Adaptive Dialogue Systems - Interaction with Interact

Kristiina Jokinen and **Antti Kerminen** and **Mauri Kaipainen**
Media Lab, University of Art and Design Helsinki
Hämeentie 135 C, FIN-00560 Helsinki, Finland
{kjokinen|akermine|mkaipain}@uiah.fi

Tommi Jauhiainen and **Graham Wilcock**
Department of General Linguistics, University of Helsinki
FIN-00014 University of Helsinki, Finland
{tsjauhia|gwilcock}@ling.helsinki.fi

Markku Turunen and **Jaakko Hakulinen**
TAUCHI Unit, University of Tampere
FIN-33014 University of Tampere, Finland
{mturunen|jh}@cs.uta.fi

Jukka Kuusisto and **Krista Lagus**
Neural Networks Research Centre, Helsinki University of Technology
P.O.9800 FIN-02015 HUT, Finland
{krista|jkuusist}@james.hut.fi

Abstract

Technological development has made computer interaction more common and also commercially feasible, and the number of interactive systems has grown rapidly. At the same time, the systems should be able to adapt to various situations and various users, so as to provide the most efficient and helpful mode of interaction. The aim of the Interact project is to explore natural human-computer interaction and to develop dialogue models which will allow users to interact with the computer in a natural and robust way. The paper describes the innovative goals of the project and presents ways that the Interact system supports adaptivity on different system design and interaction management levels.

1 Introduction

The need for flexible interaction is apparent not only in everyday computer use, but also in various situations and services where interactive systems can diminish routine work on the part of the service provider, and also cater for the users with fast and tailored access to digital information (call centers, help systems, interactive banking and booking facilities, routing systems, information retrieval, etc.).

The innovative goal of the Finnish Interact project is to enable natural language interaction in a wider range of situations than has been possible so far, and in situations where its use has not been functional or robust enough. This means that the systems should support rich interaction and also be able to learn and adapt their functionality to the changing situation. It also implies that the needs of special groups will be taken into account when designing more natural interactive systems. Within the current system, such scenarios can e.g. include an intelligent bus-stop which allows spoken and text interaction concerning city transportation, with a sign language help facility.

The project addresses especially the problem of adaptivity: the users are situated in mobile environments in which their needs, activities and abilities vary. To allow the users to express their wishes in a way characteristic to them and

the situation, interaction with the system should take place in a robust and efficient manner, enabling rich and flexible communication. Natural language is thus the preferred mode of interaction, compared to graphical interfaces for example. Adaptivity also appears in the techniques and methods used in the modelling of the interaction and the system's processing capabilities. An important aspect in this respect is to combine machine learning techniques with rule-based natural language processing, to investigate limitations and advantages of the two approaches for language technology.

In this paper we focus on adaptivity which manifests itself in various system properties:

- agent-based architecture
- natural language capability
- self-organising topic recognition
- conversational ability.

The paper is organized as follows. We first introduce the dialogue system architecture. We then explain how the modules function and address the specific design decisions that contribute to the system's adaptivity. We conclude by discussing the system's capabilities and providing pointers for future work.

2 Agent-based architecture

To allow system development with reusable modules, flexible application building and easy combination of different techniques, the framework must itself be designed specifically to support adaptivity. We argue in favour of a system architecture using highly specialized agents, and use the Jaspis adaptive speech application framework (Turunen and Hakulinen, 2000; Turunen and Hakulinen, 2001a). Compared to e.g. Galaxy (Seneff et al., 1998), the system supports more flexible component communication. The system is depicted in Figure 1.

2.1 Information Storage

The Jaspis architecture contains several features which support adaptive applications. First of all, the information about the system state is kept in a shared knowledge base called Information Storage. This blackboard-type information storage can be accessed by each system component via the Information Manager, which allows them to utilize all the information that the system contains, such as dialogue history and user profiles, directly. Since the important information is kept in a shared place, system components can be stateless, and the system can switch between them dynamically. Information Storage thus facilitates the system's adaptation to different internal situations, and it also enables the most suitable component to be chosen to handle each situation.

2.2 Flexible Component Management

The system is organized into modules which contain three kinds of components: managers, agents and evaluators. Each module contains one *manager* which co-ordinates component interaction inside the module. The present architecture implements e.g. the Input/Output Manager, the Dialogue Manager and the Presentation Manager, and they have different priorities which allow them to react to the interaction flow differently. The basic principle is that whenever a manager stops processing, all managers can react to the situation, and based on their priorities, one of them is selected. There is also the Interaction Manager which coordinates applications on the most general level.

The number and type of modules that can be connected to the system is not limited. The Interaction Manager handles all the connections between modules and the system can be distributed for multiple computers. In Interact we have built a demonstration application on bus-timetable information which runs on several platforms using different operating systems and programming languages. This makes the system highly modular and allows experiments with different approaches from multiple disciplines.

2.3 Interaction Agents and Evaluators

Inside the modules, there are several agents which handle various interaction situations such as speech output presentations and dialogue decisions. These *interaction agents* can be very

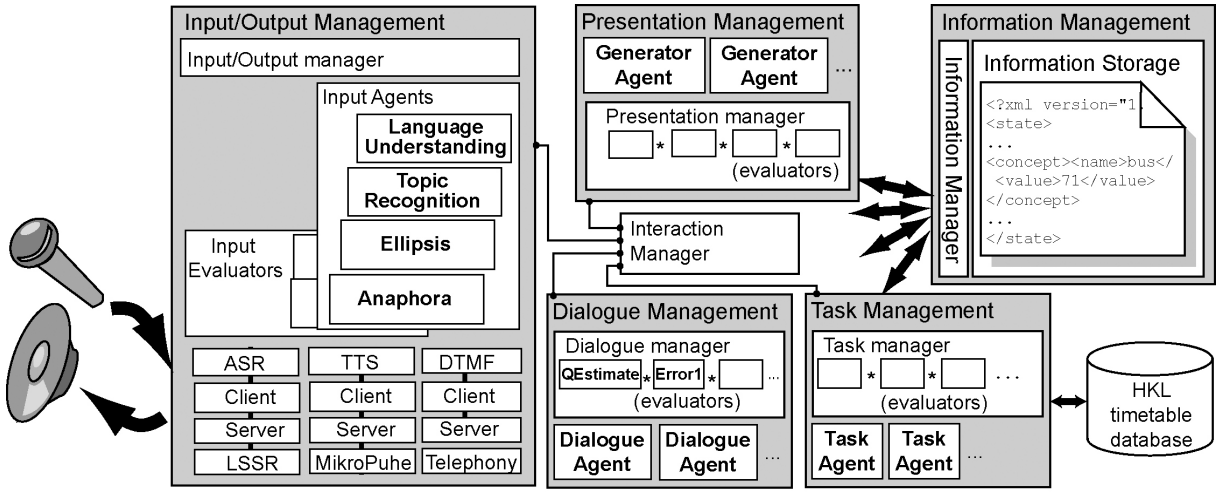

Figure 1: The system architecture.

specialized, e.g. they deal only with speech recognition errors or outputs related to greetings. They can also be used to model different interaction strategies for the same task, e.g. different dialogue agents can implement alternative dialogue strategies and control techniques. Using specialized agents it is possible to construct modular, reusable and extendable interaction components that are easy to implement and maintain. For example, different error handling methods can be included to the system by constructing new agents which handle errors using alternative approaches. Similarly, we can support multilingual outputs by constructing presentation agents that incorporate language specific features for each language, while implementing general interaction techniques, such as error correction methods, to take care of error situations in speech applications in general (Turunen and Hakulinen, 2001b).

The agents have different capabilities and the appropriate agent to handle a particular situation at hand is selected dynamically based on the context. The choice is done using *evaluators* which determine applicability of the agents to various interaction situations. Each evaluator gives a score for every agent, using a scale between [0,1]. Zero means that an agent is not suitable for the situation, one means that an agent is perfectly suitable for the situation, values between zero and one indicate the level of suitability. Scaling functions can be used to emphasize certain evaluators over the others The scores are then multiplied, and the final score, a suitability factor, is given for every agent. Since scores are multiplied, an agent which receives zero from one evaluator is useless for that situation. It is possible to use different approaches in the evaluation of the agents, and for instance, the dialogue evaluators are based on reinforcement learning.

Simple examples of evaluators are for instance presentation evaluators that select presentation agents to generate suitable implicit or explicit confirmations based on the dialogue history and the system's knowledge of the user. Another example concerns dialogue strategies: the evaluators may give better scores for system-initiative agents if the dialogue is not proceeding well with the user-initiative dialogue style, or the evaluators may prefer presentation agents which give more detailed and helpful information, if the users seem to have problems in communicating with the application.

Different evaluators evaluate different aspects of interaction, and this makes the evaluation process highly adaptive itself: there is no single evaluator which makes the final decision. Instead, the choice of the appropriate interaction agent is a combination of different evaluations.

Evaluators have access to all information in the Information Storage, for example dialogue history and other contextual information, and it is also possible to use different approaches in the evaluation of the agents (such as rule-based and statistical approaches). Evaluators are the key concept when considering the whole system and its adaptation to various interaction situations.

2.4 Distributed Input and Output

The input/output subsystem is also distributed which makes it possible to use several input and output devices for the same purposes. For example, we can use several speech recognition engines, each of which with different capabilities, to adapt the system to the user's way of talking. The system architecture contains virtual devices which abstract the actual devices, such as speech recognizers and speech synthesizers. From the application developers viewpoint this makes it easy to experiment with different modalities, since special agents are used to add and interpret modality specific features. It is also used for multilingual inputs and outputs, although the Interact project focuses on Finnish speech applications.

3 Natural Language Capabilities

The use of Finnish as an interaction language brings special problems for the system's natural language understanding component. The extreme multiplicity of word forms prevents the use of all-including dictionaries. For instance, a Finnish noun can theoretically have around 2200, and a verb around 12000 different forms (Karlsson, 1983). In spoken language these numbers are further increased as all the different ways to pronounce any given word come into consideration (Jauhiainen, 2001). Our dialogue system is designed to understand both written and spoken input.

3.1 Written and spoken input

The different word forms are analyzed using Fintwol, the two-level morphological analyzer for Finnish (Koskenniemi, 1983). The forms are currently input to the syntactic parser CPARSE (Carlson, 2001). However, the flexible system architecture also allows us to experiment with different morphosyntactic analyzers, such as TextMorfo (Kielikone Oy 1999) and Conexor FDG (Conexor Oy 1997-2000), and we plan to run them in parallel as separate competing agents to test and compare their applicability as well as the Jaspis architecture in the given task.

We use the Lingsoft Speech Recognizer for the spoken language input. The current state of the Finnish speech recognizer forces us to limit the user's spoken input to rather restricted vocabulary and utterance structure, compared to the unlimited written input. The system uses full word lists which include all the morphological forms that are to be recognized, and a strict context-free grammar which dictates all the possible utterance structures. We are currently exploring possibilities for a HMM-based language model, with the conditional probabilities determined by a trigram backoff model.

3.2 Language analysis

The task of the parsing component is to map the speaker utterances into task-relevant domain concepts which are to be processed by the dialogue manager. The number of domain concepts concerning the demonstration system's application domain, bus-timetables, is rather small and contains e.g. `bus`, `departure-time` and `arrival-location`. However, semantically equivalent utterances can of course vary in the lexical elements they contain, and in written and especially in spoken Finnish the word order in almost any given sentence can also be changed without major changes on the semantic level understood by the system (the difference lies in the information structure of the utterance). For instance, the request *How does one get to Malmi?* can be realised as given in Table 1.

There are two ways to approach the problem: on one hand we can concentrate on finding the keywords and their relevant word forms, on the other hand we can use more specialized syntactic analyzers. At the moment we use CPARSE as the syntactic analyzer for text-based input. The grammar has been adjusted for the demon-

Kuinka pääsee bussilla Malmille?
Miten pääsee Malmille bussilla?
Kuinka Malmille pääsee bussilla?
Malmille miten pääsee bussilla?
Millä bussilla pääse Malmille?
Malmille olisin bussia kysellyt.
Pääseekö bussilla Malmille?

Table 1: Some alternative utterances for *Kuinka pääsee Malmille bussilla? 'How does-one-get to-Malmi by bus?*

stration system so that it especially looks for phrases relevant to the task at hand. For instance, if we can correctly identify the inflected word form *Malmille* from the input string, we can be quite certain of the user wishing to know something about getting to Malmi.

The current speech input does not go through any special morpho-syntactic analysis because of the strict context-free grammar used by the speech recognizer. The dictionary used by the recognizer is tagged with the needed morphological information and the context-free rules are tagged with the needed syntactic information.

3.3 Language generation

The language generation function is located in the system's Presentation Manager module. Unlike language analysis, for which different existing Finnish morphosyntactic analyzers can be used, there are no readily available general-purpose Finnish language generators. We are therefore developing specific generation components for this project. The flexible system architecture allows us to experiment with different generators.

Unfortunately the existing Finnish syntactic analyzers have been designed from the outset as "parsing grammars", which are difficult or impossible to use for generation. However, the two-level morphology model (Koskenniemi, 1983) is in principle bi-directional, and we are working towards its use in morphological generation. Fortunately there is also an existing Finnish speech synthesis project (Vainio, 2001), which we can use together with the language generators.

Some of our language generation components use the XML-based generation framework described by Wilcock (2001), which has the advantage of integrating well with the XML-based system architecture. The generator starts from an *agenda* which is created by the dialogue manager, and is available in the system's Information Storage in XML format. The agenda contains a list of semantic concepts which the dialogue manager has tagged as Topic or NewInfo. From the agenda the generator creates a response plan, which passes through the generation pipeline stages for lexicalization, aggregation, referring expressions, syntactic and morphological realization. At all stages the response specification is XML-based, including the final speech markup language which is passed to the speech synthesizer.

The system architecture allows multiple generators to be used. In addition to the XML-based pipeline components we have some pregenerated outputs, such as greetings at the start and end of the dialogue or meta-acts such as wait-requests and thanking. We are also exploiting the agent-based architecture to increase the system's adaptivity in response generation, using the level of communicative confidence as described by Jokinen and Wilcock (2001).

4 Recognition of Discussion Topic

One of the important aspects of the system's adaptivity is that it can recognize the correct topic that the user wants to talk about. By 'topic' we refer to the general subject matter that a dialogue is about, such as 'bus timetables' and 'bus tickets', realized by particular words in the utterances. In this sense, individual documents or short conversations may be seen to have one or a small number of topics, one at a time.

4.1 Topically ordered semantic space

Collections of short documents, such as newspaper articles, scientific abstracts and the like, can be automatically organized onto *document maps* utilizing the Self-Organizing Map algorithm (Kohonen, 1995). The document map

methodology has been developed in the WEBSOM project (Kohonen et al., 2000), where the largest map organized consisted of nearly 7 million patent abstracts.

We have applied the method to dialogue topic recognition by carring out experiments on 57 Finnish dialogues, recorded from the customer service phone line of Helsinki City Transport and transcribed manually into text. The dialogues are first split into topically coherent segments (utterances or longer segments), and then organized on a document map. On the ordered map, each dialogue segment is found in a specific map location, and topically similar dialogue segments are found near it. The document map thus forms a kind of *topically ordered semantic space*. A new dialogue segment, either an utterance or a longer history, can likewise be automatically positioned on the map. The coordinates of the best-matching map unit may then be considered as a latent topical representation for the dialogue segment.

Furthermore, the map units can be labeled using named topic classes such as 'timetables' and 'tickets'. One can then estimate the probability of a named topic class for a new dialogue segment by construing a probability model defined on top of the map. A detailed description of the experiments as well as results can be found in (Lagus and Kuusisto, 2002).

4.2 Topic recognition module

The topical semantic representation, i.e. the map coordinates, can be used as input for the dialogue manager, as one of the values of the current dialogue state. The system architecture thus integrates a special topic recognition module that outputs the utterance topic in the Information Storage. For a given text segment, say, the recognition result from the speech recognizer, the module returns the coordinates of the best-matching dialogue map unit as well as the most probable prior topic category (if prior categorization was used in labeling the map).

5 Dialogue Management

The main task of the dialogue manager component is to decide on the appropriate way to react to the user input. The reasoning includes recognition of communicative intentions behind the user's utterances as well as planning of the system's next action, whether this is information retrieval from a database or a question to clarify an insufficiently specified request. Natural interaction with the user also means that the system should not produce relevant responses only in terms of correct database facts but also in terms of rational and cooperative reactions. The system could learn suitable interaction strategies from its interaction with the user, showing adaptation to various user habits and situations.

5.1 Constructive Dialogue Model

A uniform basis for dialogue management can be found in the communicative principles related to human rational and coordinated interaction (Allwood et al., 2000; Jokinen, 1996). The speakers are engaged in a particular activity, they have a certain role in that activity, and their actions are constrained by communicative obligations. They act by exchanging new information and constructing a shared context in which to resolve the underlying task satisfactorily.

The model consists of a set of dialogue states, defined with the help of dialogue acts, observations of the context, and reinforcement values. Each action results in a new dialogue state. The dialogue act, *Dact*, describes the act that the speaker performs by a particular utterance, while the topic *Top* and new information *NewInfo* denote the semantic content of the utterance and are related to the task domain. Together these three create a useful first approximation of the utterance meaning by abstracting over possible linguistic realisations. Unfilled task goals *TGoals* keep track of the activity related information still necessary to fulfil the underlying task (a kind of plan), and the speaker information is needed to link the state to possible speaker characteristics. The expectations, *Expect* are related to communicative obligations, and used to constrain possible interpretations of the next act. Consequently, the system's internal states can be reduced to a combination of these categories, all of which form an indepen-

dent source of information for the system to decide on the next move.

5.2 Dialogue agents and evaluators

A dialogue state and all agents that contribute to a dialogue state are shown in Figure 2. The Dialogue Model is used to classify the current utterance into one of the dialogue act categories (Jokinen et al., 2001), and to predict the next dialogue acts (Expect). The Topic Model recognizes the domain, or discussion topic, of the user input as described above.

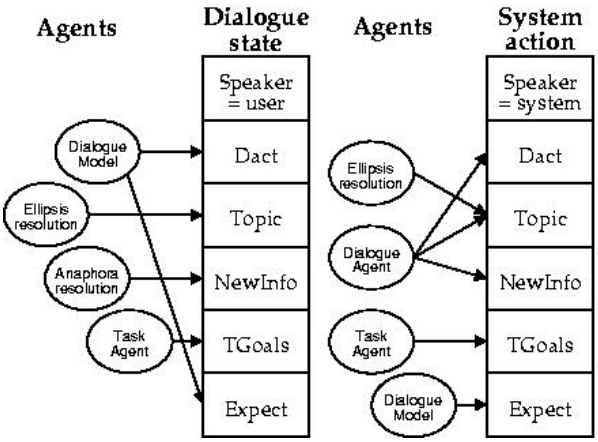

Figure 2: Dialogue states for user's utterance and system action, together with dialogue agents involved in producing various information.

All domains out of the system's capabilities are handled with the help of a special OutOfDomain-agent which informs the user of the relevant tasks and possible topics directly. This allows the system to deal with error situations, such as irrelevant user utterances, efficiently and flexibly without invoking the Dialogue Manager to evaluate appropriate dialogue strategies. The information about error situations and the selected system action is still available for dialogue and task goal management through the shared Information Storage.

The utterance Topic and New Information (Topic, NewInfo) of the relevant user utterances are given by the parsing unit, and supplemented with discourse knowledge by ellipsis and anaphora resolution agents (which are Input Agents). Task related goals are produced by Task Agents, located in a separate Task Manager module. They also access the backend database, the public transportation timetables of Helsinki.

The Dialogue Manager (DM) consists of agents corresponding to possible system actions (Figure 3). There are also some agents for internal system interaction, illustrated in the figure with a stack of agents labeled with *Agent1*. One agent is selected at a time, and the architecture permits us to experiment with various competing agents for the same subtask: the evaluators are responsible for choosing the one that best fits in the particular situation.

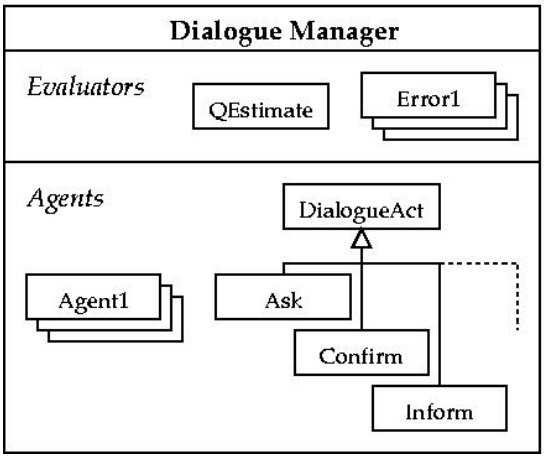

Figure 3: The Dialogue Manager component.

Two types of evaluators are responsible for choosing the agent in DM, and thus implementing the dialogue strategy. The QEstimate evaluator chooses the agent that has proven to be most rewarding so far, according to a Q-learning (Watkins and Dayan, 1992) algorithm with online ϵ-greedy policy (Sutton and Barto, 1998). That agent is used in the normal case and the decision is based on the dialogue state presented in Figure 2. The underlying structure of the QEstimate evaluator is illustrated in Figure 4.

The evaluator is based on a table of real values, indexed by dialogue states, and updated after each dialogue. The agent with the highest

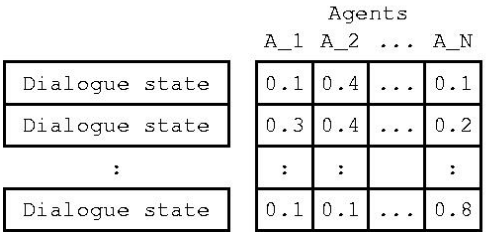

Figure 4: The QEstimate evaluator.

value for the current dialogue state gets selected. Adaptivity of the dialogue management comes from the reinforcement learning algorithm of this evaluator.

On the other hand, if one of the error evaluators (labeled with `Error1..N`) detects that an error has occurred, the QEstimate evaluator is overridden and a predetermined agent is selected to handle the error situation (Figure 5). In these cases, only the the correct agent is given a non-zero value, forcing the dialogue manager to select that agent. Examples of such errors include situations when the user utterance is not recognized by the speech recognizer, its topic is irrelevant to the current domain, or its interpretation is inconsistent with the dialogue context.

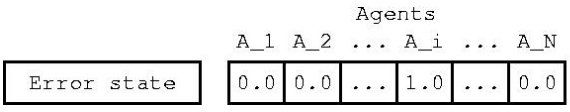

Figure 5: An error evaluator.

Because all possible system actions are reusable agents, we can easily implement a different dialogue management strategy by adding evaluators, or replacing the current QEstimate evaluator. We are developing another strategy based on recurrent self-organizing maps, that learns to map dialogue states to correct actions by fuzzy clustering, minimizing the amount of human labor in designing the dialogue strategy.

6 Demo System and Future Work

The project deals with both speech and text communication, interfacing with the user by telephone and web-applications. An initial integrated system with basic functionality and interaction capabilities concerning bus timetable inquiries has been built. Sample interactions are given in Figures 6 and 7. Various other application types such as intelligent question-answer systems, automatic call-centers, and other services that would benefit from flexible natural interaction can also be envisaged.

The current system is the first prototype and intended to provide useful basis for further experimentation and system development. The planned activities include further investigations of the limitations and advantages of various approaches for language technology, as well as usability studies concerning real users and users with special needs.

(1) UT: *mmilläs bussilla mä pääsen malamille*
'with which bus do I get to Malami'
UR: millä bussilla pääsen malmille
'with which bus do I get to Malmi'

SU: *mistä olittekaan lähdössä malmille*
'where do you want to leave for Malmi'

UT: *no hakaniemestä*
'well from Hakaniemi'
UR: hakaniemestä
'from Hakaniemi'

SU: *bussi 70 lähtee sieltä malmille*
'bus 70 leaves from there to Malmi'

UT: *selvä kiitos hei*
'ok thank you bye'
UR: bussi kiitos hei
'bus thank you bye'

SU: *kiitos hei*
'thank you bye'

Figure 6: Sample dialogue with low recognition score for the destination *Malmille* 'to Malmi' (UT = Utterance as actually made by the user, UR = Utterance as recognized by the system, SU = System utterance).

(2) UT: *koskas mennee seuraava pussi arapiasta malmille*
'when will the next bus go from Arapia to Malmi'
UR: koska menee seuraava bussi arabiasta malmille
'when will the next bus go from Arabia to Malmi'

SU: *kello 14 30*
'at 14:30'

UT: *minkä numeroinen se on*
'which number is it'
UR: minä nelonen selvä
'I four okay'

SU: *olen pahoillani osaan vastata vain bussiaikatauluja koskeviin kysymyksiin*
'sorry I can reply only to questions concerning bus timetables'

UT: *aha no joo jaa kiitos hei*
'I see ok thank you bye'
UR: onko aijaa kiitos hei
'is I see thank you bye'

SU: *kiitos hei*
'thank you bye'

Figure 7: Sample dialogue with misrecognition of user utterance *minkä numeroinen se on* (UT = Utterance as actually made by the user, UR = Utterance as recognized by the system, SU = System utterance).

7 Acknowledgements

The research is carried out in the context of the USIX Interact project, a collaboration project between four Finnish universities, and funded by the National Technology Agency, the leading IT companies ICL Invia oyj, Sonera oyj, Lingsoft oy, and Gurusoft oy, as well as the Finnish Association for the Deaf and the Arla Institute.

References

J. Allwood, D. Traum, and K. Jokinen. 2000. Cooperation, dialogue and ethics. *International Journal of Human-Computer Studies*, 53:871–914.

L. Carlson. 2001. CPARSE manual. http://www .ling.helsinki.fi/ lcarlson/cparse09en.html.

T. Jauhiainen. 2001. Using existing written language analyzers in understanding natural spoken Finnish. In *Proceedings of Nodalida '01*, Uppsala.

K. Jokinen and G. Wilcock. 2001. Confidence-based adaptivity in response generation for a spoken dialogue system. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, pages 80–89, Aarhus.

K. Jokinen, T. Hurtig, K. Hynnä, K. Kanto, M. Kaipainen, and A. Kerminen. 2001. Self-organizing dialogue management. In *Proceedings of the 2nd Workshop on Natural Language Processing and Neural Networks*, pages 77–84, Tokyo.

K. Jokinen. 1996. Goal formulation based on communicative principles. In *Proceedings of the 16th COLING*, pages 598–603.

F. Karlsson. 1983. *Suomen kielen äänne- ja muotorakenne*. WSOY, Juva.

T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, V. Paatero, and A. Saarela. 2000. Organization of a massive document collection. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks for Data Mining and Knowledge Discovery*, 11(3):574–585, May.

T. Kohonen. 1995. *Self-Organizing Maps*. Springer, Berlin.

K. Koskenniemi. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. University of Helsinki, Helsinki.

K. Lagus and J. Kuusisto. 2002. Topic identification in natural language dialogues using neural networks. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, Philadelphia.

S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. 1998. Galaxy-II: A reference architecture for conversational system development. In *Proceedings of ICSLP-98*, Sydney.

R. Sutton and A. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts.

M. Turunen and J. Hakulinen. 2000. Jaspis - a framework for multilingual adaptive speech applications. In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing.

M. Turunen and J. Hakulinen. 2001a. Agent-based adaptive interaction and dialogue management architecture for speech applications. In *Text, Speech and Dialogue. Proceedings of the Fourth International Conference (TSD-2001)*, pages 357–364.

M. Turunen and J. Hakulinen. 2001b. Agent-based error handling in spoken dialogue systems. In *Proceedings of Eurospeech 2001*, pages 2189–2192.

M. Vainio. 2001. *Artificial Neural Network Based Prosody Models for Finnish Text-to-Speech Synthesis*. Ph.D. thesis, University of Helsinki.

C. Watkins and P. Dayan. 1992. Technical note: Q-learning. *Machine Learning*, 8:279–292.

G. Wilcock. 2001. Pipelines, templates and transformations: XML for natural language generation. In *Proceedings of the 1st NLP and XML Workshop*, pages 1–8, Tokyo.