send concurrent requests to 'X' *knowledge farms*.(2 threads will need to be scheduled for the example of *"Who founded Microsoft?",* since 2 domains were found) Every thread contains the query information along with the path it needs to follow, as directed by the *Routing Information.*
- The threads are scheduled and received by the respective Manager of the *knowledge farm.* The knowledge farm first check the load on the executors and only then schedules the thread to ALL its executors. The first executor who gets the answer sends it up the hierarchy.

IV. Future Work

We aim at two major issues in the future viz. Personalization and integration with future semantic grids.

Personalization: By personalization, our assistant should be able to learn from user conversations and store it within the knowledge farm for future references. One way of avoiding privacy issues between multiple users is to adopt an implementation where a separate domain is created for each user and is activated only when the user "logs in" to the system.

Semantic Grid : The Semantic Grid Research Group is currently working on an extension of the current Grid in which information and services would have 'more' well defined meaning. The Research Group aims at exploiting the Semantic Web technology, using RDF and OWL. With the ongoing research on converting RDF to AIML files [9], we can use the combination of these two technologies to access RDF information interactively using natural language.

V. References

[1] A Sys Called Qanda - Eric Breck et al.

[2] Experiments with Data-Intensive NLP on a Computational Grid – Hughes et al.

[3] Blueprint for a high performance NLP infrastructure - James Curran

[4] Grid enabling natural language engineering by stealth – Hughes, Bird

[5] Building distributed language resources by Grid Computing – Tamburini

[6] Webservices - www.msdn.microsoft.com/webservice

[7] Alchemi - www.alchemi.net/

[8] AIML - www.alicebot.org/TR/2001/WD-aiml/

[9] Enhancing AIML Bots using Semantic Web Technology – Eric Freese