Managing Spoken Dialogues for Information Services*

Wieland Eckert
Universität Erlangen-Nürnberg
Lehrstuhl für Mustererkennung (Informatik 5)
Martensstraße 3, 91058 Erlangen, F.R. Germany
wieland.eckert@informatik.uni-erlangen.de

Scott McGlashan
Social and Computer Sciences Research Group
Department of Sociology
University of Surrey, Guildford, UK.
scott@soc.surrey.ac.uk

Abstract

This paper presents an approach to managing spoken dialogues in information services systems. We describe how the approach, based upon a tri–partite model of interaction, addresses the problems of co–operativeness and portability across languages and task domains. This approach has been implemented in the generic dialogue manager of the SUNDIAL dialogue systems. We outline the capabilities of this dialogue manager and describe our testing methodology.

KEYWORDS: Oral Dialogue, Dialogue Management, Interpretation of Utterances, Cooperative System

1 Introduction

Management of spoken dialogues requires the interpretation of the user utterances and the generation of system utterances coherent with user utterances. The underlying model of interaction is partitioned into semantic, task and dialogue models (cf. [4]). Utterances in different languages are mapped into a common semantic representation language from which instances of the semantic, task and dialogue models are constructed.

Semantic interpretation of user utterances is performed against a semantic model of the interaction and results in the instantiation of task concepts (cf. [6]). Task interpretation determines whether the user has provided sufficient information for retrieval of a solution from a database, or whether further information is required. Dialogue interpretation evaluates the semantic interpretation against a structured dialogue model (cf. [1]). This results in the generation of system utterances which confirm information extracted from the user utterance, request further information, or provide a solution to the user request. Using this tri–partite interactional model, dialogue management is co–operative and independent of the task and language of the service domain.

This approach has been implemented as a generic dialogue manager component in SUNDIAL dialogue systems (cf. [8]). The generic dialogue manager is designed to operate language independently in a wide variety of information dialogues. Prototypes have been built for French, English, German and Italian, and service domains include flight and train inquiries. Features of this dialogue manager include the ability to handle the specification of an enquiry over multiple user utterances, negation and modification of previous input, complex temporal reasoning, constraint relaxation and the ability to dynamically switch the dialogue strategy so as to minimize the risk of breakdown (cf. [3]). Currently, systems are being evaluated with respect to performance metrics such as robustness, accuracy of interpretation, and appropriacy of system response (cf. [10]). Evaluation of one SUNDIAL system is reported in [2].

2 The SUNDIAL System

Since an overall description of the system has already been presented in [9], we only provide a brief overview. The architecture, adopted by all partners in the project, is illustrated in Figure 1. Speech input from a microphone or telephone line is processed in the acoustic front end, operating on hidden markov models. Different recognizers are running either on specialized hardware or general purpose workstations, with varying lexicon sizes between 600 and 1100 words. To improve recognizer accuracy, nearly all partners use techniques which dynamically reduce the search space, such as stochastic bigram models correlated with dialogue states. The acoustic front end delivers either a best string or a word lattice to a parser for linguistic analysis. Various parsing methods and strategies are used to construct syntactic and semantic representations of user utterances. In the majority of systems, the semantic representations are couched in SIL, a knowledge representation language designed to provide a language and domain independent representation of parsing results. The dialogue manager then interprets this representation using a tri-partite model of interaction, and plans a system utterance for generation and synthesis. Since its behaviour can be customized for different task domains, the same dialogue manager can be used in each SUNDIAL system.

3 Dialogue Management

The major goal of dialogue management is to manage the interaction in a co–operative manner. To achieve this, the dialogue manager must handle a number of phenomena observed in human–human and (simulated) human-computer dialogues such as: anaphora and ellipses, ambiguity, topic-shifting, turn taking, strategies for dealing with communication failure, implicit and explicit confir-

*This work was partly funded by the Commission of the European Community under ESPRIT contract P 2218 (SUNDIAL). The authors are solely responsible for the views expressed in this paper.

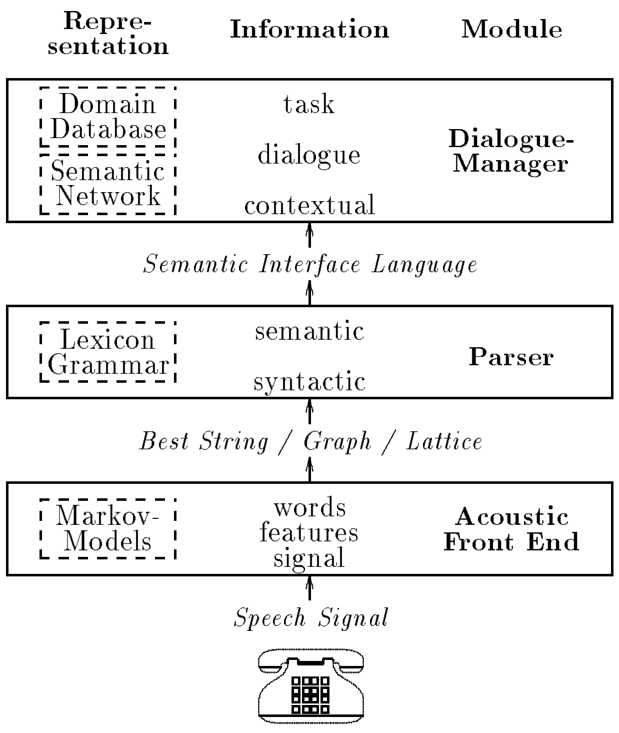

Figure 1: Processing Stages in SUNDIAL

mation, and appropriate answers when no database solution matches the caller's enquiry.

The design and implementation of the generic SUNDIAL dialogue manager is based on the architecture in Figure 2. Dialogue management functions are distributed across five independent modules, each with its own knowledge bases, which communicate by message passing and operate by interleaved control. The core functions are concentrated in three principal modules:

Belief Module (BM) semantic interpretation of user utterances

Task Module (TM) task interpretation relative to application goals

Dialogue Module (DM) pragmatic interpretation and planning of system utterances

These are described in more detail below.

The remaining modules perform peripheral functions. The Linguistic Interface (LI) is responsible for the proper information transfer to and from the parser. The LI sends predictions to guide recognition and parsing of the next user utterance, and in return receives a SIL structure for interpretation. The Message Planner (MP) is responsible for communication with the generation sub–system. It receives a plan of the system utterance from the DM and maps it into a structure suitable for generation with either a template-based or rule-based message generator. Additionally, the LI store system and user utterances in a linguistic history which can be used for anaphora resolution.

3.1 Semantic Interpretation

The belief module performs semantic interpretation by using the semantic description of user utterances to build an extension of its contextual model. The contextual model is a semantic network constructed from two declarative knowledge sources. The first is a hierarchy of surface–oriented concepts, shared with the parsing component. The second source is a hierarchy of task–oriented concepts, known to the task module, which correspond to relations in the application database. When instances of surface–oriented concepts are created which reference task–oriented concepts, instances of these later concepts are also created in the contextual model. This mapping between surface–oriented and task–oriented concepts is guided by inference rules associated with particular types of concepts.

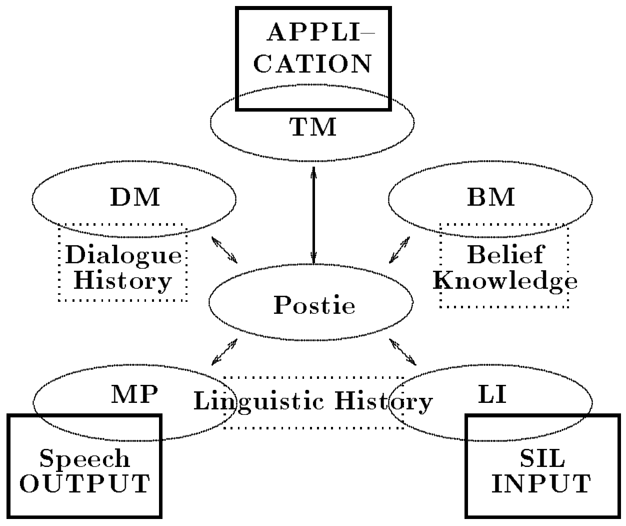

Figure 2: Architecture of the Dialogue Manager

For example, the user utterance *ich möchte nach Ulm fahren* is represented using the surface-oriented concept *go*. An inference rule, shown in (1), attached to this concept maps the value of its *thegoal* role (*ulm*) into value for the arrival place of its *journey* role.

(1) *go*: <thegoal> = <thejourney thearrival theplace>

(2) *journey*: <thearrival theplace thecity value> = <dbtrain goalcity>

(3) [id: dbtrain1, type:dbtrain, goalcity:ulm]

The second rule, (2), attached to the *journey* concept then maps the value of the arrival city into the *goalcity* attribute of the task concept *dbtrain* shown in (3).

The semantic representation of user utterances can be underspecified. This can be the case with deictic expressions, both temporal (*dem nächsten Zug nach München*) and spatial (*a train from there to Berlin*), anaphoric expressions (*does it call at Düsseldorf*) and elliptic (*nach Ulm*) expression. In the course of instantiation the BM tries to link these partially specified inputs to a concept in its contextual model. In order to decide which concept to use, the BM constructs a set of possible linking concepts on the basis of recency and accessibility information as well as the concepts underlying the last system utterance. From this set it identifies candidates for linking with the input. For example, in the case of ellipsis the candidate that matches the semantic role of the ellipsed concept is selected.

Some user utterances are not consistent with accessible concepts in the current contextual model. This situation arises in cases such as:

(4) S: *Sie wollen von Münster nach Bremen fahren*
C: *nicht nach Bremen, nach Ulm*

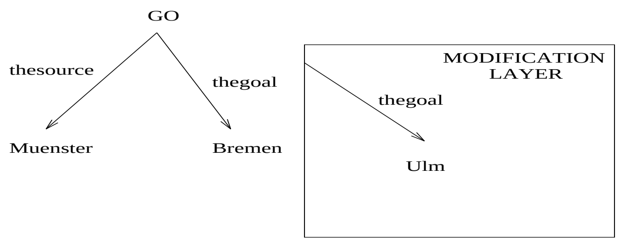

Figure 3: BM model after the input: *nicht nach Bremen, nach Ulm*

where the system has misinterpreted the arrival city as *Bremen* and the user has corrected it to *Ulm*. To deal with this, the current contextual model is overlaid with a new conceptual layer which specifies the information inconsistent with the existing layer but inherits all other information from it. With (4), the new layer would contain the information that arrival city is *Ulm* but inherit the information that the departure city is *Münster*. This is illustrated in Figure 3.

3.2 Task Interpretation

The task module is responsible for providing an interpretation of a task concept with respect to its model of the task. The TM's declarative component specifies for each task type, a set of dependent task parameters together with necessary and default inference rules. For example, the task type *dbtrain* has the dependent parameters *sourcecity*, *goalcity*, *sourcetime* and *date*. During a dialogue, the TM consults this component to determine which parameters are required. In (4) the task model is updated with the information that the task concept *dbtrain* is instantiated with values for the parameter *sourcecity* and *goalcity* (where the value *ulm* overrides the value *bremen*). Since the parameter *sourcetime* is not instantiated, TM's interpretation is that a value for this parameter is required. Given that a default rule cannot be used — unlike *date* where application of a default rule results in today's date being used — the TM informs the DM of the required parameter and the DM, in turn, requests it from the user. Once the task model is instantiated with values for all dependent parameters, the appropriate application database is consulted and a solution presented to the user. In this way, the TM proposes the general direction of the whole dialogue by successively requesting values for parameters and then providing a solution.

One central aspect of co–operativeness is embodied in the TM. A co–operative system provides *corrective* and *suggestive* answers when no solution in the application database match the user's requirements (cf. [5]). Rather than simply signal failure, the system can indicate the reason for failure and/or make modifications which may lead to a solution. The latter can be achieved by constraint relaxation: constraints, i.e. parameter values, can be systematically modified until a solution is found. For example, constraint relaxation can be applied to the departure time parameter. If the user requests a train from Münster to Ulm leaving at 5pm, but no solution can be found, then the time can be incremented — 5 01, 5 02 and so on — until either a threshold is encountered or a solution recovered.

3.3 Pragmatic Interpretation

The dialogue module is responsible for the pragmatic interpretation of user input and planning of the system response.

Pragmatic interpretation is determined by a declarative component which specifies rules for updating the dialogue model on the basis of the semantic and pragmatic interpretation of user input. The dialogue model is composed of a set of dialogue flags controlling, for example, how parameters are confirmed, and a set of dialogue goals representing the system's intentions.

The semantic interpretation is used by the DM to evaluate the set of active dialogue goals. Satisfied goals are assigned the pragmatic status *succeed*, other goals are assigned the status *fail*. For example, in (4) the system utterance realizes two *confirm* goals with the parameters *sourcecity:muenster* and *goalcity:bremen* respectively. The user's response is a re–specification of the *goalcity* as *Ulm*. In this instance, application of the semantic interpretation update rules results in both goal being assigned the pragmatic status *succeed*, but, as a side-effect of re–specification, a new goal confirming the modified parameter *goalcity* is added to the dialogue model.

When the DM receives a task interpretation from the TM, it creates an appropriate goal in the dialogue model. For example, if the TM requires a value for the parameter *sourcetime*, a *request* goal will be added. The DM then decides which goals should be realized in the next system turn. This decision is influenced by the confirmation flag. For example, if the model contains a *request* and *confirm* goals, then both will be realized if the flag is set to *confirm-plus-initiative* but only the *confirm* goal if the flag is set to *confirm-alone*. When goals are realized, they become the active goals used to evaluate the user's response.

Since the DM's interpretation rules take into account the value of repair threshold flag, the dialogue strategy is responsive to communicative difficulties. For example, if repetition of an initial request goal exceeds the threshold, the system switches to a menu–style interaction:

(5) S: *Flight Inquiries. How can I help you?*
C: *My boss told me to book a flight from London to Paris.*
S: *I am sorry I didn't understand you. Tell me your enquiry.*
C: *erhm,...*
S: *I still didn't understand you. Please tell me the departure city.*

And in situations where the user's response to a *request* goal repeatedly results in a *fail* status being assigned, the goal is replaced with a *spell* goal and the user is asked to spell the parameter.

4 Testing

As shown in Figure 2, the dialogue manager is constructed from a number of modules. Since these were developed by partners on different sites, a method for testing modules developed on different sites had to be devised.

The *testfile method* exploits the centralized message passing of the dialogue manager architecture. Since interactions between modules are mediated by a central component (the 'Postie'), it is possible to define the exact sequence and content of messages required for the correct behaviour of the dialogue manager in a given dialogue. Furthermore, the behaviour of each module can be defined in terms of a sequence of received and transmitted messages for a set of test dialogues. Application of this method ensured that modules conforming to the defined behaviour would fit together, yet each module could be developed and tested independently.

The testfiles were initially constructed manually, a time consuming and error prone task, from a selection of dialogues gathered from Wizard of Oz simulations (cf. [7]). Once the system became operational, a large set of testfiles was created through automatic logging. Consequently, any modifications to a module could be checked against the behaviour defined by the test dialogues and misbehaving modules easily identified. Once the modules became relatively robust, this exhaustive definition of behaviour was superseded by one based upon input from the parser: i.e. a dialogue was specified in terms of a sequence of SIL representations of user input.

When each version of the dialogue manager is integrated with the rest of the system and evaluated with users, the logged results are examined for phenomena not yet described in the testfiles. New phenomena are specified using a testfile and the extended set of testfiles define the intended functionality of the next version. By adopting this technique, we have been able to chart the progress of the dialogue manager in terms of the number of correctly processed testfiles as shown in Figure 4.

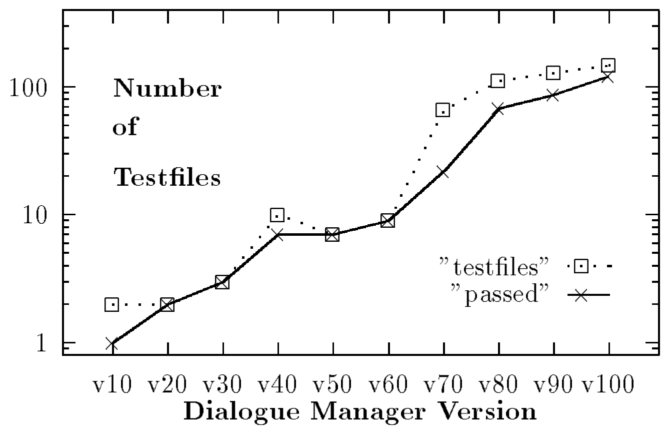

Figure 4: Dialogue Manager Progress

One additional benefit of the testfile method is that it provides a simple way of evaluating the dialogue manager. The metrics used to evaluate the whole dialogue system, while important performance ratings for components, are not able to evaluate the dialogue manager in terms of the range of dialogue phenomena it is capable to handle correctly. On the other hand, the testfiles define a reference set of dialogue phenomena which can be used in this manner. Some of the phenomena which the current version is capable of handling correctly are listed in Table 1.

5 Conclusion

We have described an approach to dialogue management which addresses the problems of co–operativeness and portability across tasks and languages. The approach, based upon a tri-partite model of interaction, has been adopted in the generic dialogue manager of the SUNDIAL systems. Three independent modules are responsible for the contextual interpretation of user utterances, the guidance of dialogues according to conversational principles and task requirements. Prototype systems utilizing this dialogue manager are operational in different languages, with different tasks and domains. Flexible dialogue strategies (including degradation to spelling of city names) control the dialogue progress and are adapted to the current recognition performance. The SUNDIAL dialogue manager is Europe's first multilingual, task independent dialogue manager operational in four languages.

Context–dependent interpretation of utterances
Processing ellipses
Information transfer in a single user utterance
Information transfer in multiple user utterances
User can modify previous input
User can negate previous input
Complex temporal reasoning:
approximate times, intervals, temporal deixis
Overinformative answers from the user
Several phrases as input
Repetition as confirmation
Different dialogue modes (guided, spelling)
Switching modes dynamically
Different parameter confirmation strategies
Initial WH–questions
Initial Yes/No–questions

Table 1: Some phenomena handled by the Dialogue Manager

6 Acknowledgments

Partners in this project are CAP Gemini Innovation, CNET, CSELT, Daimler–Benz, University of Erlangen, Infovox, IRISA, LOGICA, Politechnico di Torino, Sarin, Siemens, University of Surrey. We wish to thank all the people who participated in fruitful work of Work Package 6 (Dialogue Management).

References

[1] E. Bilange. A task independent oral dialogue model. In *Proceedings of the 5th EACL*, Berlin, Germany, 1991.

[2] W. Eckert, T. Kuhn, H. Niemann, S. Rieck, A. Scheuer, and E.G. Schukat-Talamazzini. A Spoken Dialogue System for German Intercity Train Timetable Enquiries. In *Proc. European Conf. on Speech Technology*, Berlin, Germany, 1993 (this issue).

[3] C.R. Frankish. Conversations with computers: problems of feedback and error correction. In *Proc. European Conf. on Speech Technology*, Paris, 1989.

[4] B. Grosz and C. Sidner. Attention, Intensions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, July–September 1986.

[5] M. Guyomard and J. Siroux. Suggestive and corrctive answers: A single mechanism. In *Proc. Workshop The Structure of Multimodal Dialogue Including Voice (held 1986 in Venaco)*, 1989.

[6] P. Heisterkamp, S. McGlashan, and N.J. Youd. Dialogue Semantics for a Spoken Dialogue System. In *Proc. Int. Conf. on Spoken Language Processing*, October 1992. Banff, Canada.

[7] C. MacDermid. Features of Naive Caller's Dialogues with a Simulated Speech Understanding and Dialogue System. In *Proc. European Conf. on Speech Technology*, Berlin, Germany, 1993 (this issue).

[8] S. McGlashan, N.M. Fraser, N. Gilbert, E. Bilange, P. Heisterkamp, and N.J. Youd. Dialogue Management for Telephone Information Services. In *Proceedings of the International Conference on Applied Language Processing*, Trento, Italy, 1992.

[9] J. Peckham. Speech Understanding and Dialogue over the Telephone: an Overview of Progress in the SUNDIAL Project. In *Proc. European Conf. on Speech Technology*, volume 3, pages 1469–1472, 1991.

[10] A. Simpson and N. Fraser. Black Box and Glass Box Evaluation of the SUNDIAL System In *Proc. European Conf. on Speech Technology*, Berlin, Germany, 1993 (this issue).