# HTTP://WWW.SPEECH.SRI.COM/DEMOS/ATIS.HTML

**Luc JULIA**
**Leonardo NEUMEYER**
**Mehdi CHARAFEDDINE**

SRI International
STAR Laboratory
333 Ravenswood Avenue
Menlo Park, California 94025
{julia,leo,mehdi}@speech.sri.com

**Adam CHEYER**
**John DOWDING**

SRI International
Artificial Intelligence Center
333 Ravenswood Avenue
Menlo Park, California 94025
{cheyer,dowding}@ai.sri.com

## Abstract

This paper presents a speech-enabled WWW demonstration based on the Air Travel Information System (ATIS) domain. SRI's speech recognition technology and natural language understanding are fully integrated in a Java application using the DECIPHER™ speech recognition system and the Open Agent Architecture™.

## Introduction

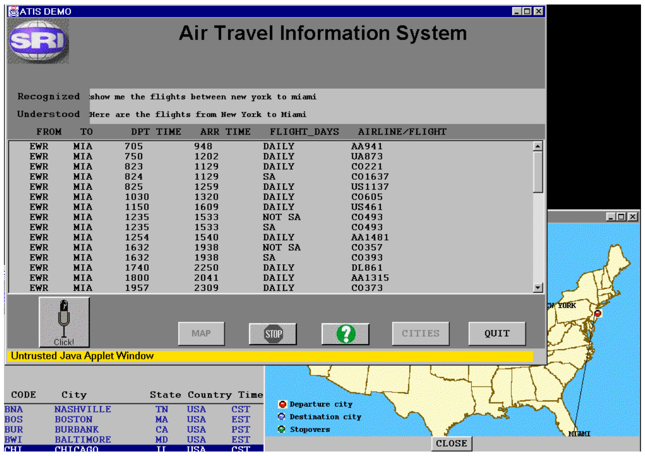
The WWW, especially in conjunction with Java, offers the potential of an unprecedented degree of platform independence, and consequent enormous reduction in costs to distribute, port, and maintain software, data resources, and updates. Speech-enabled WWW interactions vastly increase accessibility to the WWW, including access from the nearest telephone (which may be in your pocket), and access by disabled populations, or whenever typing and mouse interactions are inconvenient. Our first demonstration of speech-enabled WWW access is based on the Air Travel Information System (ATIS). This system, combining speech recognition and natural language (NL) understanding technology, is capable of responding to spoken queries in the domain of air travel.

## System Overview

The ATIS demonstration can be found at **HTTP://WWW.SPEECH.SRI.COM/DEMOS/ATIS.HTML**. This web site presents instructions for running the demonstration, and gives a telephone number to call; since no standard yet exists for recording sound from a Web browser, the telephone was chosen as the best means of allowing a large user population to provide speech input to the system.

Before beginning the demonstration, the user can determine whether the system is already in use by pushing a button in the browser. If asked to proceed (i.e., telephone line is not busy), the user calls the given number and is asked to type an identification sequence to ensure that the call corresponds to the current ATIS session. Once this security step is completed, and a disclaimer message (saying that the session may be recorded) is approved, the user is presented with the main screen of the application, as shown in Figure 1. An animation illustrates which button to push to begin entering spoken queries.

A typical first spoken query might be: "*Show me all flights from Boston to San Francisco on Monday afternoon*". While the speech engine is working, the user is apprised of progress through partial recognition results displayed as they are calculated. Once the recognition process has completed, a paraphrase of what the system understood is shown on the line following the recognized string. This juxtaposition of recognized words and inferred meaning is useful, as the natural language process often can overcome misrecognitions and correctly determine the appropriate intention from vague or nonsyntactic queries. After interpretation and retrieval, the results of the request are shown in a table listing the set of attribute values (e.g., departure time, arrival time, flight number) that solve the query. The user then has the opportunity to narrow the search, refining the search criteria of the previous query: “*Which is the cheapest flight?*”. Automatic detection of context switches is one of the difficult problems addressed by the NL component.



Recognized show me the flights between new york to miami

Understood Here are the flights from New York to Miami

| FROM | TO | DPT TIME | ARR TIME | FLIGHT_DAYS | AIRLINE/FLIGHT |
|---|---|---|---|---|---|
| EWR | MIA | 705 | 948 | DAILY | AA941 |
| EWR | MIA | 750 | 1202 | DAILY | UA873 |
| EWR | MIA | 823 | 1129 | DAILY | CO221 |
| EWR | MIA | 824 | 1129 | SA | CO1637 |
| EWR | MIA | 825 | 1259 | DAILY | US1137 |
| EWR | MIA | 1030 | 1320 | DAILY | CO605 |
| EWR | MIA | 1150 | 1609 | DAILY | US461 |
| EWR | MIA | 1235 | 1533 | NOT SA | CO493 |
| EWR | MIA | 1235 | 1533 | SA | CO493 |
| EWR | MIA | 1254 | 1540 | DAILY | AA1481 |
| EWR | MIA | 1632 | 1938 | NOT SA | CO357 |
| EWR | MIA | 1632 | 1938 | SA | CO393 |
| EWR | MIA | 1740 | 2250 | DAILY | DL861 |
| EWR | MIA | 1800 | 2041 | DAILY | AA1315 |
| EWR | MIA | 1957 | 2309 | DAILY | CO373 |

| CODE | City | State | Country | Time |
|---|---|---|---|---|
| BNA | NASHVILLE | TN | USA | CST |
| BOS | BOSTON | MA | USA | EST |
| BUR | BURBANK | CA | USA | PST |
| BWI | BALTIMORE | MD | USA | EST |
| CHI | CHICAGO | IL | USA | CST |

Figure 1: ATIS screen

A user who has defined an itinerary can ask for a map where the selected cities are hyperlinked to Web sites containing information about them. Other on-line help is available, including a list of the 46 U.S. cities that the user can talk about and a general guide for entering acceptable queries.

## Open Agent Architecture

The Open Agent Architecture™ (OAA) is a framework for constructing applications through the process of assembling and organizing communities of multiple software agents [Cohen et al., 1994; Moran et al., 1997]. Agents participating in an application can be distributed across multiple computers, and can be written in different programming languages (currently supported languages include C, Prolog, Lisp, Delphi Pascal, Visual Basic, and Java[1]).

In terms of extensibility (easily adding new functionality or upgrading one technology by another), we have found that OAA provides numerous advantages over approaches for building monolithic, stand-alone applications. In addition, we have found the OAA more flexible than most distributed object frameworks such as CORBA, in that the interactions among components are not necessarily fixed in advance; as agents connect and disconnect from the network, the system adapts to the dynamic set of available resources when attempting to resolve a given goal. These properties are achieved by the use of a high-level “Interagent Communication Language” (ICL) that can be interpreted by a special class of agents, called Facilitator Agents, which are responsible for reasoning about the best way to accomplish a goal given the current set of agents and resources.

In the demonstration system, the OAA has been used to bring together the modules and technologies that provide the application’s functionality: the user interface, login and identification over the telephone (security), speech recognition, NL understanding, database access, fail-safe monitoring, and log file generation. We will now look in more detail at the technologies behind some of these components, and at how they were integrated to create this demonstration system.

## Speech Recognition

The speech recognition component is a real-time version of SRI’s DECIPHER™ continuous speech recognition system based on context-dependent genonic hidden Markov models (HMMs) [Digalakis et al., 1996]. SRI’s DECIPHER™ technology recognizes natural

[1] All product and company names mentioned in this paper are the trademark of their respective holder.

speech without requiring the user to train the system in advance (i.e., speaker-independent recognition) and can be distinguished from the few other leading-edge speech recognition technologies by its detailed modeling of variations in pronunciation and its robustness to background noise and channel distortion. These features make the DECIPHER system more accurate in recognizing spontaneous speech of different dialects and less dependent on the idiosyncrasies of different microphones and acoustic environments.

The telephone acoustic models were trained on approximately 50,000 utterances from SRI's proprietary Partyline corpus, a digitally collected telephone corpus containing read utterances from 3000 callers. The backed-off bigram language model was trained on approximately 23,000 utterances of ATIS spontaneous speech data.

## Natural Language Understanding

The NL understanding component accepts word hypotheses from the recognizer, and produces two outputs, the simplified logical form (SLF) and the answer to the query.

A major advantage of using an agent-based architecture is that it provides simple mix and match of components. When constructing an application requiring NL, we can select the NL system that best meets the application's requirements. Several OAA-enabled technologies have been used to provide NL for various projects: the simplest one, based on Prolog's Definite Clause Grammars (DCGs), is fast, easy to add vocabulary to, and capable of changing its vocabulary and grammar dynamically as a function of which agents are connected to the network. Another NL system is based on CHAT [Pereira, 1983] and is good at representing temporal expressions. For the ATIS domain, we are using a Template Matcher [Jackson et al., 1991] in the application described here; our most capable and efficient NL system, GEMINI [Dowding et al., 1993], has been incorporated in a telephone-only version of the ATIS system [Bratt et al., 1995].

The Template Matcher operates by trying to build "templates" from information it finds in the sentence. Based on an analysis of the types of sentences observed in the ATIS corpus, we constructed eight templates that account for most of the data in the domain: flight, fare, aircraft, city, airline, airport, ground transportation and meanings of codes and headings. Templates consist of slots that the Template Matcher fills with information contained in the user input. For example, the sentence

> Show me all the United flights Boston to Dallas nonstop on the third of November leaving after four in the afternoon.

would generate the following flight template:

```
[flight, [stops,nonstop],
         [airline,UA],
         [origin,BOSTON],
         [destination,DALLAS],
         [departing_after,[1600]],
         [date,[november,3,current_year]]]
```

In contrast to the template matcher system, the Gemini NL system uses a broad-coverage grammar of English to produce a full syntactic and semantic analysis of the speaker's utterance. This analysis produces a scoped logical form expression, which is then translated into a database query. While Gemini is not as fast as the template matcher for the ATIS application, it has several advantages. Since Gemini uses a word-synchronous bottom-up parser, it is possible to start processing partial recognition hypotheses before recognition is completed. This allows speech recognition and language processing to be carried out in parallel. Gemini also has specialized components to recognize and correct certain types of nongrammatical utterances, including sentence fragments and verbal repairs. In recent applications, Gemini grammars have been used for both language interpretation and speech recognition, by using a grammar compiler to convert the grammar in Gemini's formalism into a grammar usable by the DECIPHER recognizer.

One of the primary advantages of using the OAA is that it allows us to pick and choose between these competing approaches for NL analysis, or even combine them in the same system when that is more appropriate.

## Implementation

Figure 2 illustrates the OAA agents assembled to produce our ATIS-Web demonstration. Each pictured component connects to a Facilitator agent, which manages all communication and interactions among the client agents. Because of a security restriction imposed by Java and Internet browsers, the Facilitator must run on the same machine as SRI's Web server; all other agents may be distributed across different machines at SRI, spreading the load as desired. In addition, a fail-safe agent (not pictured) monitors the status of all running agents, and can restart an agent if it crashes or if the machine on which it is running goes down.

Our main objective for this demonstration was to take advantage of the distribution properties of the Web to make ATIS easily available to a large number of users. We first developed the system by using standard HTML and a CGI executable to provide a Web-accessible text version of the user interface. Many problems, such as restricting access to the WWW demonstration only to the user actually talking to the system over the phone, were worked out by using the CGI interface. However, interactive visual feedback was limited with this technology, so the user interface component was rewritten in Java.

As this was the first OAA agent module written in Java, much of the implementation work was to port the OAA agent library to this new, still developing language. To fit into Java's Object Oriented paradigm, the OAA library was structured as two class objects: a low-level communication object (TCP) and a high-level object providing common functionality available to all OAA agents (e.g., ICL, ability to set triggers or monitors).

The multithreading capabilities of Java were used to improve the speed and the efficiency of each communication layer. We used multithreaded parallelism at both the TCP level (waiting continuously for data) and at the high level for processing and routing ICL messages.

The technologies we have described (speech recognition and natural language) had been encapsulated as OAA agents for other projects, and were therefore directly reusable. In addition, we were able to "borrow" several other pre-existing agent functionalities, such as telephone control and fail-safe monitoring. New Login and LogFile agents were developed for this application.

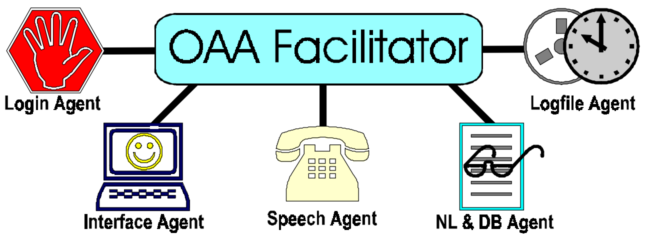


Figure 2: ATIS demonstration architecture

## Current & Future Work

Continued work will focus on three general areas: improving individual components, adding new functionality to the demonstration, and taking better advantage of OAA's inherent parallelism to produce more robust interactions.

The system currently generates a log file containing the audio files and the interpreted queries for each user interaction. One improvement would be to use this large amount of data (1700 connections) to retrain the recognizer, and to reevaluate and adapt the NL modules.

Another improvement would be to allow more users to access the demonstration at the same time. Currently, only a single user can be connected at a time, a restriction imposed by the single telephone line used to acquire speech input. We might address this issue by adding a multiple telephone line board, or by taking advantage of new standards for audio input in Java to capture speech locally with the user's Web browser, using the network to transmit the audio data to our speech server.

We would like to add some new functionality to the demonstration, as well. By plugging in new agents encapsulating SRI's English-to-French translation technology and a commercial French text-to-speech engine, we will be able to effortlessly extend the current demonstration to exhibit real-time simultaneous spoken language translation.

At the interface level, we would like to combine the demonstration's speech input with other modalities, such as the mouse, touch-screen or pen. Incorporating techniques developed in [Cheyer and Julia, 1995] would enable the user to simultaneously speak and draw directly on the U.S. map, as in "*Show me all flights from here to Dallas leaving tomorrow*".

Finally, we would like to further explore OAA's parallelism capabilities, by setting up two NL systems, the Template Matcher and Gemini, in competition with each other. Gemini is slower but gives better results; when Gemini can give a result within an acceptable amount of time, the system should use this result. Otherwise, the system will rely on results computed by the Template Matcher.

## References

Bratt, H.; Dowding, J. and Hunicke-Smith, K. 1995. The SRI Telephone-based ATIS System, 22-25. Austin, Texas: ARPA Workshop on Spoken Language Technology.

Cheyer, A. and Julia, L. 1995. Multimodal maps: An agent-based approach, 103-113. Eindhoven, The Netherlands: International Conference on Cooperative Multimodal Communication.

Cohen, P.; Cheyer, A.; Wang, M. and Baeg, S. C. 1994. An Open Agent Architecture, 1-8. Stanford, California: AAAI Spring Symposium Series on Software Agents.

Dowding, J.; Gawron, J. M. Appelt, D.; Bear, J.; Cherny, L.; Moore, R. and Moran, D. 1993. GEMINI: A natural language system for spoken-language understanding, 54-61. Colombus, Ohio: 31st Annual Meeting of the Association for Computational Linguistics.

Jackson, E.; Appelt, D.; Bear, J.; Moore, R. and Podlozny, A. 1991. A Template Matcher for Robust NL Interpretation, 190-194. Pacific Grove, California: 4th DARPA Workshop on Speech and Natural Language.

Moran, D.; Cheyer, A.; Julia, L.; Martin, D. and Park, S. 1997. Multimodal User Interfaces in the Open Agent Architecture, to appear. Orlando, Florida: Intelligent User Interfaces'97.

Pereira, F. C. N. 1983. Logic for Natural Language Analysis. Ph.D. diss., University of Edinburgh.

Digalakis, V.; Monaco, P. and Murveit, H. 1996. Genones: Generalized Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognizers, 281. IEEE Transactions of Speech and Audio Processing, Vol.4, Num. 4.