AMICA: the AT&T Mixed Initiative Conversational Architecture

Roberto Pieraccini, Esther Levin, Wieland Eckert

AT&T Labs - Research, 180 Park Avenue , Floram Park, NJ 07932-0971, USA
{roberto,esther,eckert}@research.att.com

ABSTRACT

In this paper we show how it is possible to design and implement a general architecture that is suitable for the rapid development of human/machine natural language, mixed initiative dialogue systems. The architecture proposed here relies on the assumption that a dialogue system can be modularized into different actions or functions that can be designed separately and implement basic aspects of the dialogue behavior, and a strategy that is fairly independent of the particular application.

INTRODUCTION

Developers of human/machine natural language dialogue systems often state that one of the main problems of the field is that of finding a general framework that would easily fit different applications, and would allow for a rapid development of new system. One of the reasons of this difficulty lies in the lack of separation, in many existing systems, among the different levels of competence that intervene during the dialogue activity. When one looks at the dialogue as the result of logical activity whose basic rules are independent of the particular application, the language, and the medium in question (as for instance in [5]), the design of dialogue systems becomes more of an engineering problem and less of an art. For instance, in the design of a form filling application the dialogue flow is generally represented by a tree that takes into account all the possible outcomes,. Instead one could design a function that implements the basic logic principle that, when some pieces of information is not present in the current memory, the best move for the system is asking for it.

In this spirit we present AMICA, a general model of a mixed initiative dialogue system based on the identification of a set of general, logically motivated functions, called *dialogue actions*. We think that for certain classes of dialogues there exists a finite (and small) number of such actions that can be implemented in a general way and parametrized in order to be used for different applications.

In this work we restrict the discussion to those dialogue systems that are devoted to the extraction of information from a database. For an effective dialogue the machine needs to be able to accomplish the actions in the following inventory:

Understanding: that is the transduction of the user input (generally written or spoken natural language) into a formal representation conveying the semantics of the message.

Verbalization: that consists in transducing the machine output into a form that is promptly understood by the user (e.g. natural language). We distinguish between *data verbalization*, that requires specific knowledge about the semantic structure of the domain, and *sentence verbalization* that needs general, domain independent, knowledge.

Contextual Interpretation: consists in the interpretation of the current user input in terms of the history of the interaction. It includes both *dialogue interpretation*, namely the ability of dealing with expectations set by the machine itself (e.g. disambiguation on the basis of previous questions asked by the machine), and *discourse interpretation*, namely the ability of taking into account the context set by the user during the course of the whole interaction. In both cases it is necessary to be able to recognize ambiguity and to formulate disambiguation questions.

Constraint Consistency Verification: consists in the operation of spotting the presence of inconsistent sets of constraints provided by the user. These inconsistencies might be the result of user errors, like false presupposition, or simply errors of the recognizer.

Data Retrieval: consists in the assembly and submission of a query to a local or remote database according to the current information gathered by the system.

Constraining: is the operation the system performs when asking the user for additional information. This operation is required due to both the limited bandwidth of the communication protocol, and the limited capacity of humans to analyze big sets of data. In general constraining is required when an under-constrained query produces too many results. In certain cases it is possible to predict that a query is under-constrained without actually accessing the database, for instance by specifying a set of *minimal constraints* for each particular topic.

Relaxation: consists in the ability of analyzing the failure of a database query (i.e. empty set of data), and proposing the user alternative solutions obtained by relaxing one or more constraints.

Sequencing: is the operation required when presenting a long list of items that exceed the capability of the bandwidth and cannot be further reduced by constraining. Sequencing should allow the user to navigate through the list.

Although there are other basic functions the dialogue system should be able to perform, like for instance *ambiguity resolution* and the capability of coping with possible noisy input (e.g. the errors of a speech recognizer), we limit here our discussion to the previous functions.
Once these (and maybe other) basic functions of dialogue have been defined we need two other components in order to build a dialogue system, namely a representation of the current status of knowledge of the machine (called *state* [6]), and a mechanism that invokes the required function when needed, that we call *strategy*.
In our implementation the strategy is represented by recursive transition network, whose arcs represent conditions on the state, and whose nodes represent handles to the above mentioned functions.
The idea of implementing the dialogue as a constraining/relaxation activity can be found in [2], in [4] most of the functions described here were also and in [5] the idea of separating the dialogue activity into several levels of competence, starting with the innermost more general logical functions is introduced.

USER INPUT
CITY: BOSTON
CONTEXT
SUBJECT: GROUND TRANS
ORIGIN: DENVER
DEST: BOSTON
TIME: MORNING
AIRLINE: DELTA
CURRENT
SUBJECT: GROUND TRANS
GROUND CITY: BOSTON
EXPECTATIONS
GROUND CITY: ?
DATA
TRANSPORT: TAXI, LIMO, RENT-CAR
SYSTEM OUTPUT
SUBJECT: GROUND TRANS
GROUND CITY: BOSTON
TRANSPORT: TAXI, LIMO, RENT-CAR

Figure 1: Dialogue State

DIALOGUE CONTROL

For explaining how the AMICA architecture works it is necessary to distinguish between *dialogue state* (i.e. all the information available at a certain time during the course of the dialogue) and *control state* (i.e. the identification of a particular situation in the control flow of the dialogue). The dialogue state is identified by the information contained in a data structure similar to the one shown in Figure 1; each field in the dialogue state pertaining to different kinds of information that, in the current implementation, is represented by a flat keyword/value structure[3].
Control states correspond to the nodes of a graph representing the strategy, like the one shown in Figure 2. A control state includes the reference to one of the dialogue functions introduced above (represented by the node labels in Figure 2), and a set of transitions to other control states that depend on conditions set on the dialogue state. Each dialogue function job is to read and update the current dialogue state. For instance, the dialogue function DATA RETRIEVAL reads the current dialogue state, uses the CURRENT MEANING portion of the state for building a database query, and writes the query result (i.e. the set of tuples) in the DATA field.
An example of condition, for instance for the arc labeled TOO LARGE, could be (N(DATA) > 3), expressing the condition of a number of retrieved tuples greater than 3.
The basic operation performed by the dialogue controller consists in:

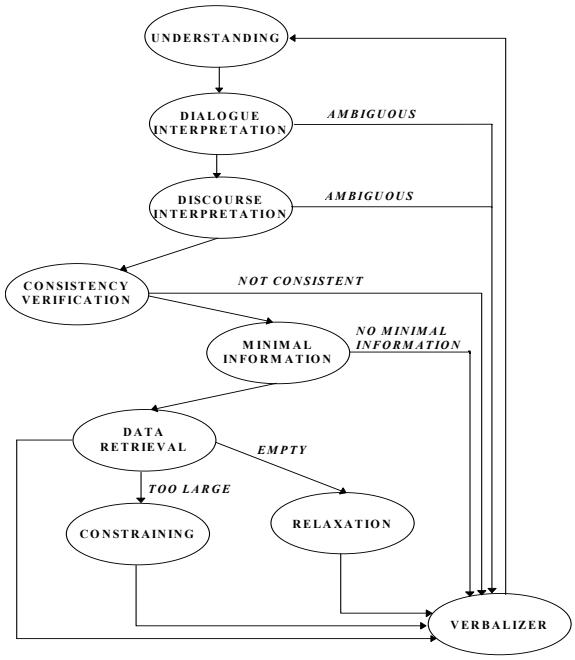

Figure 2: Example of Dialogue Control

Invoking the function indicated by the current control state (this has the effect of updating the current dialogue state),
Moving to the next state according to the transition that has a matching condition on the current dialogue state.
An example of a fragment of dialogue drawn from the ATIS demonstrator is shown in Table 1. Each sentence in the example is annotated with the dialogue actions that were used during the processing. For instance, in order to produce the sentence at turn 7, the controller went through the DISCOURSE INTERPRETATION control state where a situation of ambiguity was detected (arc AMBIGUOUS in Figure 2) and the request for a disambiguating question was included in the SYSTEM OUTPUT field of the dialogue state. The SENTENCE VERBALIZER then produced the requested English sentence. The example of dialogue state of Figure 1 refers to the situation at the end of turn 9, where the DIALOGUE INTERPETER was able to correctly interpret the user's input CITY:BOSTON as GROUND CITY because the corresponding expectation was set at turn 7. Hence the correct current meaning is used for formulating the database query, RETRIEVE the requested data, and then formulating a request for the VERBALIZER.

PORTABILITY

The process of developing a dialogue system for a new application is greatly eased by the functional independence of the various modules that compose the system. Practically each module can be designed and optimized independently. Moreover the algorithmic structure of each module is application independent. This leads to purely *table driven* modules: in principle no software has

to be rewritten for a new application, but only sets of parameters have to be updated. In the following we will give a brief description of some of the modules.

Understanding

The understanding module is based on stochastic conceptual models[3]. We enhance the basic stochastic models by introducing the possibility of including handcrafted concept descriptions when those concepts are not represented in a corpus, and by importing concepts from other corpora. An interesting point that contrasts AMICA with other dialogue system is the complete independence of the understanding system with the dialogue. Understanding is seen as a process of transduction between natural language and a symbolic representation (in some sense is the analogous of a quantization process). The interpretation of the symbolic representation is demanded to other modules, and can be different for different applications.

Verbalization

The verbalization process is currently implemented as a set of sentence templates for the verbalization of sentences like "Please tell me your origin city". For the verbalization of database tuples the process relies on the semantic description of the domain (corresponding to the database relational structure) annotated with lexical items. For instance verbs and prepositions are attached to pairs of database attributes, like:

FLIGHT_ID *flies to* DESTINATION_AIRPORT
FLIGHT_ID *leaves from* ORIGIN_AIRPORT
FLIGHT_ID *is served by* AIRLINE
FLIGHT_ID *uses a* AIRCRAFT

These pairs of predicates are organized into a finite state network that is then used for building sentences with an arbitrary number of attributes, like

Flight UA706 leaves from Denver, flies to Newark, is served by United, and uses a Boeing 747.

Discourse and Dialogue Interpretation

Discourse and dialogue interpretation are general modules based on a set of tables that specify in which way the current symbolic information derived from the user input has to be interpreted in the context of past user (discourse) or system (dialogue) information. An example of how this processing is parametrized, is the *context masking table* that specifies in which way symbols in the context are masked by new symbols. For instance, if a new MONTH symbol is given, it masks MONTH, DAY, and TIME symbols that might be present in the context. Similarly and *expectation table* specifies how ambiguous information can be interpreted in presence of expectations set by the system (e.g. questions). For instance a symbol CITY can be interpreted as DEPARTURE_CITY, ARRIVAL_CITY, or GROUND_CITY, depending on the value of on or more EXPECTATION symbols in the current dialogue states. The result of the discourse and dialogue interpretation modules is a keyword/value data structure that represents all the current information provided by the user.

Consistency Verification

This module relies on the structure of the database. A table specifies which sets of partial information have to be verified for consistency. For instance MONTH and DAY (avoiding mistakes like September 31st), or CITY and STATE (like Denver, New Jersey). If elements of those sets are present in the current interpretation, queries are formed for verifying their consistency in the database, and output sentences are generated in case of inconsistency (e.g. *September 31st is not a valid date*).

Data Retrieval

The database retrieval module is probably the most complex component of a dialogue system that is generally approached in a ad-hoc fashion. Instead we provide a mechanism based on the idea of constraint networks [1] that easily generalizes to any relational database. A constraint network is a graph that represents a set of variables that are related by logical constraints. In the case of a relational database multidimensional variables represent the values (i.e. tuples) of the relations, and some variable dimensions (i.e. tuple attributes) are

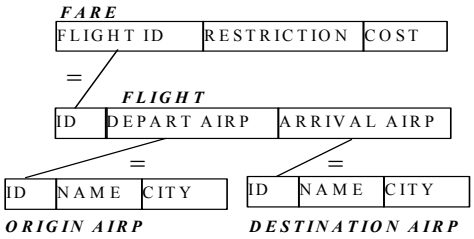

Figure 3: Example of constraint network

linked by equality constraints that have to hold (i.e. the linked attributes correspond to the same entity). This is shown in Figure 3 where the boxes represent database relations that assume different roles in the application (e.g. the relation AIRPORT can assume the role ORIGIN and the role DESTINATION)., and the arcs (internal constraints) represent equality constraints. When external constraints are applied to the network (i.e. a set of constraints that apply to some of the relations as the result of a query), an efficient algorithm finds the tuples in each relation that comply both with external and internal constraints. The data retrieval algorithm is application independent and requires only the description of the database structure.

Relaxation and Constraining

Relaxion is the process of finding approximate solutions to a given query. The problem can be approached by using a metric on the space of the database attributes in order to establish the semantics of 'distance' of different solutions, and also introducing an order in the significance of different constraints with respect to the relaxation operation (e.g. relaxing time and airline is preferable to relaxing the origin airport). Again the algorithm can be generalized and parametrized for particular applications. A relaxation results in an output sentence like: *There are no flights with airline A, but there are flights with airlines B, C, D, and E.*

The constraining operation consists in selecting a constraint (and its possible values) to suggest to the user in

order to reduce the number of current solutions to the query. The significance of different constrains used in the relaxation is used as well in constraining. This results in output sentences of the kind: *There are too many flights to list, please choose an airline among A, B, C or D*.

It is worth noticing that the user does not have to answer the requests of the system at any point during the dialogue. For instance, when the *airline* is requested (as in the previous example) the user might decide to say: *I don't care about the airline, I want the earliest flight.* In this case the constraining mechanism is going to be bypassed at the current turn by the restriction imposed by the user (the *earliest*). Both constraining and relaxation keep track of the attributes that have been constrained/relaxed through special fields of the dialogue state. If the user keep avoiding to provide other constraints as suggested by the system

SUMMARY

We introduce in this paper a general architecture for dialogue systems based on the identification of logical functions and their use by an explicit strategy. Both the algorithmic structure of the functions and the strategy are application independent (in the class of database interfaces) and can be easily customized. In fact the same architecture has been used for prototypical implementations of the ATIS application, a conference room reservation, a restaurant and a movie information task. We believe that this approach can scale up to more complex applications and exhibit more complex behavior.

REFERENCES

[1] Kumar, V., "Algorithms for constraint-satisfaction problems: A survey," *AI Magazine*, 13/1, pp.32-44, 1992.

[2] Stallard, D., " The BBN ATIS4 Dialogue System," *Proc. of 1995 ARPA Spoken Language Systems,* Technology Workshop, Austin Texas, Jan. 1995.

[3] Levin, E., Pieraccini, R., " Concept-based Spontaneous Speech Understanding System," *Proc. EUROSPEECH 95*, Madrid, September 1995.

[4] Abella, A., Brown, M., Buntschuh, B., "Development Principles for Dialogue-based Interfaces," *Proc. of ECAI-96 Workshop on Dialogue Processing in Spoken Language Systems*, Budapest, Hungary, August 1996.

[5] M.D. Sadek et. al. "Effective Human-Computer Cooperative Spoken Dialogue: the AGS Demonstrator," in *Proc. of ICSLP 96*, Philadelphia, PA, Oct. 96.

[6] E.Levin, R.Pieraccini, " A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies," *Proc. of Eurospeech 97*, Rhodes, Greece, Sept. 1997.

1	VERBALIZER	This is the AT&T Chronus Air Travel Information System, how may I help you?
2	UNDERSTANDING	**I want a direct flight from Denver to Boston between eight and nine thirty tomorrow morning**
3	RETRIEVAL RELAX VERBALIZER	There are no flights from Denver to Boston leaving between 8 and 9:30 in the morning. Instead there are flights leaving at 10:39 and 11:05. Do you have any preference?
4	UNDERSTANDING	**Give me a delta flight**
5	DISCOURSE INT. RETRIEVAL VERBALIZER	Flight DL296 leaves at 10:39 am and arrives at 3:15 pm. Other questions?
6	UNDERSTANDING	**I need ground transportation**
7	DISCOURSE INT. VERBALIZER	Where, Denver or Boston?
8	UNDERSTANDING	**Boston**
9	DIALOGUE INT. RETRIEVAL VERBALIZER	The following ground transportation is available in Boston: taxi, limousine, rental car.
10	UNDERSTANDING	**Please list all the flights with Continental**
11	DISCOURSE RETRIEVAL CONSTRAIN	There are too many flights that match your request. I need more information. At what time you want to leave?

Table 1: Fragment of dialogue with the corresponding control states.