AthosMail:
a Multilingual Adaptive Spoken Dialogue System for the E-mail Domain

Markku Turunen, Esa-Pekka Salonen, Mikko Hartikainen, Jaakko Hakulinen
Department of Computer Sciences
University of Tampere
FIN-33014 University of Tampere, Finland
dumas@cs.uta.fi

William Black, Allan Ramsay, Adam Funk, Andrew Conroy, Paul Thompson, Mark Stairmand
Computation Department
UMIST
PO Box 88, GB-Manchester M60 1QD, England
dumas@co.umist.ac.uk

Kristiina Jokinen, Jyrki Rissanen, Kari Kanto, Antti Kerminen
University of Art and Design Helsinki
Hämeentie 135 C, FIN-00560 Helsinki, Finland
dumas@uiah.fi

Björn Gambäck, Magnus Sahlgren, Fredrik Olsson, Maria Cheadle, Preben Hansen, Stina Nylander
SICS – Swedish Institute of Computer Science AB
Box 1263, SE-164 29 Kista, Sweden
dumas@sics.se

Abstract

AthosMail is a multilingual spoken dialogue system for e-mail message reading on a mobile phone. The key features of the application are adaptivity and the integration of different approaches to spoken interaction. The application has a flexible system structure supporting multiple components for both different and the same purposes. The AthosMail system includes components for input interpretation, dialogue management, output generation, user modelling, and text processing. Suitable components are selected dynamically to make the interaction adaptive, and various adaptive techniques have been experimented with. For example, the system's responses are tailored according to the user's actions so as to adapt to the observed skill levels of the user. The paper presents the overall AthosMail functionality and the main components of the system.

1 Introduction

AthosMail is a multilingual (Finnish, English, Swedish) speech-based e-mail application. The main purpose of the application is to allow the user to access his/her mailbox using a standard mobile or desktop phone.

AthosMail is based on the existing Mailman application (Turunen and Hakulinen, 2000b). In the EU-funded DUMAS project (IST-2000-29452) new components are produced for text-processing, dialogue management, semantic template construction, user modelling, integrated tutoring, and random indexing. The overall goal and participants of the project were presented by Jokinen and Gambäck (2004). The system has been evaluated with a subjective evaluation paradigm, and preliminary results from a user study are promising (Hartikainen et al., 2004). In this paper the system functionality and its main components are presented.

2 AthosMail Functionality

AthosMail offers functionality for most e-mail reading tasks. The system provides both speech input (speech recognition) and DTMF (touchstone) interfaces which may be used in a multimodal fashion. The functionality of the system is presented in more detail in the sections following; a complete dialogue example is included in Appendix A.

2.1 Overview of e-mail messages

After a successful login the system informs how many messages there are in the user's mailbox, how many of them are new, and how many are already seen ones. After that, the system groups messages into semantically meaningful groups and lists them automatically. After the presentation of groups the system is user-initiative. The user is able to navigate between message groups and the messages that they contain, and read, mark, and delete messages.

2.2 Navigation in message groups

The user selects groups using grouping indexes (i.e., "Go to the second group") or grouping criteria (e.g., "What messages do I have from Mary?"). User may also request all sender names (e.g., "Please list the senders"). When necessary, the system dynamically creates new groups based on the user queries.

The user can ask the system to read the overall information (sender, subject) of all messages of the selected group, e.g., to list the messages of a group.

Messages are also listed automatically upon a selection of a message group.

2.3 Selection of messages

Messages can be selected for reading and further examination within the selected group. User can select a message using message indexes (e.g. “Select the second message) or using references such as “Next” and “Previous”. When the user selects a message its basic information, such as date and time, is read. More information about the message can be requested. This includes a summary of each message. Users can mark messages as deleted or important. These help the system to learn the characteristics of each individual user.

2.4 Message reading

Messages are spoken in chunks corresponding to the identified paragraphs of the original message. The language of paragraph is identified and the system uses appropriate synthesizers for each paragraph. Many special elements (e.g., citations, e-mail addresses, etc.) are also identified and spoken in a suitable manner for each language.

The user can use touchtone keys to navigate between chapters in a message. User can skip a chapter, repeat a chapter, move to the previous chapter and cancel the reading.

2.5 Help and guidance

The AthosMail system uses both implicit and explicit guidance. The user’s actions and the system’s performance and error rates are monitored and used to estimate the user’s expertise levels, according to which system responses are tailored. This provides implicit guidance to users.

For explicit guidance the system contains a set of tutoring agents, which introduce the system functionality and monitor how well the interaction proceeds.

The standard help message is a static description of the most important system functionality. Several universal commands are available for all system functionality. For example, context-specific help messages are presented for each individual situation when requested. Similarly, a detailed description about the last feature used is presented when requested. In addition, system can repeat the previous output in a way that only meaningful information is repeated.

3 AthosMail System Overview

The AthosMail system consists of two parts, as depicted in Figure 1. The AthosMail application is the actual dialogue system that the user has interaction with. The functionality of the AthosMail system was presented in the previous section.

The offline system receives requests from the online system and performs operations on the mailbox. For this purpose it utilizes a collection of components for the handling of e-mail messages. These components offer services for communicating with the IMAP and POP3 servers, transforming text messages into XML documents, filtering out unwanted messages and modifying the contents of messages. Messages are prioritised and categorized into meaningful groups. The MailServer also contains components for user modelling. The online AthosMail application and the offline MailServer communicate using the XML-RPC communication interface. The offline system is necessary because some of the techniques used for message processing are resource intensive, and it is not possible to perform them in real-time.

The main modules of the AthosMail application and the offline MailServer are depicted in Figure 2. Next, the system architecture and the main system modules are presented.

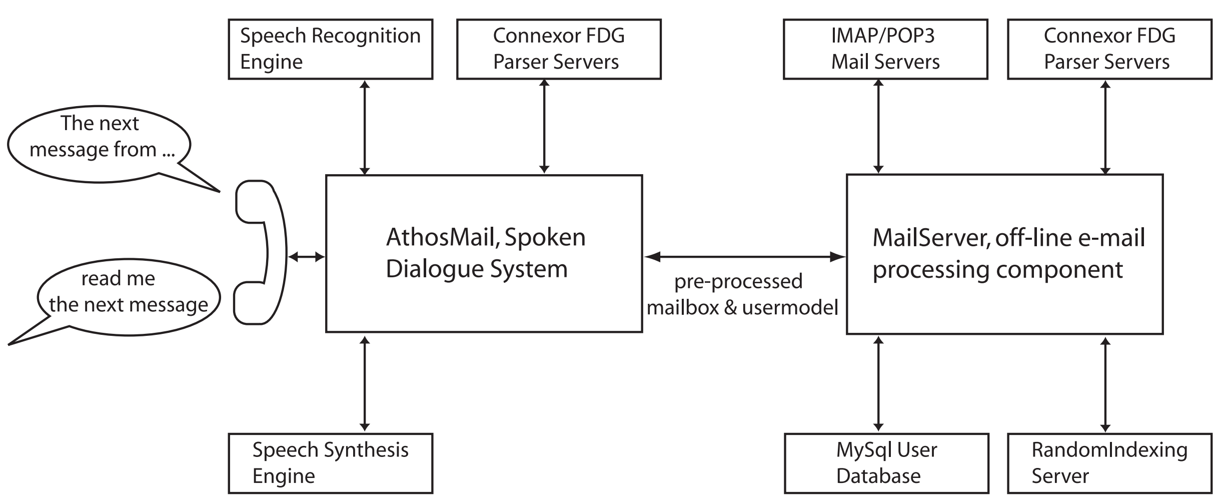

Figure 1: Overall AthosMail structure.

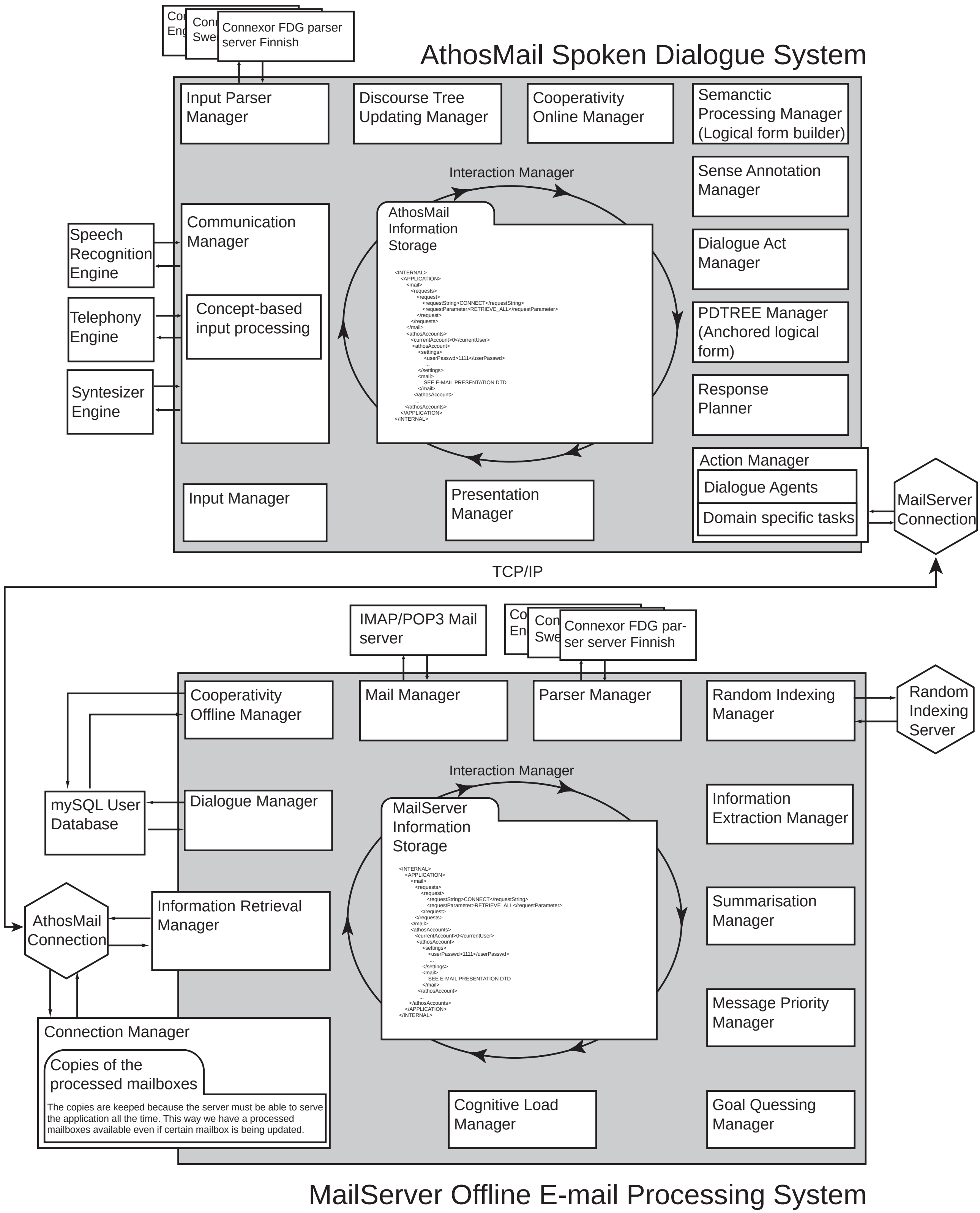

Figure 2: AthosMail and MailServer modules and managers.

4 Adaptive System Architecture

The AthosMail system is constructed on top of the Jaspis architecture (Turunen and Hakulinen, 2000a; Turunen and Hakulinen, 2003). The Jaspis architecture supports highly distributed but coordinated components, shared system knowledge and system-level adaptation. This means in practise that the system consists of several managers that are under central coordination. Each of the managers includes various amounts of agents to handle different tasks, and there are also different agents that handle the same task in different ways. Evaluators are used to choose between different agents. This feature plays a major role in the system adaptation. The system architecture is distributed so that different managers and even agents can run on different computers and platforms.

The Jaspis architecture uses shared information management. AthosMail components store all their information in the Information Storage. It contains the discourse tree, which holds all dialogue related information, including user inputs, dialogue data and system outputs. In this way, all information is shared between various system agents and evaluators.

The general system-level adaptation mechanism is applied to various tasks. An example from the output generation follows. The presentation agents are evaluated with five different evaluators. Each evaluator monitors a specific feature that should be taken into account and gives a score for each agent. The evaluators look for the following attributes: (i) dialogue data (e.g. event, dialogue act or action), (ii) language of the output (e.g. Finnish, Swedish or English), (iii) the explicity value that the user modelling suggests for the current output, (iv) organization of messages in the mailbox, and (v) self-evaluation results of agents. Evaluation scores are combined and the agent with the highest overall score is selected to generate the output.

Similar adaptation mechanisms are applied to other system modules. In total, the AthosMail online system contains more than 10 managers and over 100 agents and evaluators. Further details can be found in Turunen et al. (2004). The MailServer has its own set of agents, evaluators and managers.

5 Input Processing

Three approaches to input processing are used. They are used together to provide a robust processing of spoken inputs for different languages and situations.

5.1 Concept-based input processing

In the concept-based input processing approach a keyword and key phrase spotting method is used. Predefined and dynamically built concept definitions are matched to recognized inputs. The predefined parts match the system functionality and the dynamic parts are related to the domain information (e.g., sender names). The concepts are kept in the Information Storage, and they can be updated dynamically. Input processing is performed by a set of agents and evaluators, which implement the concept model on the base of the concept definitions. There can be a separate set of agents for each used input language.

The concepts are represented with two information fields, the name of the concept and a normalised parameter. Two synonymous phrases have exactly the same information field content. The original input is stored to identify actually used words (for example, to adapt the system outputs to the language of the user), and to distinguish what part of the input has been processed already by other concept agents.

Multiple concepts are used to allow flexible parsing, for example to filter out unnecessary or unwanted parts of the input. For example, if the user says “Mark maybe uh let’s see seventh message” and recognition result is “Mark do I have quit seventh message”, the concept agents produce three concepts, first one about marking, second about quitting and third one about selecting seventh message. The dialogue agents distinguish the number of input concepts and choose the most likely concept. In the example above, an agent that handles the marking of messages would be chosen.

5.2 Semantic processing

The second input parsing approach is based on Connexor’s robust parser, which produces a functional dependency analysis with morphosyntactic tags for each token (Tapanainen and Järvinen, 1997). This analysis is handled by the logical-form-building agent, which carries out a semantic analysis based on the Parasite system (Ramsay, 1990). Because the Connexor parser marks each token in the user utterance with morphosyntactic information and specifies the dependencies between the tokens, the semantic agent is largely language-independent, although it does contain a small lexicon for each project language with determiners, pronouns, auxiliary verbs and numbers.

The logical-form-building agent produces a sign (similar to an HPSG sign; Pollard and Sag, 1994) for each token, merges them together into one sign that spans the whole utterance, and uses compositional semantics to produce a logical form for the utterance. Finally, it returns the resulting logical form to the Information Storage.

It is important to note that the logical form is unanchored, i.e., its referring expressions have not been dereferenced, and that the logical-form builder is stateless and treats each utterance independently.

5.3 Sense annotation

AthosMail has two types of semantic analysis tasks to address: the interpretation of the (spoken) user commands to the system, and the interpretation of the (written) documents, the e-mails. The language of e-mails in many ways resembles that of speech, with shorter and grammatically incomplete utterances. To handle this type of input we need a very robust interpretation strategy; thus we aim to complement the deep-level, logical-form-based processing of the previous section with a more template-based, domain-dependent semantic interpretation (Cheadle and Gambäck, 2003), in order, for example, to retrieve some e-mails relevant to a user query, to extract a piece of knowledge from a set of e-mails, or for summarising some e-mails selected by the user.

6 Dialogue Management

The interaction model of the AthosMail application is based on the user initiative dialogue strategy with some mixed-initiative features. Depending on the user input, the dialogue control is handled by a suitable set of agents using a flexible dialogue management approach (Salonen et al., 2004). In the first approach the PDTREE agent and the Response planner agent are used. In the second approach a set of highly specialized dialogue agents are used.

6.1 Anchoring and discourse modelling

The PDTREE agent carries out much of the pragmatic analysis: anchoring, discourse modelling and interpretation of the user's goal. This agent is based on the Parasite system (Ramsay and Seville, 2000). PDTREE collects the unanchored logical form (see Section 5.2) from the Information Store and processes it to maintain a dialogue model and to evaluate the user's goal as expressed in the user's utterance. In particular, it creates a sequentially numbered "discourse state" for each utterance in the dialogue.

The PDTREE agent processes a query containing the unanchored logical form and a representation of the propositional content and entities mentioned in the last utterance from the system to the user. The agent first creates a discourse state corresponding to the system's previous utterance, then begins processing the logical form by "anchoring" the entities by comparing their semantic representations with existing information; each entity is tied to a Skolem constant and is classified in one of the following categories: *mentioned*, *infelicitous*, *accommodated*, *ambiguous*, and *new*. Anchoring consists principally of looking for the most recently mentioned (and therefore most salient) entity that matches the characteristics specified in the reference (e.g., gender, number, entity type).

Each discourse state contains the following information: a sequential number, the theme of the utterance, a list of the entities to which the utterance refers with their respective classifications, and a list of Skolemized facts representing the propositional content of the utterance.

The PDTREE agent examines the current discourse state and the inferences to produce a goal which represents how the system can fulfil the user's intention. It produces a goal which consists of a goal type, a query summary and a solution list in one of the following four ways: *wh-question* (e.g., "What messages do I have from Bob?"), *polar question* (e.g., "Do I have any messages from Bob?"), *command* (e.g., "Delete all messages from Bob.", and *statement* (e.g., "All messages about Viagra are spam.").

Confirmations and cancellations are treated as commands and produce the propositions yes and no, respectively, in the solution list. If the user asks for something that does not exist in PDTREE's model (e.g., "Messages from Fred" when there are none) or uses an ambigous expression (e.g., "Read I"' after the system says "There are three messages from Bob"), the anchoring will mark these reference failures as accomodated or ambiguous and this information will be included in an extra section of the goal.

6.2 The Response Planner

The Response Planner produces an abstract plan for an utterance that can be realised in different ways by the output generation components. The propositions output are analysed to populate a dialogue model of dialogue events, objects and relationships between them; this effectively records what happens, to which objects and when in the dialogue. Application-specific knowledge is required to know what objects, events and relationships are likely to occur. The system may look specifically for "a mail from person x", and if found record a new mail object, a new person object and a new relationship between them all occurring at this point in the dialogue.

Sets of pre-specified declarative rules are then unified with the dialogue model. These rules may be nested, and may have associated conditions to test the number of matching results. Rules in a set are evaluated in sequence, respecting their hierarchical organisation. If unification is successful, then an "action" node can be written referencing unified values in the matching clauses.

The rules can apply to the whole dialogue model recorded at that time, so the logic need not be restricted to the current dialogue "turn". In some cases it is appropriate to reference previous turns, for example to help recover from dialogue failures.

6.3 Specialized dialogue agents

AthosMail contains more than 50 dialogue agents which represent generic dialogue tasks and application functionality. For example, the system contains an agent for listing of messages, a set of agent giving context-sensitive help, and agents for various confirmations. The dialogue agents use inheritance to separate generic tasks from domain specific tasks.

Dialogue agents operate on the base of language independent conceptual inputs. Dialogue agents produce conceptual events, which will be further processed by the domain specific action agents. For example, if the recognized input concept is "READ" with a parameter "7", event READ_MESSAGE_EVENT will be produced if there are at least seven messages. If there are fewer messages, a different event, such as READ_MESSAGE_EVENT_ERROR, will be produced. This information, along with other parameters, is stored to the active node in the discourse tree.

In addition to the basic functionality, AthosMail contains dialogue agents for special situations, such as universal commands and interactive tutoring (see Section 9). User modelling agents (see Section 8) support dialogue agents.

6.4 Action Manager

The Action Manager is responsible for carrying out the tasks that the dialogue management suggests (e.g. activating a certain message for reading, fetching a specified mailbox from MailServer etc.). The Action Manager coordinates several agents specialised for different tasks. As mentioned previously, inheritance is used to encapsulate domain specific tasks. Domain specific action agents carry out the requests of the Response Planner and the specialized dialogue agents. In addition, some of the action agents operate directly on the base of the dialogue state (the discourse tree). For example, there is an agent that handles the updating of the current chapter when a message is being read to the user.

The output of the Action Manager is a request to produce a surface form out of the *action* produced by the Response Planner or the *event* produced by the dialogue agents.

7 Output Generation

AthosMail contains two approaches for output generation. In the first approach the Generator Agent is used to produce the system responses, and in the second approach a set of highly specialized presentation agents is used for response generation. In both approaches, the explicitness of the system response reflects the user's assumed competence level as determined by the User Model (Section 8).

7.1 The Generator Agent

The Generator Agent takes as its input an abstract representation of a system utterance that has previously been produced by the dialogue components. The surface strings are generated by concatenating together a sequence of pre-defined utterance segments. These segments may be fixed strings (e.g., help messages), or they may contain variables that are instantiated according to the contents of the abstract utterance representation (e.g., the number of messages in the set retrieved in response to a user query).

A configuration file specifies which utterance segments should be concatenated together, in which order, for each type of abstract utterance representation that the response planner can produce. There are three separate surface utterance specifications for each type of abstract representation. This is because the actual realization of each type of utterance varies according to the value of the "explicity" user modelling parameter. This parameter, whose value varies from 1 to 3, defines how verbose system utterances should be.

As an example, consider the user query "Do I have any messages from Bob". The abstract representation would lead to the generator producing one of the following responses, according to the value of the explicity parameter: (i) "Two", (ii) "Two messages from Bob, or (iii) "There are two messages from Bob. What would you like to do?".

7.2 Specialized presentation agents

In the second generation approach, numerous presentation agents are used to handle different generation tasks. Each agent is capable of producing output for a specific situation, dialogue act or event. There can be a set of presentation agents for each output language. Agent selection is based on the adaptive mechanism presented in Section 4. This makes the system easy to extend, since new components can be added without modifications to the existing ones.

There are many types of agents. For example, there is a set of agents based on a configurable agent, which concatenates pre-defined and dynamic information together, such as login phase greetings, help messages, and message listings. Another type of agents read e-mail messages. They can be configured to generate suitable surface forms out of specific elements from e-mail contents, such as emoticons and URLs.

There can be multiple output requests to handle each turn, and for each request an agent is selected. The results are concatenated together, when appropriate, to form a single system response. Alternatively, multiple responses can be produced, for example when multiple synthesizers are used.

8 User Modelling

The purpose of the AthosMail User Model component is to provide flexibility and variation in the system utterances, so as to allow the users to interact with the system in a more natural way. We also aim at implementing and testing machine learning techniques on the UM data, especially concerning message prioritization and message categorization to produce suitable views of the incoming mails to be presented to the user (see Section 10). In our earlier work (Jokinen et al., 2002), we have described our experiments on user modelling and learning systems.

The UM has the following sub-components: Cooperativity component, Message Prioritization, and Message Catogorization. The components produce recommendations for the dialogue management and output generation components so that they can use these recommendations when producing system responses (see Section 4). The system also records a set of user preferences that are fixed properties and preferences of the user dealing with aspects like the preferred speaking style, speed, and voice, or the preferred message senders and topics. In the beginning of the interaction the default user preferences are loaded into the system from the UM.

The Cooperativity Model records the user's actions, and estimates the user's competence levels that will be further used to give recommendations on the appropriate way of responding. For instance, when the user is a novice, the system provides longer and more explicit utterances than when the user is familiar with the system and its functionality. The user's expertise levels are tied to the individual commands, and thus the length and the amount of information in different system responses can vary depending on how frequently the user has used the particular commands.

The model comprises an online component and an offline component. The former is responsible for observing runtime events and calculating expertise recommendations on the fly, whereas the latter makes long-time observations and calculates default expertise values to be used at the beginning of the next session. The User Model adaptivity is thus exemplified dynamically in each actual interaction, but it is also recorded in the system's memory of the long-term interaction, where the overall learning curve behind the local variations is tried to be caught. The model is described in more detail in (Jokinen and Kanto, 2004).

9 Context-sensitive Help and Tutoring

AthosMail supports a set of universal commands which adapt to the interaction. These commands are available in other applications and other domains as well. With the "Tell me more!" command the user can get a detailed presentation of the current dialogue situation. For example, when a message is selected, only the main information is spoken. With this command the user can access additional information, such as length of the message, date and time.

Another universal command is the "What now?" command that triggers a context sensitive help, in which only relevant guidance is given to the user, taking into account the current dialogue situation and history.

One of the experimented additions to the system is tutoring agents (Hakulinen et al., 2003), which give guidance to the user by introducing the system and monitoring how well the user interacts with the system. Tutoring components take the initiative in certain situations. They utilize the dialogue history, the user model and their own tutoring plan in this process. No modifications are needed for other components, and the tutoring feature can be turned dynamically on and off.

10 Message Processing

The message processing in AthosMail is carried out in the separate MailServer application, as discussed in Section 3. Each user has an account, where the e-mail specific information (e.g., name of the actual mailbox) is defined. MailServer processes accounts continuously. MailServer also maintains the long time user model and exchanges information with the dialogue application. Next, these tasks are presented.

First the messages are retrieved from the user's mailbox and converted to XML format. Several agents are employed to identify and mark e-mail specific elements, such as emoticons, URLs, digit patterns, e-mail addresses, and citations. Regarding to the parsing results, keywords are extracted and summaries are generated. The messages are then categorised to meaningful groups. Various techniques are employed.

Message Prioritization allows the system to sort out incoming messages so that messages that the user most likely finds interesting and important are in the beginning of the list. The Message Priority component analyses how important a message is for a user and it suggests possible message orderings for folders based on the importance of the messages in the folders. The importance of a message is a function of user actions: what the user has done earlier with the same kind of messages. The message priority component analyses message features such as the sender, received group, subject, keywords and topics, and gives a score from -1 to +1 to each of these pieces of information separately. The importance of the whole message is a weighted sum of the scores of message features.

The message categorization component suggests possible classifications for incoming email. The idea is to categorize new email based on the content of the email rather than basing the categorization on surface cues (such as keywords or phrases). The content-based categorization is based on the Random Indexing vector space methodology (Karlgren and Sahlgren, 2001), which accumulates semantic representations of words based on co-occurrence statistics.

Finally, the messages are grouped using an algorithm which tries to make both meaningful and easily manageable groups. Information about the messages, such as header information, is used for this. The number of messages within each group is kept in the limits of human capabilities to handle spoken data. This helps the user in getting an overview of the mailbox even with large number of messages. By dynamically adapting the contents of the mailbox for each situation it is possible to use alternative input methods and strategies, such as DTMF keys.

In addition, the project has investigated methods for the three main information refinement tasks: Information Retrieval, Information Extraction, and Text Summarisation. In AthosMail, the e-mails are the documents, and the e-mails that best match the query are retrieved, ranked and presented to the user, usually as a list of possibly relevant documents. The IR engine is here then mainly used as a supporting agent in the IE and summarisation tasks. In addition to this, the project has investigated the use of cooccurrence-based methods for automatic query expansion and lexicon extraction (Sahlgren et al., 2002), and for content-based Text Categorisation (Sahlgren and Cöster, 2004).

11 Conclusion

In this paper we have presented the functionality and main components of a multilingual e-mail application. In particular, we have demonstrated how the AthosMail application uses various approaches to make the interaction adaptive.

References

Cheadle, M. and Gambäck, B. (2003). Robust Semantic Analysis for Adaptive Speech Interfaces. Proc. of Universal Access in HCI: Inclusive Design in the Information Society (4): 685-689.

Hakulinen, J., Turunen, M. and Salonen, E.-P. (2003). Agents for Integrated Tutoring in Spoken Dialogue Systems. Proc. of Eurospeech 2003: 757-760.

Hartikainen, M., Salonen, E-P. and Turunen, M. (2004). Subjective Evaluation of Spoken Dialogue Systems Using SERVQUAL Method. Proc. of ICSLP 2004 (to appear).

Jokinen, K. and Gambäck, B. (2004). DUMAS - Adaptation and Robust Information Processing for Mobile Speech Interfaces. Proc. of Baltic Conference “Human Language Technologies – The Baltic Perspective”: 115-120.

Jokinen, K. and Kanto, K. (2004). User Expertise Modelling and Adaptivity in a Speech-based E-mail System. Proc. of ACL 2004.

Jokinen, K., Rissanen, J. , Keränen, H, and Kanto, K. (2002). Learning Interaction Patterns for Adaptive User Interfaces. Proc. of the ERCIM Workshop on User Interfaces for All.

Karlgren, J. and Sahlgren, M. (2001). From Words to Understanding. In Foundations of Real-World Intelligence: 294-308, Stanford.

Pollard, C. and Sag, I. (1994). Head-Driven Phrase Structure Grammar. The University of Chicago Press.

Ramsay, A. (1990). The Logical Structure of English: Computing Semantic Content. Pitman, London.

Ramsay, A. and Seville, H. (2000). Models and Discourse Models. Journal of Language and Computation, 1(2): 159-174.

Sahlgren, M. and Cöster. R. (2004). Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization (Submitted).

Sahlgren, M., Karlgren, J., Cöster, R. and Järvinen. T. (2002). SICS at CLEF 2002: Automatic Query Expansion Using Random Indexing. In Working Notes for the 3rd Workshop of the Cross Language Evaluation Forum.

Salonen, E-P., Hartikainen, M.,Turunen, M., Hakulinen J. and Funk, J.A. (2004). Flexible Dialogue Management Using Distributed and Dynamic Dialogue Control. Proc. of ICSLP 2004 (to appear).

Tapanainen, P. and Järvinen, T. (1997). A Non-projective Dependency Parser. Proc. of the 5th Conference on Applied Natural Language Processing. Association for Computational Linguistics: 64-71.

Turunen, M. and Hakulinen, J. (2000a). Jaspis - A Framework for Multilingual Adaptive Speech Applications. Proc. of ICSLP 2000.

Turunen, M. and Hakulinen, J. (2000b). Mailman - a Multilingual Speech-only E-mail Client based on an Adaptive Speech Application Framework. Proc. of Workshop on Multi-Lingual Speech Communication (MSC 2000), 2000: 7-12.

Turunen, M. and Hakulinen, J. 2003. Jaspis^2 - An Architecture For Supporting Distributed Spoken Dialogues. Proc. of the Eurospeech 2003: 1913-1916.

Turunen, M., Salonen, E-P., Hartikainen, M. and Hakulinen, J. (2004). Robust and Adaptive Architecture for Multilingual Spoken Dialogue Systems. Proc. of ICSLP 2004 (to appear).

APPENDIX A: EXAMPLE DIALOGUE

	Dialogue transcription:	English translation (where needed):
S:	*Tervetuloa käyttämään Athosmailia. Ole hyvä ja anna käyttäjätunnuksesi.*	Welcome to the AthosMail system. Please enter your four digit user code.
U:	`XXXX [DTMF]`	XXXX [DTMF]
S:	*Anna salainen tunnuksesi.*	Please give your secret passcode.
U:	`XXXX [DTMF]`	XXXX [DTMF]
S:	*Hetki, otan yhteyden postipalvelimeesi.*	Please wait, connecting to your mail server.
S:	*Tervehdys, Testikäyttäjä. Ei uusia viestejä. 17 vanhaa viestiä. Listaan ryhmät. Sinulla on 5 ryhmää. Ensimmäisessä ryhmässä, viestit otsikolla "Reply: about deliverables".* *Toisessa ryhmässä viestit lähettäjältä Adam Funk.* *Kolmannessa lähettäjältä Kristiina Jokinen.* *Neljännessä lähettäjältä Kari Kanto.* *Viides, loput viestit.*	Hi, Test user. No new messages. 17 old messages. Listing groups. You have 5 groups. Group one, messages with subject "Reply: about deliverables". Group two, messages from Adam Funk. Three, messages from Kristiina Jokinen. Four, messages from Kari Kanto. Fifth contains miscellaneous messages.
T:	*Hei. Olen opastaja ja kerron Sinulle miten järjestelmää käytetään. Valitse seuraavaksi jokin ryhmistä. Tämä tapahtuu sanomalla esimerkiksi "kolmas ryhmä". Käytä siis haluamasi ryhmän numeroa.*[1]	Hi, I'm your tutor. I'll teach you how to use the system. Next choose one of the available groups. You can do this by saying, for example, 'third group'. So, use the group number you wish.[1]
U:	`Valitse kolmas ryhmä`	Select the third group
T:	*Hyvä. Nyt olet siis valitsemassasi ryhmässä.*[1]	Good. Now you are in the group of your choice.[1]
S:	*Kolmas ryhmä, viestit lähettäjältä Kristiina Jokinen.* *Ensimmäisen viestin otsikko on "D6.1".* *Toisen otsikko "tele-meeting on Tuesday 21/10".* *Kolmannen "Deliverables due!!"* *Voit valita viestin sanomalla esimerkiksi ensimmäinen viesti tai toinen viesti.*	Group three contains messages from Kristiina Jokinen. Subject of first message is: D6.1. Subject of second, tele-meeting on Tuesday 21/10. Third, Deliverables due!! You can choose a message saying for example first message or second message.
U:	`Listaa lähettäjät`	List senders
S:	*Viisi lähettäjää. Listaan lähettäjät. Adam Funk, Markku Turunen, Kristiina Jokinen, Kari Kanto, Maria Cheadle ja Veikko Surakka.*	Five senders. Listing senders. Adam Funk, Markku Turunen, Kristiina Jokinen, Kari Kanto, Maria Cheadle and Veikko Surakka.
U:	`Mitä viestejä minulle on Adam Funkilta?`	What messages do I have from Adam Funk?

[1] These are examples of the tutoring outputs (see Section 9). They would appear like presented in the dialogue, but for the demonstration the tutoring agents will be turned off.

S:	*Toisessa ryhmässä on viisi viestiä lähettäjältä Adam Funk.* *Ensimmäisen otsikko on ”AthosMail review, urgent”.* *Toisen otsikko”Reply: Running AthosMail”.* *Kolmannen ”Reply: SICStus Bugreport (SPRM 5425)”.* *Neljännen ”AthosMail: Integrating the generator”.* *Viidennen ”A few questions about the speech recognition components”.*	Second group contains three messages from Adam Funk. Subject of first message is: ”AthosMail review, urgent”. Subject of second, ”Reply: Running AthosMail”. Third ”Reply: SICStus Bugreport (SPRM 5425)”. Fourth ”AthosMail: Integrating the generator”. Fifth ”A few questions about the speech recognition components”.
U:	`Haluan lisätietoja ensimmäisestä viestistä`	I'd like to hear more information about the first message
S:	*Message number one from Adam Funk, subject is ”AthosMail review, urgent”. Message was sent on Wednesday at 14.24. It has 6 lines. Summary of message: “AthosMail review, urgent. Hi, I got this message from you.”*	
U:	`Lue viesti`	Read the message
S:	*Message with subject ”AthosMail review, urgent” is sent by Adam Funk.*	
S:	*Lainaus edellisestä viestistä: “Onko kaikki siellä päässä jo valmiina?”. Lainaus päättyy.*	Quote from previous message: “Do you have everything ready? Quote ends.
S:	*”Hi, I got this message from you. Could you please translate what it means?”*	
S	*”Our components are finished. I made a draft of changes. You can find it at http://www.umist.edu.uk/AthosMail”* *Cheers, Adam.*	
S:	*The message has been read.*	
S:	`Poista se`	Delete it
U:	*Poistetaanko viesti otsikolla ’AthosMail review, urgent’, jonka on lähettänyt Adam Funk. Sano kyllä tai ei.*	Please confirm that message ‘AthosMail review, urgent’, send by Adam Funk, is to be deleted. Answer yes or no.
S:	`Kyllä`	Yes
S:	*Viesti on merkitty poistettavaksi.*	The message has been marked for deletion.
S:	*Ei uusia viestejä. 17 vanhaa viestiä.*	No new messages. 17 old messages.
U:	`Kiitti hei!`	Thank you and bye!
S:	*Haluatko nyt lopettaa? Sano kyllä tai ei.*	Do you want to quit now? Please say yes or no.
U:	`Kyllä`	Yes
S:	*Heippa, soitellaan.*	Bye, please call again.