USING DIA-MOLE FOR UNSUPERVISED LEARNING OF DOMAIN-SPECIFIC DIALOGUE ACTS FROM SPONTANEOUS LANGUAGE

Jens-Uwe Möller

Natural Language Systems Division,
Dept. of Computer Science, Univ. of Hamburg,
✉ Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany,
☎ ++49 40 5494 - 2516 / Fax: - 2515,
💻 jum@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de/NATS/staff/moeller

Abstract. This report introduces DIA-MOLE, a tool that supports an engineering-oriented approach towards dialogue modelling for a spoken-language interface. Our approach is applied to the domain of appointment scheduling. A major step towards dialogue models is to know about the basic units that are used to construct a dialogue model. DIA-MOLE does not employ theory-based dialogue units because they are subject to human interpretation and often cannot be recognized from data available in a spoken-language system. We pursue a data-driven approach and apply unsupervised learning to a sample set of spontaneous dialogues using multiple knowledge sources, i.e. domain and task knowledge, word recognition and prosodic information. Using these data, DIA-MOLE supports segmentation of turns and interpretation of their illocutionary force based on a model of the task. For this purpose we had to develop a model of interactive problem solving in the domain of appointment scheduling. As a result of learning we obtain domain- and task-specific dialogue acts (DDA).

A first validation of the set of learned DDAs shows that they are prominent for this domain and task. Some DDAs show significant correspondence to specific nodes in an RST-structure. Automatic DDA labeling was compared with human dialogue act labeling according to a predefined labeling scheme. Dialogue act prediction was also employed to evaluate our approach.

Kurzfassung. Das Werkzeug DiaMoLE unterstützt einen ingenieursmäßigen Ansatz zur Dialogmodellierung für eine gesprochen-sprachliche Mensch-Maschine-Schnittstelle. Der vorgestellte Ansatz findet Anwendung in der Domäne Terminvereinbarung. Ein bedeutender Schritt auf dem Weg zu einem Dialogmodell ist die Kenntnis der zugrundeliegenden Einheiten, aus denen es sich zusammensetzt. In Dia-MoLE finden keine theoriebasierten Dialogeinheiten Anwendung, die menschliche Interpretation voraussetzen und die auf der Basis vorhandener Daten in einem gesprochen-sprachlichen System nur eingeschränkt erkannt werden können. Unter Verwendung verschiedener Wissensquellen, wie Domänen- und Aufgabenwissen, Worterkennung und Prosodie wird anstattdessen ein datengetriebener Ansatz verfolgt. Ausgehend von diesen Daten wird in Dia-MoLE die Segmentierung von Äußerungen und die Interpretation der Illokution basierend auf einem Modell der Aufgabe vorgenommen. Hierzu mußte für die Domäne Terminvereinbarung ein Modell interaktiver Problemlösung entwickelt werden. Anschließend wird ein unüberwachtes Lernverfahren auf eine Reihe derartig vorverarbeiteter spontansprachlicher Dialoge angewendet. Als Ergebnis des Lernverfahrens werden domänen- und aufgabenspezifische Dialogakte (DDA) gebildet. Weitere Dialogbeiträge können aufgrund der DDA klassifiziert werden.

Erste Analysen bestätigen, daß die Menge der gelernten DDAs charakteristisch für die Domäne und Aufgabe sind. So zeigen einige DDAs signifikante Übereinstimmung mit bestimmten Knoten in einer RST-Struktur. Automatisches Labeling mit DDA wird verglichen mit human-gelabelten Dialogakten. Darüberhinaus wurde eine Dialogaktvorhersage implementiert, um die Qualität der Ergebnisse dieses Ansatzes auch in dieser Hinsicht zu evaluieren.

CONTENT

FBI-HH-B-191/96

Bericht des Fachbereichs Informatik,
Universität Hamburg

Report of the Dept. of Computer Science,
University of Hamburg

Vogt-Kölln-Str. 30
D-22527 Hamburg
Germany

1. INTRODUCTION

Dialogue modelling for natural language interfaces has become a topic of substantial interest during the last decade. It relates work on discourse analysis and speech acts with tasks in an application domain and human computer interaction. Most approaches assumes the existence of a set of units, often called dialogue act which are considered building blocks for a dialogue model.

We should first have a look at different views at and definitions of *dialogue acts*. In some approaches, a set of dialogue acts is defined within a theory of action that is based on models of mutual knowledge. Dialogue act perform some change of knowledge [Coh87, Coh90, Faw89]. Bunt [Bun94] describes dialogue acts as contributing to different dimensions of context, like linguistic, semantic, physical, social and cognitive context. All of these theories ascribe some meaning to a single unit, while structure-oriented approaches mainly ascribe meaning to relations between units. They build up exchange structures [Wac86, Jön91] or are made for text structuring, as for instance Rhetorical Structure Theory (RST) [Man88] that does not cover interaction [Moo92]. Several researchers argued that the domain and task structure has an important impact on discourse structure and thus integrated them into their theories and systems [e.g. Woo84, Gro86, Lam91]. There is also some work on integrating different structures into a unique framework [Faw92, Dar94].

The *recognition* of any of these discourse structures from linguistic structures is by itself a proper subject of research [Hir93, Lit94, Kno91, Mas95].

Basic units like dialogue acts are used to build up some *dialogue model* . Two approaches for representing a model or structure of discourse are competitive: plan-based models [All82, Lit85, Car91, Lam91, Ram91a, You94, Chu94] and discourse grammars [Bil91, Faw89, Bun89, Ste93]. Some approaches combine both representations, e.g. [Ale94].

Basic units and plans or grammars are means of representing models. We will now concentrate on the question of how to obtain the content of a model for the natural language interface in an application, i.e. how to *engineer dialogue models*. Engineering has at least two facets:

- Which information or knowledge is relevant for dialogue models?
- How can we acquire this knowledge?

Our own prior work has contributed to the design of dialogue structures starting out from a knowledge-based system [Möl90, Möl92a, Möl94]. Other engineering-oriented work contributes to dialogue-acquisition techniques [Jön93].

1.1. Some problems of dialogue engineering

When modelling a dialogue for the natural language interface of an application, we have experienced that all the theories mentioned above provide some means for structuring a dialogue, but none of them well fits to what is needed in a specific application environment

[c.f. also Jön93]. All theories fail to explain some phenomena, and we have shown in prior work that there are even domain-specific structures constituting certain kinds of domain-specific dialogues [Möl92b, Ram91b]. Most existing dialogue system engineers thus develop their own set of dialogue acts [c.f. Mai94, Jek95].

Additionally, labeling dialogues is a very time-consuming work, it requires substantial expertise in linguistics and domain knowledge, and it usually does not provide reproduceable results because the classification of a dialogue into units as specified by those theories is sometimes ambiguous [c.f. Hir92] and depends upon the labeling person's understanding and interpretation. To overcome these deficiencies statistical methods have been developed to evaluate the consistency of labels set by a person and the inter-personal quality and integrity of a labeling system [Cun95].

A further throw back for the integration of natural language front-ends into application systems is that the acquisition of appropriate dialogues is already a complex and long-lasting task. Either naturally occuring dialogues have to be recorded, transcribed (at great expense) and afterwards to be restricted to a set of possible interactions as supported by the application system. Alternatively, one may adopt some Wizard-of-Oz experiments to acquire dialogues [Jön93, Bad94], but for practical reasons in a dialogue engineering task Wizard-of-Oz, is only useful for written language: Transcriptions of phonetic and prosodic data in spoken language can be afforded for research prototypes, but they are far beyond the possible effort in an industrial dialogue engineering task, thus there is a loss of certain information of spoken language when analysing transcribed dialogues.

As a consequence, an approach to engineering of dialogue models for spoken dialogue front-ends should only use a small number of transcripts, and transcripts should be easy to make, like simple speech-to-text transcriptions. Instead, the dialogue engineering approach should rely more on resources which are available in the specific speech-processing system, e.g.domain knowledge of the underlying knowledge based system, prosody recognizer and word recognizer.

1.2. Steps towards a solution for dialogue engineering

1.2.1. RST and the idea of a new theory of dialogue structures

In a first experiment, we used data from the VERBMOBIL (VM) project[1] and ascribed rhetorical relations to textual transcriptions. It turned out that they are not always applicable, notably to interaction phenomena, e.g. when motivating turn shift etc., and we had to add two relations COMMITMENT and WIDEN to enable a RST-structure for whole dialogues.[2] We compared applicable relations with prosodic transcriptions and discovered some correlations between RST relations and prosodic events as well as topic shifts, thus supporting the results given in [Hir92].

[1] VERBMOBIL is a german joint project working on a speech translation system.

[2] COMMITMENT became necessary for some structures in a dialogue, where the satellite confirms the nucleus and states some commitments. WIDEN is the counterpart of ELABORATION.

This raises the question, whether it makes sense to develop a new theory that integrates different structures, e.g. to incorporate interaction phenomena into RST. Fischer et al. chose this option in the CORINNA system [Fis94]: Based on a speech act-oriented RTN-like dialogue model that represents illocutionary aspects[3] they built up RST-structures for information seeking dialogues. While some of their exchange structures map to existing RST-relations, they had to add some task-specific relations, like SUPERTERM, which they call tactics, to be able to get a RST-structure for whole dialogues.

Resulting RST-structures from our data appeared not to be sufficient as base for a dialogue model, and we did not find any indications to reliably recognize relations. Our previous experiences with domain-specific structures were one objection to the idea of developing a new theory.

Another challange was our goal of reducing the cost of dialogue modeling. A new theory does not help solving the engineering problem, it only contributes to the problem of providing appropriate representations. One task of dialogue model engineering is to recognize structures. Litman has successfully shown that machine learning is an easy way of inducing classification rules for cue words. This allows for processing a larger amount of data which raises accuracy compared to manually abstraction of rules [Lit94, c.f. also Sie94]. Similar results have been reported on discourse-level information extraction [Leh94].

1.2.2. Learning dialogue structures

One possible approach for building a dialogue model under the objective of efficiency is to use similar supervised machine learning techniques to classify data from dialogues into classes each of which corresponds to some previously known unit from one of the dialogue or discourse structuring theories.

Even if supervised learning would lead to a good recognition of units from existing discourse or dialogue theories, we would be confronted with the problem of combining different theories in such a way that we are able to build up a dialogue model which considers interaction aspects as well as discourse structure. As we are interested in classes of these combinations, we pursued a data-driven approach. We followed the suggestion of Litman: "Another advantage of the machine learning approach is that the ease of inducing ruleset from many different sets of features supports an *exploration of comparative utility of different knowledge sources*"

We use a learning algorithm for a) inventing new classes of dialogue structuring units and b) describing how these classes could be characterised in terms of data that are available in a spoken dialogue system. A very natural unit is the dialogue turn, but some turns are lengthy and include multiple dialogue acts. Therefore, turns have to be segmented into meaningful units. The problem of segmentation in spoken language arises. Our solution to segmentation is pointed out later in this paper.

[3] Fischer, Maier and Stein call their dialogue model COR (COnversational Roles).

The advantage of an unsupervised learning approach to dialogue structuring lies within the reduced effort to obtain a dialogue model. In manually built dialogue models as well as with supervised learning, it is necessary to transcribe, analyze and label dialogues according to some theory. By using unsupervised learning, one may process hundreds or thousands of dialogues.

Our data-driven approach to dialogue engineering discussed so far relies on meaningful segments in a dialogue from which a model is abstracted. To support this approach, we need a tool that enables us to form these segments and their meaning from data in a spoken language system environment and that enables us to learn some type of models. This tool is introduced below.

2. DIA-MOLE: A TOOL THAT SUPPORTS LEARNING OF MODELS OF SPONTANEOUSLY SPOKEN DIALOGUE

It is the goal of DIA-MOLE (DIAlogue MOdel Learning Environment) to bring to light domain-specific dialogue structures. Available data in natural language systems are used to form segments and to ascribe some meaning to them. Domain and task knowledge is used to interpret the meaning of segments. All these information sources are put together, and an unsupervised learning algorithm supports deriving classes of segments.

These classes are characterized by different aspects each contributing to the model of co-operative and interactive problem-solving, i.e. the domain task. We call our classes domain-specific dialogue acts (DDA), because the interpretation of the illocutionary force of dialogue contributions is grounded in the model of the domain task. The learning of dialogue acts is a central issue in our approach for domain-oriented engineering of spoken dialogue systems.

In section 4 we will show that some of the DDAs in our domain correspond to acts in the speech act tradition, while the illocutionary force of other DDAs is directly related to the underlying domain and task model.

2.1. General design guidelines for DIA-MOLE

Our engineering-oriented approach to dialogue modeling is based on some design guidelines, which are characteristic features of the overall DIA-MOLE tool and its subcomponents:

- *Task-Orientation:* Domain-independent approaches to dialogue modelling tend to be less specific, and therefore their recognition guarantees less accuracy. Domain-independent dialogue act recognizers can be exclusively based on linguistic phenomena to interpret the illocutionary force and do not consider the specificity of a domain-specific sublanguage [Gri86, Leh86]. Our approach does not pursue general dialogue capabilities, instead it should be restricted to goal-oriented dialogues and to the sublanguage of a specific domain and task. We argue that this is the most reliable way of recognizing dialogue acts in a spoken dialogue system.

- *Robustness:* In a spoken dialogue system robustness means that there must be a useful reaction even if only parts of speech have been recognized. Complete parsing of speech will fail in most cases. For this reason we applied partial parsing that picks up relevant parts while irrelevant or unrecognisable parts remain unconsidered. Similar decisions have been made in most application-oriented spoken dialogue systems [e. g. Nov96, Lup96, Mus93, War96], and recently Eckert [Eck96] pointed out, why partial parsers must be used when analyzing spontaneous speech.
- *Use data from the system:* The upper-level interpretation of data into some meaningful units strongly depends on lower-level data (e.g. probability scores from word recognition). Consequently, we only use data that are already present in an existing spoken natural language system environment, or data which are not subject to system change.
- *Minimise the engineering effort:* Data analysis is very expensive, leads to many errors and often includes human interpretations which cannot be reconsidered by a computer. For this reason we should used learning methods.
- *Use multiple knowledge sources:* Spoken dialogue contains many channels of information, and it is the combination of all of those channels including domain and task knowledge as well as situational knowledge that enables understanding.

The following section gives an overview on different knowledge sources within a dialogue system that may constitute these channels.

2.2. Evaluating multiple knowledge sources that could be used to engineer dialogues in a spoken language dialogue system

This section deals with the availability and structure of knowledge sources, their use for dialogue modelling and their integration into DIA-MOLE.

2.2.1. Domain-specific knowledge

The structure of the task and of the domain knowledge are major factor for creating coherence in goal-oriented dialogues [Möl90, Ram91b, Möl94]. Knowing these structures is useful for the recognition of a dialogue act as it strictly reduces ambiguity.

A domain and task model is an integral part of most knowledge-based systems. Although knowledge about the domain and the task is already available this kind of system environment, some additional modelling has to be done in order to adapt it to the dialogue module. Especially the task model for interactive problem solving which is part of a dialogue is different from the task model for automated problem solving. An example for different task models is given later on in the context of building up models for the domain of application scheduling.

Task knowledge controls inference steps and is typically represented using goals and plans. In natural language systems as well as in some other knowledge based systems, domain facts describing the structure of and objects in a domain are often represented by

terminological logics. A system that clearly distinguishes different knowledge levels and allows different well investigated representations is KARL [Ang91, see also Fen94]. We adopt their notions denoting different knowledge levels, which stem from the KADS methodology [Bre86].

2.2.2. Data from the word recognizer

A first step in processing *written* language is to extract words from character strings and to access a lexicon. When processing *speech*, a word recognizer analyses a signal using pattern recognition techniques and provides word hypotheses or graphs of word hypotheses. Word hypotheses are probabilistic and may be ambiguous, but they are certainly one of the basic source of information even for recognizing dialogue acts. For our first experiments we used a word recognizer system with a logarithmic acoustic score and time intervals that are retrained by forced alignment. Later experiments were conducted with real data from the word recognition including errors like wrong assertion or deletion of words and word changes. The word recognizer is based on the system described by Hübener et al. [Hüb96]. Figure 1 shows the 'best path'-output of the word recognizer we used. The first two columns provide absolute begin and end times of a word, the last column consists of the acoustic score.

```
310     460     ja          -76.611984
460     800     prima       -82.753677
800     1010    dann        -87.347595
1010    1230    lassen      -78.228905
1230    1380    Sie         -74.399940
1380    1470    uns         -83.025497
1470    1590    doch        -80.284233
1590    1650    noch        -87.236458
1650    1780    einen       -75.157867
1780    2060    Termin      -83.896736
2060    2580    ausmachen   -80.108749
2580    2760    wann        -81.232422
2760    2860    w"are       -87.095200
2860    2970    es          -78.521584
2970    3210    Ihnen       -74.976280
3210    3330    denn        -84.806023
3330    3700    recht       -70.472916
```

Figure 1: Output of a word recognizer

2.2.3. Prosodic data

Besides word recognition, prosodic features like accent or pause can be deduced from the speech signal, too [Nöt91]. Prosody also yields a sentence melody, but this is of less importance in our context. By intonation, accents are set and thus, important information is stressed and less important things are put into the background [Bat93]. Semantic units are often marked by pauses at the beginning and at the end (cf. section 3.3.1 and figure 12). These properties are supposed to be clues to the understanding of spoken language, whereas in written language properties like syntax, interpunctation and more elaborate formulation have to compensate the function of prosody.

```
 590  --    690 ms accent, length  100 ms, confidence 0.471084
1010  --   1130 ms accent, length  120 ms, confidence 0.650081
1330  --   1430 ms accent, length  100 ms, confidence 0.561492
1690  --   1780 ms accent, length   90 ms, confidence 0.330556
1840  --   2070 ms accent, length  230 ms, confidence 0.477964
2300  --   2470 ms accent, length  170 ms, confidence 0.400369
2650  --   2750 ms accent, length  100 ms, confidence 0.442033
2780  --   2890 ms accent, length  110 ms, confidence 0.551336
2960  --   3490 ms accent, length  530 ms, confidence 0.410003
```

Figure 2: Output of an accent recognizer.

Kompe et al. [Kom94] have shown that it is even possible to derive some functional role of utterances from prosody. We used data from a prosodic labeling component that yields accent and senctence modality (question, assertion, continuation) [Str95]. Figure 2 shows some of these data.

2.2.4. Syntax and Semantics

Spoken language does usually not consist of well-formed sentences [Web95, Eck96], thus a dialogue system cannot rely on syntactically sound sentences. Nevertheless, smaller units like nominal phrases may contribute a little to contribute understanding of spontaneous speech. Compared to dialogue systems, processing written natural language, syntax is less important for spoken dialogue systems [Sch92, Sch93]. For this reason, syntactic parsing in DIA-MOLE is restricted to partial parsing. Semantic processing has not been integrated into this prototype yet.

2.3. Architecture of the DIA-MOLE dialogue-act learner

The DIAlogue MOdel Learning Environment (DIA-MOLE, figure 3) addresses the problem of learning dialogue models within a spoken language system environment. The main module learns domain-specific dialogue acts (DDA).

Word recognizer [Hüb96] and prosodic tagger [Str95] are modules that were developed independent of this work. Both processes (round boxes) provide input data (rectangular boxes) for our system. Domain knowledge and task knowledge are either already existing in the background knowledge-based system, or they have to be engineered. Domain specialists carry out syntactic and meaning analysis according to the domain structure and the underlying task model. As a result they provide domain-relevant events. Segmentation breaks turns into segments which are labeled with features from prosody and domain events. These labeled segments are presented to a learning algorithm resulting in a classification hierarchy for domain-specific dialogue acts (DDA).

The resulting DDA hierarchy can be used to classify and automatically label segments. From automatically labeled dialogues, a prediction module learns about predicting a subsequent act in a dialogue. Predictions may be used by again other modules of the spoken dialogue system to adapt their environment. E.g. the word recognizer may use dialogue act predictions to choose a specific language model trained on these DDA classes to improve

word recognition. Furthermore, a dialogue-planner module can use predictions to display natural behaviour. The architecture of DIA-MOLE also allows self-adapting dialogue models. If DIA-MOLE is integrated in a spoken dialogue system which is in practical use, all occuring dialogue turns, or more precisely, segments may be presented to both, the DDA classifier for further processing of that turn and to the DDA learner to improve the model in its application situation. For this reason we adopted an incremental learning algorithm within the DDA learning module. In this report, we focus on DDA learning within DIA-MOLE.

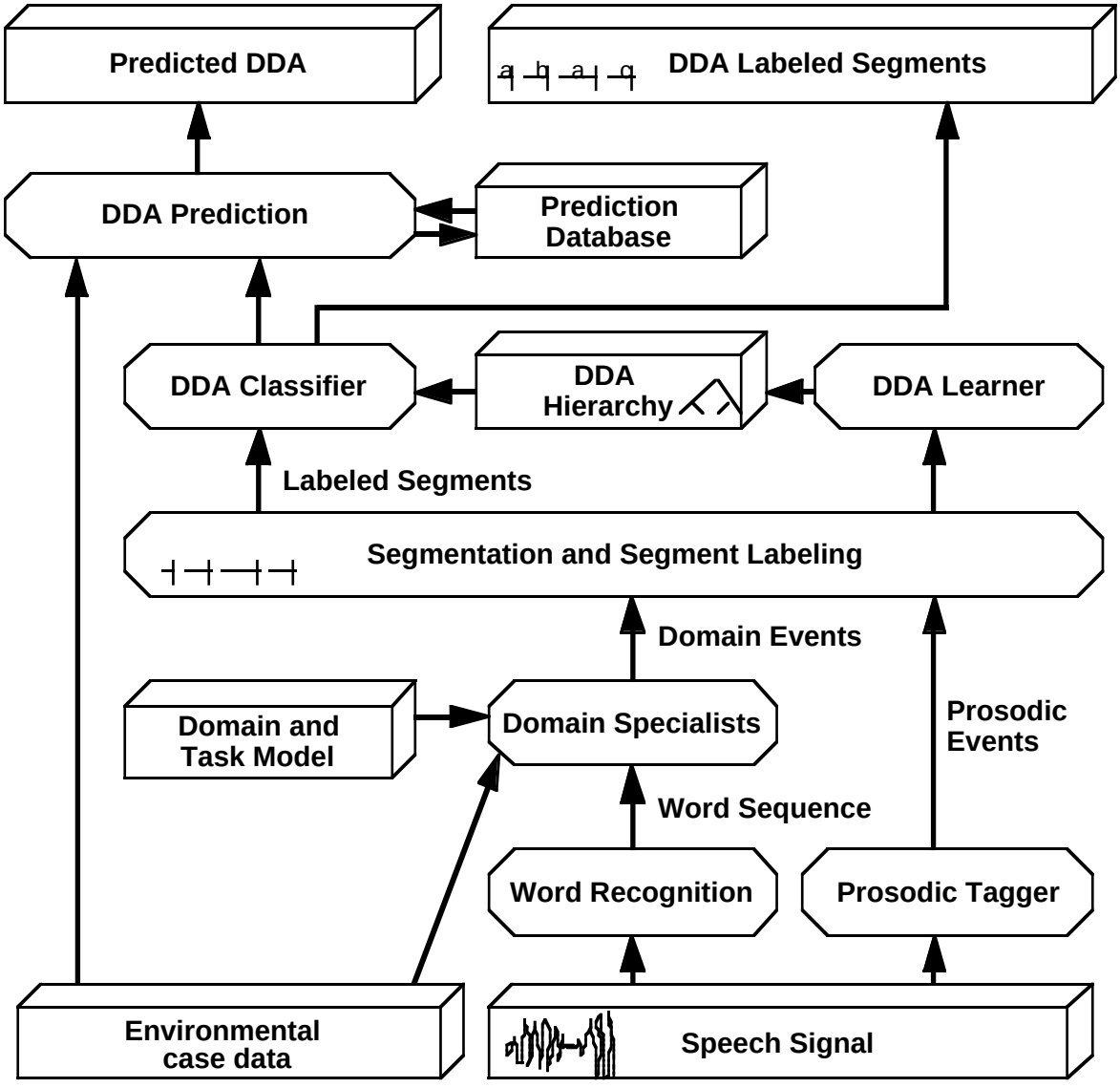

Figure 3: Architecture of DIA-MOLE

2.4. Learning algorithm

2.4.1. Requirements and choice

Some criteria for distinguishing learning algorithms has been provided by Fisher [Fis85]. Supervised learning of dialogue acts – as it is done by most learning algorithms – would mean that the recognition of existing dialogue acts from data has to be learned. Although this an interesting research topic, supervised learning of dialogue acts is out of the question for our research because we will deliberately not rely on an existing theory of dialogue structure. Our learning problem is that of aggregating segments or, with other words, we do concept formation for new classes of dialogue structuring units. Consequently, learning should be purely data-driven, i.e. unsupervised.

As we know from many parts of natural language processing, different information channels, e.g. topicalisation (syntax) and accent (prosody), are often combined to express things. This is an argument against using monothetic learning algorithms because they would classify along a single attribute. Instead, we have to use one of the rare polythetic learning algorithms that classifies along several attributes, simultaneously.

Another distinguishing feature of learning algorithms is the kind of the result that we achieve. Does learning result in a set of mutually exclusive clusters, a classification tree or a set of clusters which might overlap? We would prefer a learning algorithm that produces a hierarchy because we are interested in the combination of different attributes characterizing a dialogue act. If we set up different clusters, we would more or less regain the attributes that we provided to the learning algorithm.

A last distinguishing feature of learning algorithms is the way in which data have to be presented to the algorithm. Most algorithm need to know all data to compute a result, while some learning algorithms perform incremental learning and produce a result after every piece of data, they have processed. An incremental learning algorithm is usually less optimal, but it allows to adopt a system in practice with little computational effort.

There are not many unsupervised learning algorithms working polythetic. The demand for a hierarchy and directly inspectable classifying attributes rules out some neural network algorithms that carry out unsupervised learning like Kohonen networks [Rit89, Koh84], competitive learning [Rum80, Gro87] or ART [Car88, Car90]. One of the best known symbolic machine-learning algorithm that fulfills our criteria is COBWEB [Fis87] or its decendent CLASSIT [Gen89], respectively. Thus a COBWEB-like algorithm is the basis for our work. There are some extensions concerning overlapping clusters and multiple hierarchies [Mar94], which we have not considered for the reasons given above. For the experiments presented in this report, we have developed our own algorithm that comprises ideas from COBWEB and CLASSIT and added features for dealing with uncertain and incomplete knowledge.

2.4.2. Basic definitions from COBWEB and CLASSIT

For abstracting classes, COBWEB uses a *measure of clustering quality*. In the following formula, A are attributes, V are values, and C are classes. $P(A_i=V_{ij}|C_k)$ is the probability of a value in a class or its predictability. $P(C_k|A_i=V_{ij})$ is the probability of a value being part of a class, or its predictiveness.

$$\Sigma k \Sigma i \Sigma j \; P(A_i=V_{ij}) \, P(C_k|A_i=V_{ij}) \, P(A_i=V_{ij}|C_k)$$

Function 1: Clustering quality

From this quality measure, a value called *category utility* is derived by transformation and normalization that allows to determine where to integrate new cases. K is the number of cases. The algorithm manipulating the hierarchy is described in Fisher [Fis87].

$$(\Sigma k \; P(C_k) \; \Sigma i \Sigma j \; P(A_i=V_{ij}|C_k)^2 \; - \; \Sigma i \Sigma j \, P(A_i=V_{ij})^2) \; / \; K$$

Function 2: Category utility in COBWEB

CLASSIT replaces discrete values from COBWEB with numeric values using the same algorithm to manipulate the hierarchy. σ_{ik} is the standard deviation of an attribute in a given class and σ_{ip} is the standard deviation for a attribute in its parent node.

$$(\Sigma k \; P(C_k) \; \Sigma i \; 1/\sigma_{ik} \; - \; \Sigma i \; 1/\sigma_{ip}) \; / \; K$$

Function 3: Category utility in CLASSIT

Computing the category utility according to function 3, raises two problems:

- If a class comprises only one value, standard deviation is zero and the value of the devisions is infinite. To catch this error, Gennari et. al introduce a minimal σ called *acuity*.
- Numeric values with very small difference will create different classes and they are never classified as being identical, although the difference is just some noise in numeric data. To enable the algorithm to classify unsharp values into a single class, they needed to introduce a *cutoff*.

2.4.3. Extensions and improvements for learning in DIA-MOLE

In our implementation, we integrated numeric and symbolic values. For this purpose, we had to normalize numeric values. The category value for numeric values in our implementation is then modified to

$$(\Sigma k \; P(C_k) \; \Sigma i \; 1/\sigma_{ik} 2\sqrt{\Pi} \; - \; \Sigma i \; 1/\sigma_{ip} 2\sqrt{\Pi}) \; / \; K$$

Function 4: Normalized category utility for numeric values in DIA-MOLE

In addition to integrating symbolic and numeric values, we extended the expressiveness towards the needs of dialogue modelling and spoken language. Data that are presented to the learning algorithm may now contain

a) multiple same attribute-value pairs,

b) probability values for attribute values and

c) probability values for whole cases.

The extension (a) allows us to handle double existence of the same attribute-value pairs being different from the single existence of such a pair, although their place in the classification hierarchy will be very close. Input of probability values for attribute-value pairs (b) enables us to use probability values stemming from underlying modules in a spoken language system directly for processing within the learning algorithm. They may express the quality of data. Moreover, it is possible to exert influence on the importance of specific attributes and values for the resulting classification hierarchy. The input of probability values for whole cases (c) enables us to consider a measure of quality for cases.

Computation of the functions 2 and 4 had to be changed to incorporate these extensions. Instead of computing the attribute-value probability for symbolic values according to function 5 as the number of an attribute having a specific value divided through the overall number of that attribute, we added extensions b) and c) resulting in function 6. AQ is the quality of an attribute having one specific value and CQ the quality of a whole case, both given as probability values, c stands for cases.

$$P(A_i=V_{ij}) <=> P(V_{ij}|A_i) <=> \Sigma\, c\, A_{ci}=V_{ij} / \Sigma\, c\, A_{ci} <=> \#A_i=V_{ij} / \# A_i$$

Function 5: Attribute-value probability in COBWEB

$$P(A_i=V_{ij}) <=> \Sigma\, c\, (A_{ci}=V_{ij})\, Q_{ij} / \Sigma\, c\, A_{ci} Q_{ix} \text{ , where } Q_{ab}=AQ_{ab}\, CQ_{ab}$$

Function 6: Attribute-value probability in DIA-MOLE

Similar changes had to be made for numeric values. Instead of computing mean and standard deviation like in CLASSIT (functions 7) both are weighted with the quality Q_i in DIA-MOLE resulting in functions 8, where $Q_i=AQ_i\, CQ_i$.

$$\mu = 1/n\ \Sigma\, i\, V_i \quad \text{and} \quad \sigma = \sqrt{(1/n\ \Sigma\, i\, (\mu - V_i)^2)}$$

Functions 7: Mean and standard deviation for numeric attributes in CLASSIT

$$\mu\omega = \Sigma\, i\, V_i\, Q_i / \Sigma\, i\, Q_i \quad \text{and} \quad \sigma\omega = \sqrt{(\Sigma\, i\, (\mu\omega - V_i\, Q_i)^2 / \Sigma\, i\, Q_i)}$$

Functions 8: Weighted mean and weighted standard deviation for numeric attributes in DIA-MOLE

While the goal of learning is to find some abstractions from a learning set of data that also hold for similar but different data (test set), over-learning is the effect of loosing the abstraction by adapting too much to the learning set. To prevent over-learning, we

implemented the cutoff as described by Gennari [Gen89], but it did not provide any useful results. For pruning the classification hierarchy during learning we developed a tree-pruning method that uses a

- maximal equality boundary.

This pruning methods works for numeric and for symbolic values. Normally, the learning algorithm classifies a case down and sometimes changes the hierarchy as long as the category utility for the integration of one case into one class is less than the category utility for the creation of a new node and and thus an enlargement of the classification tree. This is always the case, if the attribute set of a case is not identical to the attribute set of a class and leads to single classes for every different attribute combination at the leaves of the classification tree. In our implementation further activities of the learning algorithm are stopped, if the category utility for the integration of one case into one class exceeds the value of the maximal equality boundary. For the readers comprehension we would mention at this point that this is a value that holds for all cases that are classified into or under this class.

2.4.3. Extensions and improvements for classification in DIA-MOLE

While the authors of most learning algorithm concentrate on how to learn classification hierarchies, they do not particularly address how to use their classification hierarchy. This hold for Fisher and Gennari, too. Our implementation of a classification algorithm in DIA-MOLE was realized similar to learning: It computes the category utility for all subclasses of root, chooses the best, computes again the category utility for all subclassses of that class and so on. It does of course not compute the category values for different hierarchy modifications or change the classification hierarchy. We suppose, that this is the way, Fisher and Gennari applied their classification hierarchy, too.

Application of this step-wise classification reveals a trace or path through the classification hierarchy, which in its reversed list form (from the most specific class to the root) we will call *classification path*. Additionally, when classifying cases, we apply two further pruning methods to the resulting classification path:

- as an absolute boundary the minimal number of cases that were classified into that class, and
- as a relative boundary the maximal case prediction.

The first pruning method *minimal case number* cuts off all classes that are too specific and were based on a too small number of cases during learning. The second and relative boundary corresponds to the maximal equality boundary during learning and should be smaller than it. Case prediction is the category utility for a given class if a new case would be integrated, and *maximal case prediction* is a boundary, where classification is stopped, if it is reached. Both pruning methods reduce the number of classes into which cases are typically classified. Although this mostly provides good results, we added a further criterion for classification path pruning that works together with the maximal case prediction:

- a good prediction gain value.

Sometimes when classifying a case, the case prediction starting at root stays for several levels down the classification hierarchy near 0, then it jumps to a high value, e.g. 0.7, from one level to the next and again stays for several levels under some finishing boundary. Other cases show a linear prediction gain for every level that they are classified down the hierarchy. When classifying cases, e.g. segments into DDAs, we are interested in getting a small number of classes with high predictions, but realizing the structure of the hierarchy, it is obvious that the higher the number of classes, the better prediction rates are. So there is a trade off between the number of classes and the prediction that could be achieved. To maximize informational content in our DDA classes, we stop classification, if there is a significant gain of case prediction, e.g. 0.5, from one level to the next.

Our learning algorithm provides a classification path for each segment in a dialogue. We use the most specific class in the classification path as a label for the dialogue act. This label is ascribed to all segments that are classified into this class. The label itself is meaningless, a number, but the class itself should be a meaningful cluster of DDAs. In the next section we introduce one module within DIA-MOLE that allows to evaluate a learned set of DDAs.

2.5. Dialogue Act Prediction

One measure for evaluating the quality for a given number of types of dialogue acts is their predictability. DIA-MOLE offers a module for building up n-gram models, i.e. the prediction of a case n when knowing all n-1 predecessors and the probability of this prediction, or speaking in terms of dialogue modelling, this module allows to predict the most probable subsequent dialogue act. For evaluating predictability, we only consider the two most probable branches.

The way segments in DIA-MOLE are labeled with a dialogue act class allows some additional treatment of relaxation from the most probable prediction. Instead of comparing the prediction with the class into which the succeeding segment will be classified, we compare the path of a predicted class and the path of succeeding segment. If the most specific classes in both paths are not the same, we could compare more general classes from the path. With this technique, we have the opportunity to recombine those classes of dialogue acts which can be clearly distinguished by data, but which are too specific to serve as a base for prediction.

3. AN IMPLEMENTATION FOR THE DOMAIN OF APPOINTMENT SCHEDULING

As an example domain we chose appointment scheduling. This choice enabled us to use data on spoken dialogue that are available in our department within the VERBMOBIL project (VM). We could also use preprocessed data from word recognition and prosody without having to build these components on our own within DIA-MOLE. Our scenario differs from the VM scenario in that it does not describe a multilingual translation situation, but a monolingual situation with a human speaking to an appointment scheduling system. This system could be e.g. a dentist's telephone servant. Consequently, our scenario of a spoken

dialogue system for appointment scheduling lies in the tradition of natural language frontends.

3.1. Analysis and Modelling of the Domain Structure

3.1.1. A First Model of the Functional Structure for Appointment Scheduling

An important precondition for our approach is the existence of a domain and task model. They will allow us to interpret the intention of the communcative agent's utterances. As argued above, domain and task model are usually part of the background system and are thus already provided. Nevertheless, it is not always possible to simply use these models, especially not the task model, because the structure of interactive problem solving may be different from automatic problem solving.

When we started working on the VM domain, a compact domain model was not available, so we had to do this work on our own. Using the modelling method KADS [Bre86] we first did an analysis of the problem and a typical problem solving method for that domain. In a first step we neglected the fact that appointment scheduling is an interactive task.

The main *problem* of appointment scheduling in the VM scenario is to find a time slot for an appointment. A simple solution structure would include *duration, start time* and *end time, location, participants* and the *goal of an appointment*. *Constraints* concerning an appointment, like its goal, usually are fixed in advance, but if they become the topic of negotiation, we deal with intertwinned problems. Intertwinned problems are very typical for real-world applications. [Bre94] writes on the coupling of different problems: "In real-life applications, a problem type hardly ever comes alone, but rather in chains that reflect the fact that the conclusion of one problem is a required input to solve a next problem." The analysis of the VM corpus brought to light that constraints are not necessarily fixed as a first step of the dialogue, if they become a topic at all. Instead, a typical situation for making explicit constraints is if some constraints are violated by the interaction partner. This means that constraints are fixed before the dialogue was recorded, or the dialogue partners have intuitive ideas of their constraints. Additionally, in some cases previously fixed constraints are changed during the dialogue, if otherwise no satifying solution could not be found.

If we compare the problem in appointment scheduling dialogues with the definition of problem types given by Breuker [Bre94], the problem in these dialogues is an assignment problem, at least when constraints are fixed: An appointment is matched to calendars. The typical problem solving method for assignment problems is *propose-and-revise* [Sun94]. Sundin's methodological steps to come to a functional structure for problem solving were applied to our problem. We will not introduce intermediate steps, but just present the resulting structure. In figure 4 round boxes are processes, and data are in rectangle boxes. *Components* are appointments that have to be scheduled. In our domain mostly one appointment has to be scheduled giving one *unit*. *Resources* are calendars and their entries. In automatic problem solving theses resources are aggregated to a kind of common calendar. *Partial assignments* could be parts of appointment sequences. We do not consider the left-most partial assignment box. From *resources* and *unit* a *proposal* is computed, which is then

evaluated according to constraints resulting in conflicts. The *proposal* is *modified* to solve *conflicts*. *Assignment* contains the final solution.

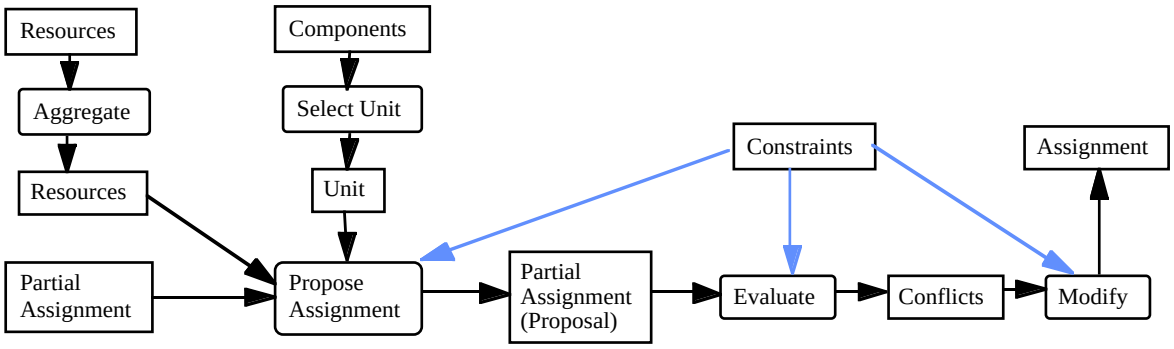

Figure 4: Functional structure of appointment scheduling

3.1.2. Differences between Automated and Interactive Appointment Scheduling

The functional structure of appointment scheduling reflects a solution for *automated* problem solving, notably constraint satisfaction. It was the starting point for developing a functional model of *interactive* appointment scheduling. There are several differences between automated and interactive appointment scheduling that forced us to develop a new task model:

- While automated problem solving would consider all dimensions of the solution, i. e. time and location simultaneously, in interaction a solution is refined step by step, which means that all dimensions are negotiated successively.
- Automated problem solving considers all solutions at once (there is no cycle in Figure 4), while in interaction all solutions are tested subsequently, and conflicts are build up incrementally.
- There is no aggregation of resources in interaction as every partner keeps his own calendar. Instead of aggregation there is *propose-and-revise.*
- There are interaction control acts like the typical demand for evaluation after a partial assignment.
- Besides cyclic refinement during *propose assignment*, we found explicit stating of local constraints or focus, like "let us try the 3rd week, what about the seventeenth...". The structure of human appointment scheduling integrates the method *hypothesize-and-test*.
- Constraints and resources are never absolutely fixed in human problem solving.

3.1.3. A Functional Model of Interactive Appointment Scheduling

According to the differences between automated and interactive appointment scheduling, we modified the functional model resulting in the model in figure 5. Distribution of all

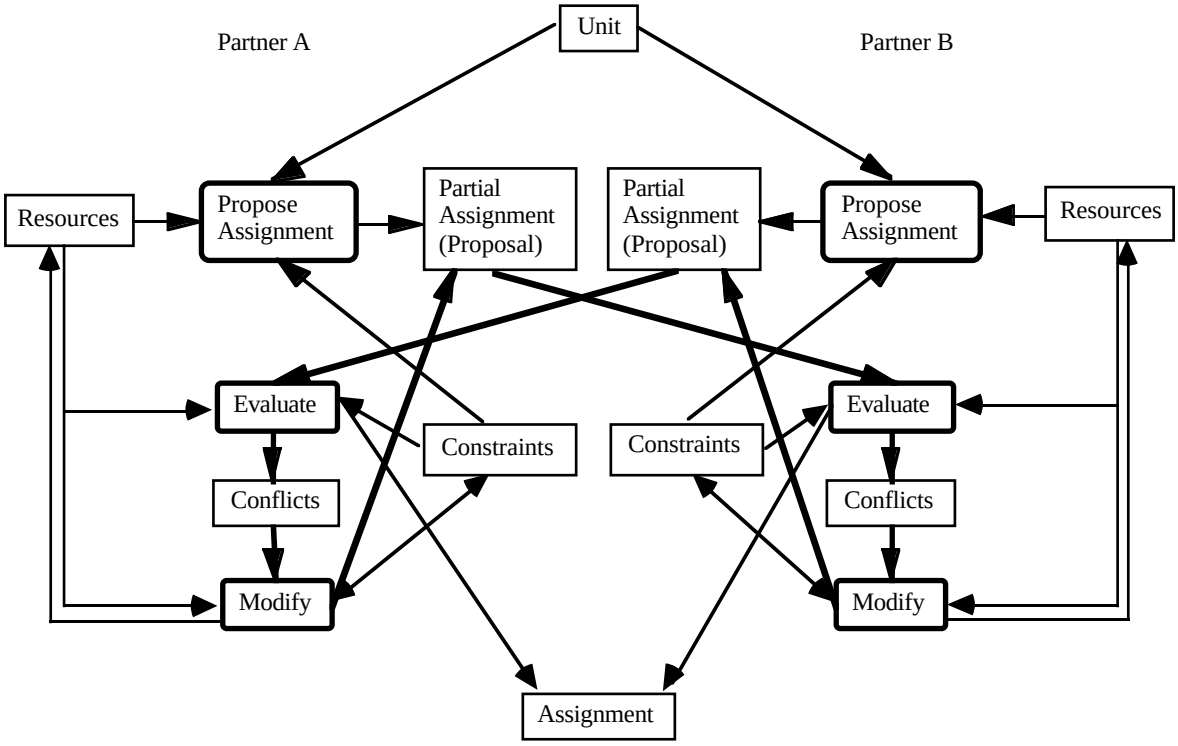

Figure 5: Functional structure of interactive appointment scheduling.

processes to two dialogue participants is the major change. Resources and other data fields each were assigned to both participants, too. Only *unit* containing one appointment that shall be scheduled and *assignment* which is the final solution are common data. Following this strict model, interaction would happen only via *partial assignment*, and after a *proposal*, the other dialogue participant is asked to evaluate this. In reality, all processes and data could be communicated. For example, *evaluations* or *conflicts* are added to the constraints of the interaction partner when they are verbalised. *Constraints* reduce the problem space. They may consist of preference rules for evaluation or describe plans for the choice of further proposals and may be relaxed during *modify*. Furthermore, it is possible to modify *resources* or, in other words, the calendar.

The question in which data box to store information coming from a dialogue participant is not central at this point and would be more interesting for building user models. Here we are interested in the information flow and the move through the functional model. All dialogues and almost all the turns follow the *propose-evaluate-modify* cycle. So we would investigate these processes a little bit more. An analysis of the dialogues from the corpus shows that these processes are not always trivial. They may have an internal structure or at least there is a set of alternative realisations.

propose assignment

This process serves to introduce a working hypothesis or a suggestion. Working hypotheses and suggestions are dependent on each other in that a hypothesis always denotes the next bigger unit, which comprises the suggestion. Reversely interpreted this means that every suggestion comes along with an implicit hypothesis.

evaluate

Evaluate means an acception or rejection according to the own calendar. In case of acception, there may be further modification in modify. In case of rejection, the whole hypothesis may be evaluated, i. e. checking all alternatives within that hypothesis against the own calendar. Acception may also be implicit by repeating the actual suggestion, or by adding further facts, or by further specification. Repetition may also secure understanding.

modify

In case of acception a suggestion may be further specified and thus result in a new suggestion, or in case of a fully specified suggestion an acception indicates a final fixing. A specification without a previous evalulate is an implicit acception.

If a suggestion was rejected, it may be modified without changing the working hypothesis leading to a new suggestion. If the working hypothesis should be rejected, it must be modified.

If it is not possible to change the working hypothesis or suggestion any more, constraints may be modified, or there may be a demand to the dialogue partner to change his or her calendar or constraints.

It is also possible to do nothing within modify and to return initiative back to the dialogue partner.

3.1.4. Implications for our system

Knowledge about the functional structure of interactive appointment scheduling and about subprocesses is an important guide for the development of processes in the avised spoken dialogue system. The major factors that guide understanding in dialogues from this corpus lie within the propose-evaluate-modify cycle. It is important to understand explicit acception or rejection, and to interpret the meaning of a suggestion in relation to the domain, i. e. the calendar. Correct interpretation also allows to recognize implicit evaluation. Every suggestion comes along with a working hypothesis that could be seen as a domain context and that supports interpretation.

Suggestions in our corpus are mainly date and time expressions, but there are some location suggestions, too. The structure and meaning of date and time expressions may be rather complex, thus they are a real challenge for parsing and correct interpretation. Parsing of explicit evaluations and local expressions in the VM corpus are comparatively simple. Consequently, the recognition of date and time expressions was the major task for parsing and semantics.

Conflicts are reasons for rejection, thus mentioning a conflict is an implicit rejection. From the point of view of a spoken dialogue system that makes appointments, it is not necessary to really understand and fully process the reasons given for conflicts. It is sufficient if the system recognizes that there is a conflict and when this conflict holds.

The implication for our system is hence that we have to develep domain specialists for

- *attitudes* that express explicit acception or rejection
- *date and time expressions* and their interpretation according to the domain and to dialogue situation,
- *locations*, and
- *conflicts*.

3.2. Domain Specialists: Parsing and domain-specific meaning interpretation

The main task of the parser and its integrated or subsequent processing of semantics and pragmatics is to indentify those basic units that fit into the domain model. Two major reasons lead us to the decision to apply partial parsing:

- Several attempt to process a complete parse over spoken real world utterances failed. It is more or less impossible to describe a grammar for spontaneously spoken dialogues, especially if we consider a certain amount of errors from the word recognizer.
- For further processing and for adequate system reactions it is not necessary to fully understand and parse an utterance, but it appears to be sufficient to identify and correctly process those islands of important meaning which are central elements of the domain model.

As a conseqence, we apply some partial parser to each turn in a dialogue. Our parsers are independent from each other and units they recognize may not overlap. This technique may be unsufficient in more complex domains, but it is adequate for dialogues similar to our corpus (cf. 2.1).

In order to be able to decide about the structure of date and time expressions, it is sometimes necessary to interpret the meaning of parts of a parse. For this reason, we decided to intertwin parsing and meaning analysis. The combination of a partial parser with its specific meaning analysis is what we call a *domain specialist*. Consequently, the task for every domain specialist is to extend units a far as possible, and as long as they are not semantically inconsistent within one unit.

3.2.1. Implementation of a date and time-parser

Most date and time expressions are extremly underspecified and could only be disambiguated by using domain knowledge, i. e. the calendar and with help of the domain-context model described below.

The parser was implemented using DCGs in Prolog. For a real application, a bottom-up parser should be used, but as long as processing time of our prototype is shorter than real-

time, we do not worry about the formalism. The parsing formalism is not of central concern to our approach.

As a result, the date and time parser returns unit boundaries, a feature matrix that contains those basic notions that were part of the parse string, like weekday(Monday), and an interval-oriented representation of the interpretation. For the simplicity of algorithms and the speed of computation it appears to be easier to expand every expression to sets of intervals and join or intersect these intervals. The use of an interval representation for processing time expressions has been shown by many applications using Allen's time logic [All83]. On the other hand, interval sets do not allow to reconstruct what has really been said, so we keep both representations. This is notably necessary for distinguishing a repetition of "13th April" following a suggestion "13th April" from an addition of knowledge like "Friday", if both denotes the same time interval.

The implemented prototype actually processes only absolute time expressions[4]. This is not a strong restriction, because relative date and time expressions are seldom compared to absolute time expressions in that corpus, and some of them even could not be interpreted. But a meaning interpretation of relative time expressions seems easy, if the actual date and time, or the exact specification of a suggestion, is known. Relative date and time expressions always relate to the actual suggestion and possible values are restricted to the domain-context.

A complex phenomena of date and time expressions are sequences like "every third Tuesday in a month". By using time interval sets, these expressions are correctly interpreted.

Time Parser

The time parser recognizes utterances of points in time as well as time intervals. Time points may be expressed with fuzzy values, which are usually a result of coordination in German (e.g. "um acht, halb neun", which might be translated to "at eight up to half past eight"). Open ended time intervals (before or after) comprise one point in time, while range interval (from-to, between etc.) comprise two points in time. Of course intuitive interval expressions (morning), and their modifications (late afternoon) are processed, too; their meaning is fixed in the domain model.

Date Parser

The date parser recognizes of course year (numbers), months (name), weekdays (name), dates (number of day in a month), but also month numbers (number of month in a year), weeks in a month (e.g. third week of april, last week of april), weeks in a year (e.g. 34th week), and absolute day numbers (day in a year). These basic date expressions may be more or less arbitrarily combined, as far as they make sense.

[4] At this point I would like to thank Jan Amtrup for providing me with a small grammar for date expressions which he and Andreas Hauenstein wrote for the VM corpus. I could not use that grammar, because its structure was not suited for being intertwinned with domain-specific meaning interpretation, but it was helpful because it included further phenomena within time expressions, so that they are covered by our parser now.

Evaluation

The date and time parser correctly recognizes and interprets, i.e. recalls about 95% of all absolute time expressions in the corpus. Precision[5] is almost 100%. A decrease of the word recognition rate from 100% to 90% leads to a 93% correct interpretation (recall) of date and time expressions. This shows that the parser is to some degree robust against errors from the word recognizer. In all but one cases, errors from the word recognizer that did affect the meaning interpretation led to a loss of information, i.e. a more general interpretation, but errors did not lead to a wrong interpretation.

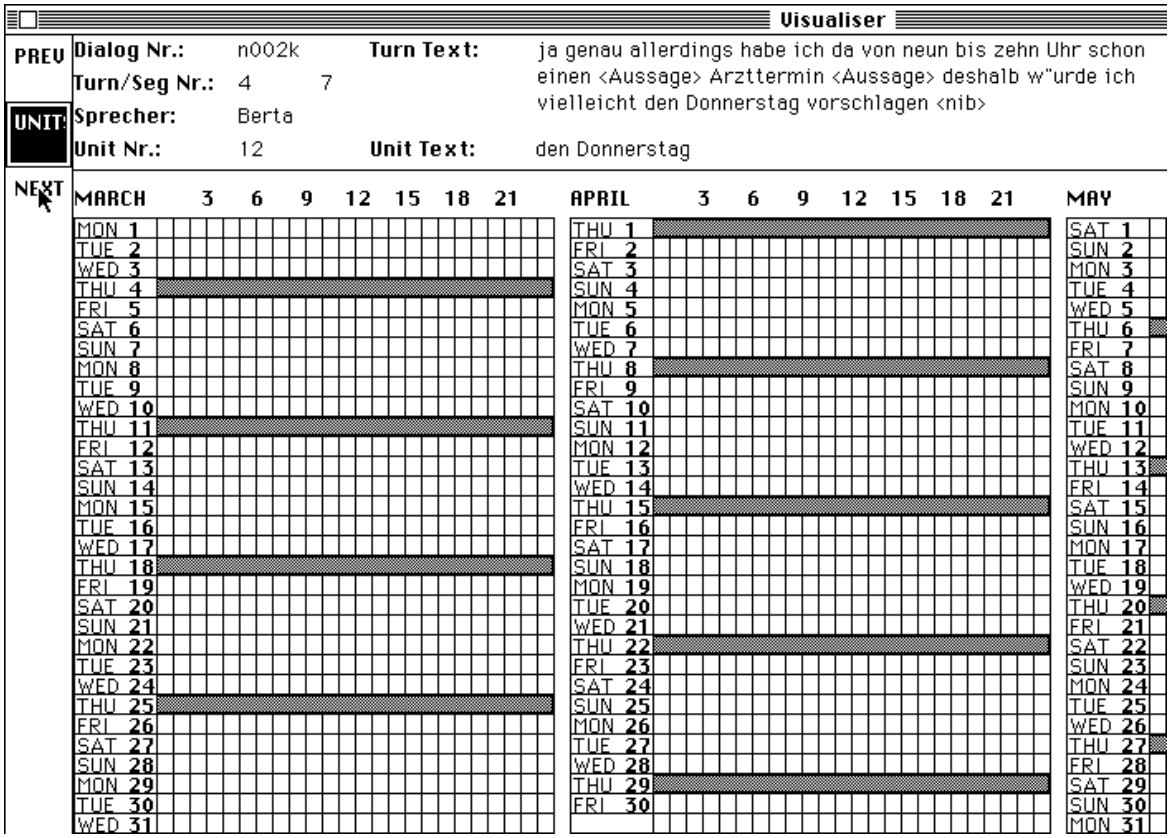

Figure 6: A set of time intervals representing the meaning interpretation of a unit without context

One major problem lies within the parsing results and is not a problem of the date and time parser itself. It is the problem of underspecified date and time expressions. "Thursday" does not tell us which Thursday this is, and thus this word may denote one of all 52 Thursdays of a year. This drastically reduces the overall precision measured as hit rates in a calendar made by a human interpreter following the dialogue. The parser with its integrated meaning interpretation is not capable to disambiguate this. Disambiguation is subject of subsequent processes. We decided to return the set of all time intervals denoted by "Thursday", thus the

[5] Precision is here the number of time expressions, that were correctly recognized minus recognized word sequences that are no time expressions divided through the total number of time expressions and to get percentage multiplied with 100.

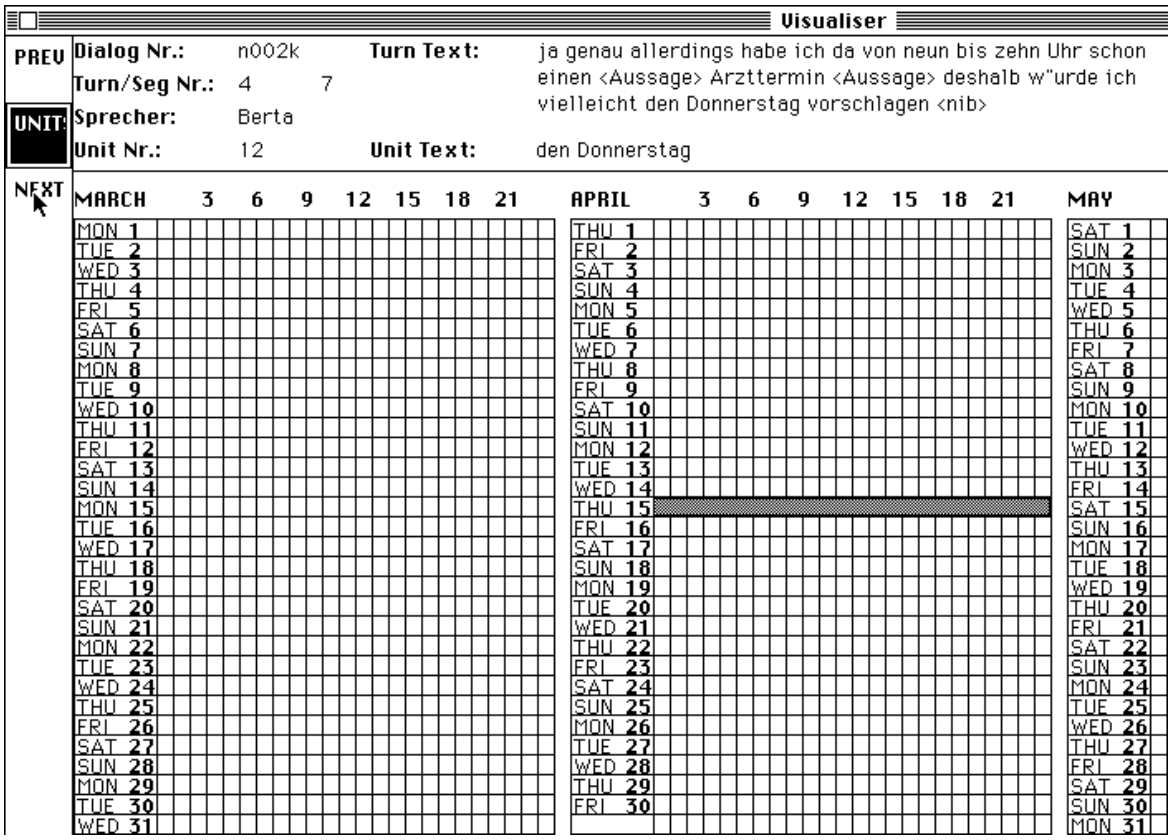

Figure 7: A set of time intervals representing the meaning interpretation of a unit after context restrictions

parsing result is a set that contains the solution. Figure 6 shows a window of the date and time parser visualizing the meaning interpretation of the actual unit "den Donnerstag".

3.2.2. Using a dialogue context model for semantic disambiguation

When talking about dialogue context, we consider just one aspect of discourse: moves in the domain. As we have argued above, every suggestion comes along with an implicit working hypothesis. This hypothesis is the domain-specific dialogue context that will help us to disambiguate results from the parser.

The dialogue context also has a twofold representation, just like the parse results: The first representation is the set of all possible intervals (e.g. 42nd week, April) that can be intersected with the result of a date and time parse (e.g. Monday) yielding an interpretation that is a more restricted than the original interval. Additionally, there is a feature-based representation that helps to denote context. The feature-based representation is maximally specified, which means that it contains all features denoting intervals which comprise the most specific interval. The feature hierarchy describes levels of specificity. With these two representations it is easy to test whether a suggestion lies within the actual context and on the appropriate level of specificity or whether the context must be shifted. Figure 6 compared to figure 7 shows the utility of a domain context.

The application of the context model to disambiguate parse results was a major break-through in the interpretation of the meaning of date and time expressions and shows the importance of domain knowledge and of appropriate domain models for understanding and

processing of natural language in restricted and task-oriented domains. Figure 8 shows the results of the analysis of the best-word-path output from 10 appointment schedulings from the VM (Blaubeuren) corpus. The first line shows the number of units with date and time expressions, while the second line contains the number of time intervals that is returned by the parser. This means that as an average for every time expression the parser generates about 30 possible time intervals (3055%) thus having a large ambiguity and a very low precision. Using context, the ambiguity could be reduced to the factor 2.3 or 237%. Comparing the number of intervals without and with context disambiguation (last column), an amount of 7.8% remains after context disambiguation. But more important for understanding is the number of non-ambiguous interpretations that could be achieved using the context model. From the date and time expressions recognized by the parser, 25.6% could be interpreted non-ambiguously. When applying context disambiguation, 84.5% of the date and time expressions have a unique interpretation.

	absolute numbers	% relative to line 1	% relative to line 2
Date and time units recognized by parser:	129	100.0%	
Time intervals returned by parser:	3941	3055.0%	100%
Time intervals with context disambiguation:	307	237.0%	7.8%
Non-ambiguous interpretations without context :	33	25.6%	
Non-ambiguous interpretations with context :	109	**84.5%**	

Figure 8: Reduction of meaning interpretation ambiguity when using domain-specific dialogue context

3.2.3. Parsing explicit attitude expressions

To identify attitudes, we just used a keyword spotter, which provides sufficient and reliable information.

3.2.4. Parsing location expressions

At the moment we use very simple parse structures and keyword spotting to parse location expressions. This leads to satisfactory results in our corpus, because location expressions are simple there.

3.2.5 Parsing conflicts

Temporarily, keyword spotting is applied to identify conflicts, and results are good, but this solution is very unsatisfactory for a larger amount of data since conflicts differ greatly. We intend to implement a meaning interpretation of what constitutes a conflict, based on an ontology.

3.3. Prosodic Information and Segmentation

Although a turn in a dialogue is a natural unit, turns often are a sequence of different segments each having its own function within the dialogue. We cannot use grammatical information to divide turns into segments, because spoken language often comprises non-

grammatical parts and - again - there are no complete parsers. Prosodic information provides information for segmentation. But it also provides information on sentence modifiers and accents.

3.3.1. The role of prosodic information for dialogue processing

If one is not familiar with prosody, one might suppose that prosodic information, notably *accent,* would also mark important content in a turn, but it turns out to be the opposite. While keywords from the domain are not accentuated, words expressing acception, rejection, negation or question are accentuated. Thus *accent* mainly provides information on social interaction. Recently, first results on the recognition of a prosodic *focus* showed, that it marks pragmatically important words [Els96]. Prosodic focus has not yet been integrated into DIA-MOLE.

```
<P> ja prima     dann lassen Sie uns   doch noch einen Termin
        L+H* H-H%       H*           L-                  H*
             B3                      B2
        PA              NA                               PA

ausmachen     <P> wann w"ar's Ihnen denn recht     <P> <#Klicken> <P>
          L-L%                  H*         L*  H-H%
          B3                                    B3
                                NA         PA  ?
```

Figure 9: Result of a manual prosodic transcription[6]

We use prosodic information in two ways: for segmentation and for tagging segments with information on sentence modifiers. Figure 9 shows the result of a manual prosodic transcription following [Pie80, Uhm91, Sil92, Fèr93, Rey94]. In contrast to those symbolic data, the output of an accent recognising system provides time intervals in which an accent occurs with some confidence value. There are certain differences between manual transcriptions and the output of a prosody recognizer with time boundaries and probability values (cf. figure 2), which might result from human interpretation within the manually tagged data. For an engineering approach to a spoken dialogue system, it is indispensible to use only automatically derived data. Prosodic data contain *accent, phrase boundaries (B0,B2,B3,B9),* and sentence modalities *continuation, question, assertion* [Str95].

When analysing prosodically labelled dialogues, we realized that for further processing not these labels themselves are useful, but combinations of them as well as their relation to information from other knowledge sources. B0 - word boundaries, B9 - non-grammatical phrase boundaries and B2 - intermediate phrase boundaries were irrelevant for our purpose. B3 - full phrase boundaries on their own appeared not to be useful either to get meaningful segments of a turn - there are too many B3 boundaries. If only those B3 are considered that appear together with some sentence modality events, the information content of this combination concerning useful segment boundaries is much higher.

[6] H* is a high level accent, L* is a low level accent, % marks the end of a phrase, B3 are phrase boundaries, PA are phrase accents and NA are secondary accents

Yet again, there are also too many sentence modality events, to process them. So we decided only to consider those in combination with B3. Furthermore, we modified the interpretation of *continuation*. Figure 10 shows those *prosodic events* that we add as prosodic modifiers to the stream of words. Only those data events are considered with a probability higher than 0.5 are considered.

Raw data		Prosodic Events
phrase boundary B3 + continuation	->	nothing
phrase boundary B3 + assertion + accent	->	<continuation>
phrase boundary B3 + query	->	<query>
phrase boundary B3 + assertion	->	<assertion>

Figure 10: Modification and interpretation of prosodic data

3.3.2. Rules of segmentation

Our *prosodic events* are sufficiently rare and well placed, so that they may serve as an information source for turn segmentation. This does not mean that each prosodic event leads to a segment boundary. Sometimes prosodic events tend to break apart date and time expressions or they would produce segments without any useful meaning. We developed the following tentative strategy to segmentation:

Prosodic events that lie within some unit boundaries are neglected because units were already recognized as being meaningful by the domain specialist. Then we combine to a segment a series of units that are recognized by a parser, up to a prosodic event after one unit. We look further, skipping non-domain-relevant information until the beginning of the next recognized unit or until the end of the turn. The last prosodic event found in this sequence is adopted as segment boundary and leads to a prosodic tag for that segment. Figure 11 shows the stream of words of turn n002k002 with integrated prosodic events. Looking at the text of unit 5, one can see that the prosodic event lying within the unit boundaries is not considered any more.

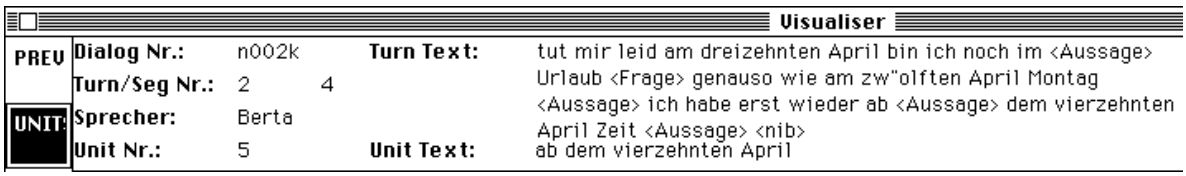

Figure 11: Turn text with prosodic events and dropping them within one unit

In most cases this works well. There are some cases, notably with accentuated connectives in a turn, where this strategy fails, but it is less than 5%. These errors would disappear when recognising connectives as own units with meaning. We did not address this yet, because it concerns inter-segmental meaning.

Failure means that the segmentation module produces segment boundaries that would have not been made by humans. We compared our segments with a human segmentation that was

built up in the context of RST labeling. The person was advised to apply RST even to subphrases level, so his segment are smaller and sometimes even break units recognized by the date and time parser. Nevertheless, most of the automatically derived segment boundaries meet exactly boundaries set by the student.

Figure 12 shows two manually segmented turns, and figure 13 contains the same turns segmented by the system. The second dialogue contains one of those typical errors ("weil ich").

```
guten Tag Herr Dingler (hello Mr. Dingler)
erinnern Sie sich noch wir m"ussen einen Termin ausmachen (do you remember, we have to
schedule an appointment)
ich w"urde vorschlagen (I would suggest)
wenn ich hier auf meinen Terminkalender schaue (if I look in my calendar)
am Dienstag den neunten (Tuesday ninth)
das w"are mir recht (would be all right)
am Nachmittag (in the afternoon)

guten Tag sch"onen guten Tag Herr Schaaf (hello hello Mr. Schaaf)
leider pa"st es mir am neunten "uberhaupt nicht (I am sorry, the ninth does not fit)
weil ich vom neunten bis zum elften au"ser Hause bin (because I am not here)
wir m"u"sten einen andern Termin nehmen (we should try another date)
w"urde Ihnen der sechzehnte passen am Nachmittag (what about the sixteenth in the
afternoon)
```

Figure 12: Students segmentation for RST-tagging

```
11  8 0      5930   guten Tag Herr Dingler <Weiterfuehrung> <nib> erinnern Sie sich noch
                    wir m"ussen einen Termin <Aussage> ausmachen
12  8 5930   10360  ich w"urde vorschlagen wenn ich hier auf meinen Terminkalender
                    schaue am Dienstag den neunten <Aussage>
13  8 10220  12210  <Weiterfuehrung> das w"are mir recht am Nachmittag

14  9 0      5520   guten Tag sch"onen guten Tag Herr Schaaf leider pa"st es mir am
                    neunten "uberhaupt nicht weil ich
15  9 5520   8230   vom neunten bis <Aussage> zum elften <Aussage> au"ser Hause bin
                    <Frage>
16  9 8230   12600  wir m"u"sten einen andern Termin nehmen w"urde Ihnen der sechzehnte
                    passen am Nachmittag
```

Figure 13: Automatic segmentation

To further validate our segmentation, we compared them with segments made for dialogue act labeling in the VM project [Jek95]. At the moment this has been done manually for exactly one dialogue, but in the future this should be done automatically. Results of this comparison are:

- In 32 turns out of 59 turns segments are identical.
- Furthermore, both segmentations contained
  - 18 equally placed segment boundaries
  - 27 segment boundaries in VM but not in DIA-MOLE (for an example see Figure 14)
  - 26 segment boundaries in DIA-MOLE but not in VM (for an example see Figure 15)

Differences in segment boundary placement can be explained in terms of different factors. While VM segmentation is mainly concerned with identifying single speech or dialogue acts, we allow multiple basic acts to be in one segment, as long as their is no phonetic evidence for a boundary and as long as they all come from different domain specialists. Segment features that can be seen in Figure 14 and 15 are explained in the following section.

```
<P> ja prima , @(FEEDBACK_ACKNOWLEDGEMENT AB)
dann lassen Sie uns doch noch einen <!einn> Termin ausmachen <P> . @(INIT_DATE AB)
wann w"ar's Ihnen denn recht <P> <#Klicken> <P> ? @(REQUEST_SUGGEST_DATE AB)

0 0 290 3720     [attitude(POSITIVE), assignment(ASSIGNMENT)] [phonMod(<Frage>)] ja
                 prima dann lassen Sie uns doch noch einen Termin ausmachen wann w"are
                 es Ihnen denn recht
```

Figure 14: Segment boundaries in VM but not in DIA-MOLE

```
<#Klicken> <P> gut <A> sollen wir's dann gleich am Montag <A> den<Z> dritten Mai machen
                 ? <P> <Schmatzen> vielleicht um halb vier <#Klicken> <P> ?
                 @(SUGGEST_SUPPORT_DATE AB)

16 10 640 970    [attitude(POSITIVE)] [phonMod(<Aussage>)] gut <Aussage>
17 10 1240 5770  [value(alternative), specificity(keep)] [phonMod(<Frage>)] sollen wir
                 es dann gleich am Montag den dritten Mai machen <Frage>
18 10 6210 6910  [value(same), specificity(specify)] [phonMod(<Frage>)] um halb vier
```

Figure 15: Segment boundaries in DIA-MOLE but not in VM

3.4. Assigning features to segments

Before being able to classify segments and to ascribe some kind of speech act to them, we have to assign some features to each segment. The implementation of prosodic feature assignment is very easy: We just use as a feature and value the last prosodic event that could be found in a segment.

Domain feature assignment is more complex. One aspect of the domain features is the *attitude* which could be directly taken from the result of the attitudes domain specialist. Comparing a suggestion with the previous suggestion and the actual context leads to two other features: *specificity* marks whether a suggestion is on the same level of specificity like its predecessor (e.g. Monday -> Wednesday), or whether it is more specific (in the morning) or general (April). *value* denotes whether the hierarchy in the context has been changed (e.g. if we move from *Monday 4th* to *Tuesday 12th,* we also change the week).

```
attitude (POSITIVE | NEGATIVE)
location (LOCAL | GLOBAL)
conflict (CONFLICT)
value (SAME | NEW | ALTERNATIVE) (date and time)
specificity (SPECIFY | GENERALIZE | SAME) (date and time)
assignment(ASSIGNMENT)
phonMod (FRAGE | AUSSAGE | WEITERFÜHRUNG)
```

Figure 16: List of segment features that are possible in this implementation

4. EVALUATING LEARNED DOMAIN-SPECIFIC DIALOGUE ACTS

4.1. Investigating the plausability of our approach

In a first experiment, we applied the conceptual clustering algorithm COBWEB [Fis87] to tagged segments from 3 appointment schedulings from the Blaubeuren corpus of VM. Implemented segment features were *attitude, value, specification* and *phonMod*. COBWEB incrementally builds up a classification tree. In this small experiment we wanted to test whether the learning algorithm produces meaningful classes at all. We kept it small because an analysis of the classification tree was very work-intensive at that stage: All segments had to be reproduced, and one had to ascribe some meaning to classes by analysing all segments within their dialogue context. Although we should have considered more information by adding some more domain specialists (e.g. for greeting and conflicts), with this prototype, we were already able to show that profound interpretation of domain meaning and prosodic information will be sufficient to learn meaningful, domain-specific speech acts. We obtained the following similarity classes of domain-specific or at least task-specific dialogue acts on the third level of the binary tree (Figure 17). The percentage distribution of segments into classes is given. The numbering system in the first column indicates the hierarchy.

```
1.1  24.4%  Assurance of understanding
1.2  31.2%  Confirmation, objection and suggestion which all keep
            the actual context
2.1  20.0%  Asserted alternative suggestions
2.2  24.4%  Alternative suggestion that keep the level of specificity
            or that generalise
```

Figure 17: Similarity classes of segments from experiment 1

As mentioned in the introduction, there are many theories for structuring discourse. As we have several dialogues from the our corpus tagged with RST, we wanted to check whether there are some correspondences between learned classes for domain-specific speech acts and RST relations. Although RST tags were subject to human interpretation, the following correspondences could be observed:

- 2/3 of the nuclei of the alternative relations start from the class of asserted alternative suggestions,
- 3/4 of the satellites of the alternative relations lie within non-asserted suggestions,
- all nuclei of justification relations lie within the class 2 of alternative suggestions,
- and half of the justification satellites are in class 1.1

Figure 18: Correspondence between RST-relations and DDA classes from Figure 17

We verified our results in a second experiments with 10 appointment schedulings from the Blaubeuren corpus. The classes learned by COBWEB are a little bit different compared to

those from experiment 1 and the total number of segments is big enough for a finer-grained analysis (Figure 19). Again these classes were checked against RST-relations. There was significance for the correspondences in Figure 20.

1.1.1	22.8%	More specific suggestions, Repetions, Generalisations
1.1.2	15%	Specifications
1.2.1	6.4%	Acception with own suggestion and query for confirmation
1.2.2	2.9%	Suggestions with query for confirmation
2.1.1	10.7%	Alternative suggestions with query for confirmation
2.1.2	22.1%	Alternative suggestions
2.2.1	8.6%	Acception with query for confirmation / Rejection
2.2.2	11.4%	Acception

Figure 19: Similarity classes of segments from experiment 2

- 3/4 of nuclei of the alternative relation are within class 2.1, more than 1/2 in 2.1.2. (the other 1/4 lies within class 1.1)
- allmost all justification relations are connected either to class 1.1 or class 2.1. Both classes often have an intra-segmental justification relation.
- most of the nuclei of the justification are connected with class 2.1, while most of the satellites are connected to class 1.1.
- all nuclei of the summary relation are within class 2, while all satellites are within class 1.1.1

Figure 20: Correspondence between RST-relations and DDA classes from Figure 19

These correspondences gave us the impression that the interpretation of classes seems right, and we postulate that a sophisticated domain interpretation in combination with prosodic information is the key to understanding and structuring dialogues.

For the following experiments, we added domain specialists for conflicts, assignments and location. We implemented the improved and extended learning algorithm with all the features described in 2.4.

4.2. Validation by DDA interpretation and comparison with VM-dialogue acts

Figure 21 shows the segment hierarchy learned from 10 appointments made in 205 segments. The probability of 1.0 was assigned to all features. One parameter used in the learning phase was the maximal equality boundary set to 0.9. This means that if a case is integrated into a class yielding a category utility that is higher than the maximal equality boundary value, it is not further integrated into any subclass.

Before automatically labeling segments with class names, all segments have again to be classified using the complete hierarchy. For this classification some other parameter apply, which were described in detail in section 2.4. They hinder the algorithm to return too specific classes. The *minimal case number* is 9, *maximal case prediction* is set to 0.8, and a *good prediction gain* is 0.5. With these values, all segments are classified into 15 classes.

Finally, segments associated to these classes were compared with manually labeled dialogue acts from the VM project (see [Jek95] and the hierarchy in Appendix B). Although their number of segments on the same dialogues is with 220 very similar to our number of 205,

segment boundaries are not identical (see 3.3.2), and words may be different because of errors in transcription or word recognition. For this reason, we implemented a word sequences synchronizer and a comparison algorithm that considers overlapping segments between both segmentations. An overlap-matrix table is shown in Appendix A. It is apparent that the number of processed cases is not yet high enough to make statistically well-founded conclusions, nevertheless we are able to provide an interpretation of the classes learned by analysing their features, and we point out some frequent relations to VM dialogue acts that are marked by capital letters:

- class 4540: It consists of prosodically unmarked positive evaluations without any domain information. It is evident that these segments mainly correspond to ACCEPT_DATE or CLARIFY_ANSWER.
- class 6756: Explicit positive evalutions without domain information, but in contrast to 4540 prosodically marked as assertions. Correspondence again mainly to ACCEPT_DATE and CLARIFY_ANSWER, but also FEEDBACK_ACKNOWLEDGEMENT. This class contains also most GREET and THANK_INIT, which are positive expressions without any domain information. Standard greeting expression have not been analysed by a domain specialist and are not added as features to segments in DIA-MOLE.
- classes 6565, 6632, 5424, 6659, 6674: These classes have in common that segments perform some actions in the domain. These actions are repetition, generalisation or specification of a suggested date, but not alternative suggestions. In contrast to the group of alternative suggestions (6766, 6260, 6486, 6780) this group contains many segments corresponding to ACCEPT_DATE and REJECT_DATE. Rejections are indirectly given by date information.
    - class 6565: A specified date is supported by prosodical query and assertion markers. Assertion mainly support a suggestion (SUGGEST_SUPPORT_DATE), while query markers indicate a query for assent (an implicit REQUEST_COMMENT_DATE) or a query for approvement of given reasons in the case of rejection of a specified date.
    - class 6632 has no prosodic markers and indicates sole domain information.
    - class 5424: Specification of a date with explicit evaluation. If the evaluation is positive, there is a correspondence to CLARIFY_ANSWER and CLARIFY_STATE, if it is negative segments correspond to REJECT_DATE.
    - class 6659* contains all those segments that could not be classified further down, i.e. those that belong to 6659 without those from 6674. For this reason this class is marked with a asterisk. All segment are marked as assertions and repeat previously mentioned dates. This class can be divided into two distinct subclasses: If the repetition is on the same level, segments overlap with SUGGEST_SUPPORT_DATE; if repetitions are more general, they correspond to REJECT_DATE. There are no explicit evaluations in 6659*.

```
Number of cases processed: 205
root <-- node_4540 = 12
     <-- node_6648 = 66 <-- node_6565 = 15 <-- node_1653 = 4
                        <-- node_6632 = 19 <-- node_4096 = 6
                                           <-- node_6066 = 4
                                           <-- node_6640 = 7
                        <-- node_6659 = 32 <-- node_5424 = 9 <-- node_4533 = 3
                                                             <-- node_5431 = 5
                                           <-- node_6607 = 8 <-- node_6205 = 4
                                                             <-- node_6613 = 4
                                           <-- node_6674 = 15 <-- node_4031 = 5 <-- ...
                                                              <-- node_5475 = 4 <-- ...
                                                              <-- node_6686 = 6
     <-- node_6756 = 20 <-- node_2323 = 2
                        <-- node_6290 = 15
     <-- node_6766 = 39 <-- node_566 = 2
                        <-- node_5828 = 6
                        <-- node_6260 = 9
                        <-- node_6486 = 10 <-- node_2100 = 2
                                           <-- node_6495 = 8
                        <-- node_6780 = 12 <-- node_6136 = 3 <-- node_6145 = 2
                                           <-- node_6541 = 4 <-- node_6546 = 3
                                           <-- node_6788 = 5
     <-- node_6870 = 68 <-- node_6086 = 9 <-- node_2731 = 3
                                          <-- node_6097 = 5 <-- node_6104 = 3
                        <-- node_6806 = 42 <-- node_5760 = 9 <-- node_4856 = 4 <-- ...
                                                             <-- node_5363 = 2
                                                             <-- node_5771 = 2
                                           <-- node_6815 = 33 <-- node_1296 = 3 <-- ...
                                                              <-- node_4276 = 5 <-- ...
                                                              <-- node_5964 = 7 <-- ...
                                                              <-- node_6850 = 5 <-- ...
                        <-- node_6881 = 17 <-- node_5714 = 5
                                           <-- node_6747 = 6
                                           <-- node_6891 = 6
```

Figure 21: Learned DDA hierarchy

- class 6674: This class is characterized by repetitions with explicit evaluations. This may be queries for clarification and their answers, and acception and rejection of a suggestion

- classes 6766*, 6260, 6486, 6780: These classes have in common that they introduce alternative suggestions. They hardly correspond to ACCEPT_DATE or REJECT_DATE, which seems evident, because it does not make sense to introduce a new suggestion in order to explicitly reject a previous one. The bigger the difference between a previous suggestion and the alternative the less probable is an acception.
- class 6086: Specified date and time expressions and conflicts let this class significantly correspond to GIVE_REASON.
- class 5760: Mentioning the task of appointment scheduling combined with explicit positive evalutions and a prosodic query mark make a correspondence to INIT_DATE obvious.
- class 6815: This class is very inhomogenous and we have to look into its subclasses.
  - subclass 6850: A mixture of date expressions and locations, usually prosodically marked.

- subclass1296: Prosodically marked conflicts without any other information
- subclass 4276: Explicitly positive evaluated alternative suggestions after REQUEST or CLARIFY_QUERY, or as a final ACCEPT.
- subclass 5964: Although it is hard to interpret the feature structure of this class, most segments correspond to SUPPORT_EXCLUDE_DATE and some to REQUEST_COMMENT_DATE.

- class 6881: This class comprises most locational information.

In contrast to manually given dialogue act labels, learned DDA classes distinguish for example explicit and implicit rejection. This means that they do not characterize the illocutionary force by interpretation, but their illocutionary force relies on acts in the domain and its task model. It seems that a hierarchy resulting from processing a larger amount of data will get a finer structure than manual dialogue act labels and perhaps some of these finer classes will be interchangeable from the point of view of sequences of acts.

4.3. Evaluation of DDAs by their predictability

In a third experiment we tried to evaluate the quality of the learned DDAs by testing whether they are well suited for dialogue act prediction. Up to now we could only use nine dialogue sessions containing 19 appointment schedulings, thus our results are a first attempt and not very reliable. For prediction we used simple 3-grams.

We will compare our approach with results reported by Reithinger and Maier [Rei95]. They examined prediction rates on manually labeled dialogue acts in VM, corresponding to those shown in Figure 14 or 15. For training, they used 41 dialogues with 2538 speech acts, and the test set contained 81 dialogues. In their test over all turns they reported a hit rate for the first prediction of 29% and a 45% hit rate within the first or second prediction. Reithinger and Maier report higher hit rates if they skip all those turns in their test set that are not part of their main dialogue network and that can occur at any point of the dialogue.

For learning, we used four dialogue sessions n001k, n002k, n005k, n007k, consisting of 120 turns. Input to the Dia-MoLE dialogue-act learner came from a word recognizer with a word accuracy of less than 90%. Prosody and domain knowledge were added, and classes of DDAs have been learned with our unsupervised learning algorithm. Classifying again all segments of these four dialogues in terms of the dialogue act hierarchy yields segments labeled with DDAs. The sequence of these labeled segments was used to build up a 3-gram model.

Test dialogues are processed with the same modules and are based on noisy data, e.g. 90% word accuracy, too, but they contribute neither to the dialogue act hierarchy nor to the construction of the 3-gram model. In Figure 22, hit rates for different dialogue sessions are presented. The first column shows the dialogue label (Dialog), the second contains the absolute number of predictions (Pred), the third column shows the hit rate using the most probable prediction, and the hit rate in the last column relies on the most probable two predictions. As we have argued in section 2.5, the dialogue act hierarchy may be over-

specified that means that some classes which are very similar but which could be well distinguished by data in spoken language may be interchangeable from the viewpoint of prediction. For this reason, we generalized both prediction class and dialogue act class assigned to segments. The hit rate for the generalized predictions are in column 4 for the most probable prediction and in column 7 for the second-most probable prediction. The fifth column contains the hit rate for a double generalization.

Dialog	Pred	1st	1st^	1st^^	2nd	2nd^	1st+2nd
1,2,5,7	205	36	39	49	18	20	54
n003k	30	16	16	33	20	23	36
n006k	54	9	14	31	14	24	23
n009k	30	20	20	30	3	7	23
n011k	28	21	25	32	7	10	28
n018k	54	18	22	39	8	13	26

Figure 22: Prediction hit rates in DIA-MOLE

Although Dia-MoLE just reaches about 60% of the hit rate compared to manually labeled dialogues, results of dialogue act prediction seem promising since the learning set was very small, and we have used noisy data. We expect better results when having a larger learning set. This is subject of future work.

At the end we have to make one remark concerning the expressiveness of prediction hit rates concerning the quality of more or less domain-specific dialogue act classes: In our opinion, this is a weak measure because it depends heavily on the branching factor in the domain. In complex domains, there are more branches, and thus there is a lower predictability. Additionally, prediction is only one application of dialogue acts. The applicability of dialogue acts in a system environment for several other purposes might engender other criteria that are more important for evaluation.

5. CONCLUSION

Our approach to learning dialogue models is influenced by the idea of dialogue engineering in the context of a spoken dialogue system. In contrast to many other recent applications of machine learning to discourse or dialogue phenomena [Hir92, Lit94, Sie94, Leh94], we will not rely on hand-labeled linguistic data, but only on data which can be automatically generated in such a system. This had two positive effects, first, we avoided the enormous effort to label dialogues, and as a consequence of this, second we were able to use very large amounts of data for learning.

A further difference to other approaches using machine learning for discourse or dialogue modelling is that we did not rely on some given discourse theory with a given set of classes of dialogue acts or similar kinds of units. Instead, we used machine learning to derive domain-specific and task-specific classes from a set of example dialogues. These DDA classes are based on real data that are available in a dialogue-system environment, and on domain knowledge. Thus a future dialogue component using DDAs will be directly applicable within this system.

Preprocesses filter and interpret data yielding some kind of relevant data as segment features. The engineer's choice of interpreted data and the underlying domain model strongly influence the DDA hierarchy. We consider this relation between DDAs and the domain model as an important benefit of our approach to pragmatics engineering in spoken dialogue systems, although linguists might miss their theories.

The decision which data are relevant at all is subject to human engineering. Consequently, DIA-MOLE does not allow to invent new information categories in speech data that might be relevant, because it does not know about other data than segment features. But we think a high-level domain- and interaction-oriented abstraction is essential for learning domain-specific dialogue acts at the pragmatic level.

We gave an outline of an architecture for learning dialogue acts within DIA-MOLE and introduced a first implementation. We think our results look very promising so that we will add some more domain specialists and apply the learning algorithm to a larger amount of data. We suppose that using more domain-specific knowledge will improve the expressiveness of our classes. This will be subject to further research.

DIA-MOLE allows to weigh the relevance of different segment features for learning DDAs. This has not yet been used and is a possible point for manual intervention by the engineer. Automatic determination of the relevance could be realised by a meta-learning process that modifies the relevance parameters and evaluates the resulting DDA hierarchy according to some evaluation criteria, e.g. compactness. Another extension to the actual DIA-MOLE system would be the integration of some means for planning dialogues according to some automatically deduced model. Meta-learning as well as dialogue planning have not been elaborated, yet. They may become subject of future research.

Our domain-specific dialogue acts should also be used to improve the word recognizer of our department by training different language models for each class of DDAs. Additionally, we want to use our system for investigating the importance of different dialogue and discourse structures and their coincidence.

REFERENCES

Ale94. Alexandersson, J., Maier, E., and Reithinger, N. A Robust and Efficient Three-Layered Dialogue Component for a Speech-to-Speech Translation System. In *Proc. of the EACL,* Dublin, 1994, pp. 15.

All82. Allen, J.F., Frische, A.M., and Litman, D.J. ARGOT: the Rochester Dialog System. In *Proc of Annual Meeting of the American Association of Artificial Intelligence,* 1982, pp. 66-70.

All83. Allen, J.F. Maintaining knowledge about temporal intervals. *Communications of the ACM* , 26 (11) (1983), 832-843.

Ang91. Angele, J., Fensel, D., Landes, D., and Studer, R. KARL: An executable language for the conceptual model. In *6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Alberta, 1991, pp. 1.1-20.

Bad94. Bade, U., Heizmann, S., Jekat, S., Kameyama, S., Krause, D., Maleck, I., Prahl, B., and Preuß, W. Wizard-of-Oz-Experimente mit dem VERBMOBIL-Simulator. Tech. Rept. VM-Memo-24-94, Verbmobil, 1994.

Bat93. Batliner, A., Kompe, R., Kießling, A., Nöth, E., Niemann, H., and Kilian, U. The prosodic marking of accents and phrase boundaries: Expectations and results. In *NATO ASI: New Advances and Trends in Speech Recognition and Coding*, Rubio, A.J., Bubion, Granada, 1993, pp. 89-92.

Bil91. Bilange, E. A Task Independent Oral Dialogue Model. In *Proc. European Chapter of the ACL*, 1991, pp. 83-88.

Bre86. Breuker, J. and Wielinga, B. Models of Expertise. In *Proc. of the European Conference on Artificial Intelligence ECAI-86*, 1986, pp. 306-318.

Bre94. Breuker, J. A Suite of Problem Types. In *CommonKADS Library for Expertise Modelling - Reusable Problem Solving Components*, Breuker, J.; Van de Velde, W., IOS Press, Amsterdam, 1994, pp. 57-86.

Bun89. Bunt, H. Information Dialogues as Commmunicative Actions in Relation to Partner Modelling and Information Processing. In *The Structure of Multimodal Dialogue*, North-Holland, Amsterdam, 1989, pp. 47-73.

Bun94. Bunt, H. Context and Dialogue Control. *Think* , May 94 (1994), 19-31.

Car88. Carpenter, G.A. and Grossberg, S. The ART of Adaptive Pattern Recognition by a Self-Organzing Neural Network. *IEEE Computer* , 3/1986 (1988).

Car90. Carpenter, G.A. and Grossberg, S. Adaptive Resonance Theory: Neural Network Architecture for Self-Organizing Pattern Recognition. In *Parallel Processing in Neural Systems and Computers (10th Cybernetics DGK)*, North Holland, Amsterdam, 1990, pp. 343-349.

Car91. Carberry, S. *Plan recognition in Natural Language Dialogue*, MIT Press, Cambridge, Mass. (1991).

Chu94. Chu-Carroll, J. and Carberry, S. A Plan-Based Model for Response Generation in Collaborative Task Oriented Dialogues. In *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle, 1994.

Coh87. Cohen, P.R. and Levesque, H.J. Rational Interaction as the Basis for Communication. Tech. Rept. CSLI-87-89, Center for the Study of Language and Information - Report, SRI, Xerox PARC, Stanford Univ., 1987.

Coh90. Cohen, P.R. and Levesque, H.J. Intention Is Choice with Commitment. *Artificial Intelligence* , 42 (1990), 213-261.

Cun95. Cunningham, H., Gaizauskas, R.J., and Wilks, Y. A Genereal Architecture for Text Engineering (GATE) a new approach to Language Engineering R&D. Tech. Rept. CS-95-21, Research memo, University of Sheffield, UK, Dept. of Computer Science, 1995.

Dar94. Daradoumis, T. Building an RST-Based Multi-Level Dialogue Context and Structure. In *Proceedings of the 10th Conference on Artificial Intelligence for Applications*, 1994, pp. 465-466.

Eck96. Eckert, W. Understanding of Spontaneous Utterances in Human-Machine-Dialog. In *Proc. Twente Workshop on Language Technology, Dialogue Management in NL-Systems*, LuperFoy, Nijholt, Z., Univ. Twente, Enschede, 1996, pp. 139-148.

Els96. Elsner, A. and Klein, A. Erkennung des prosodischen Fokus und die Anwendung im dialogaktbasierten Transfer. Tech. Rept. Memo 107, Verbmobil, 1996.

Fér93. Féry, C. *German Intonational Patterns*, Niemeyer, Tübingen, Linguistische Arbeiten(1993).

Faw89. Fawcett, R.P. and Taylor, M.M. A Generative Grammar for Local Discourse Structure. In *The Structure of Multimodal Dialogue*, North-Holland, Amsterdam, 1989.

Faw92. Fawcett, R.P. and Davies, B. Monologue as a Turn in Interactive Discourse: Towards an Integration of Exchange Structure and Rhetorical Structure Theory. In *6th International Workshop on Natural Language Generation, 1992.*, Dale, Rösner, Trento, Italy, 1992.

Fen94. Fensel, D. and Harmelen, F. A Comparison of Languages which Operationalize and Formalise KADS Models of Expertise. *The Knowledge Engineering Review* , 8 (2) (1994), 49.

Fis85. Fisher, D.H. and Langley, P. Approaches to Conceptual Clustering. In *Proc. of the Ninth International Joint Conference on Artificial Intellingence,* 1985, pp. 691-697.

Fis87. Fisher, D.H. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* , 2 (1987), 139-172.

Fis94. Fischer, M., Maier, E., and Stein, A. Generating Cooperative System Responses in Information Retrieval Dialogues. In *7th Internatinal Workshop on Natural Generation,1994,* M.A. Walker, O.R., Kennebunkport, 1994, pp. 207-216.

Gen89. Gennari, J.H., Langley, P., and Fisher, D.H. Models of Incremental Concept Formation. *Artificial Intelligence* , 40 (1989), 11-61.

Gri86. Grishman, R. and Kittredge, R. *Analysing language in restricted domains*, Lawrence Erlbaum (1986).

Gro86. Grosz, B.J. and Sidner, C.L. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics* , 12 (1986), 175-204.

Gro87. Grossberg, S. Competitive Learning: From Adaptive Activation to Adaptive Resonance. *Cognitive Science* , 17 (1987), 23-63.

Hüb96. Hübener, K., Jost, U., and Heine, H. Speech recognition for spontaneously spoken German dialogs. In *4th Int. Conference on Spoken Language Processing*, Philadelphia, 1996.

Hir92. Hirschberg, J. and Grosz, B.J. Intonational Features of Local and Global Discourse Structure. In *Proc. Speech and Natural Language Workshop, Harriman, N.Y.*, D.R.A., Morgan Kaufmann Publishers, San Mateo, CA, 1992, pp. 441-446.

Hir93. Hirschberg, J. and Litman, D.J. Empirical Studies on the Disambiguation of Cue Phrases. *Computational Linguistics* , 19/3 (1993), 501-530.

Jön91. Jönsson, A. A Dialogue Manager Using Initiative-Response Units and Distributed Control. In *Proc. European Chapter of the ACL*, 1991, pp. 233-238.

Jön93. Jönsson, A. A Method for Development of Dialogue Managers for Natural Languages. In *Proc. of Annual Meeting of the American Association of Artificial Intelligence*, 1993, pp. 190-195.

Jek95. Jekat, S., Klein, A., Maier, E., Maleck, I., Mast, M., and Quantz, J.J. Dialogue Acts in VERBMOBIL. Tech. Rept. Report 65, Verbmobil, ??, 1995.

Kno91. Knott, A. New Strategies and Constraints in RST-based Text Planning. *??* (1991), 81.

Koh84. Kohonen, T. *Self-Organization and Associative Memory*, Springer Verlag, Berlin , Springer Series in Information Sciences(1984).

Kom94. Kompe, R., Nöth, E., Kießling, A., Kuhn, T., Mast, M., Niemann, H., and Ott, K. Prosody takes over: Towards a Prosodically Guided Dialog System. Tech. Rept. Report 48, Verbmobil, ??, 1994.

Lam91. Lambert, L. A Tripartite Plan-Based Model of Dialogue. In *Proc. 29th Annual Meeting of the ACL*, 1991, pp. 47-54.

Leh86. Lehrberger, J. Sublanguage Analysis. In *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, Grishman, R.; Kittredge, R., Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 19-38.

Leh94. Lehnert, W. and Soderland, S. Corpus-Driven Knowledge Acquisition for Discourse Analysis. In *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle, 1994, pp. 827-832.

Lit85. Litman, D.J. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*, Ph.D. dissertation, ??, 1985.

Lit94. Litman, D.J. Classifying Cue Phrases in Text and Speech Using Machine Learning. In *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle, 1994, pp. 806-813.

Lup96. LuperFoy, S. Tutoring versus Training: A Spoken Language Dialogue Manager for Instructional Systems. In *Proc. Twente Workshop on Language Technology, Dialogue Management in NL-Systems*, LuperFoy, Nijholt, Z., Univ. Twente, Enschede, 1996, pp. 45-49.

Möl90. Möller, J.U. Fachdialogwissen als Mittler zwischen Fachwissen und Dialogwissen. In *GWAI-90 14th German Workshop on Artificial Intelligence, Eringerfeld*, Marburger, Heinzman Workshop on Artificial Intelligence, E., Springer-Verlagzman Workshop on Artificial Intelligence, Eringerfeld, Berliner-Verlagzman Workshop on Artificial Intelligence, Eringerfeld, 1990, pp. 58-64.

Möl92a. Möller, J.U. Leitfaden zur Modellierung von Fachdialogen zwischen Nutzern und System. Tech. Rept. Memo 211/92, Berichte des Fachbereichs Informatik, Universität Hamburg, 1992.

Möl92b. Möller, J.U. Fachspezifische Implikatoren für die Planung und Generierung von Inhalt und Form in Dialogen. In *16th German Workshop on Artificial Intelligence,* 1992.

M94. Möller, J.U. Domain-Related Focus-Shifting Constraints in Dialogues with Knowledge Based Systems. In *Trends in Natural Language Generation: an Artificial Intelligence Perspective,* Adorni, Giovanni; Zock, M., Springer Verlag, LNAI, 1994.

Mai94. Maier, E. Dialogmodellierung in VERBMOBIL - Festlegung der Sprechhandlungen für den Demonstrator. Tech. Rept. Memo 31, Verbmobil, 1994.

Man88. Mann, W.C. and Thompson, S.A. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *Text* , 8,3 (1988), 243-281.

Mar94. Martin, J. and Billman, D.O. Acquiring and Combining Overlapping Concepts. *Machine Learning* , 16 (1994), 121-155.

Mas95. Mast, M. Schlüsselwörter zur Detektion von Diskontinuitäten und Sprechhandlungen. Tech. Rept. Memo 57, Verbmobil, 1995.

Moo92. Moore, J.D. and Pollack, M. A Problem for RST: The Need for Multi-Level Discourse Analysis. *Computational Linguistics* , 18/4 (1992), 537-544.

Mus93. Music, B. and Povlsen, C. The NLP Module of a Spoken Language Dialogue System for Danish Flight Reservations. In *Eurospeech 93,* E.S., E.S., Berlin, Genf, 1993, pp. 1859-1862.

Nöt91. Nöth, E., Batliner, A., Kuhn, T., and Stallwitz, G. Intensity as a predictor of focal accent. In Aix-en-Provence, 1991.

Nov96. Novick, D.G. and Sutton, S. Building on experience: Managing spoken interaction through library subdialogues. In *Proc. Twente Workshop on Language Technology, Dialogue Management in NL-Systems,* LuperFoy, Nijholt, Z., Univ. Twente, Enschede, 1996, pp. 51-60.

Pie80. Pierrehumbert, J. *The Phonology and Phonetics of English Intonation*, Ph.D. dissertation, MIT, Cambridge, MA., 1980.

Ram91a. Ramshaw, L.A. A Three-Level Model for Plan Exploration. In *Proc. 29th Annual Meeting of the ACL,* 1991, pp. 39-46.

Ram91b. Rambow, O., Kittredge, R., and Korelsky, T. On the Need for Domain Communication Knowledge. *Computational Intelligence* , 7(4) Dec. 91 (1991).

Rei95. Reithinger, N. and Maier, E. Utilizing Statistical Dialogue Act Processing in VERBMOBIL. Tech. Rept. Report 80, Verbmobil, 1995.

Rey94. Reyelt, M. Untersuchung zur Konsistenz prosodischer Etikettierungen. Tech. Rept. TD 03-94, Verbmobil, 1994.

Rit89. Ritter, H. and Kohonen, T. Self-Organizing Semantic Maps. *Biological Cybernetics* , 61 (1989), 241-254.

Rum80. Rumelhart, D.E. and Zipser, D. Feature Discovery by Competitive Learning. In *Parallel Distributed Processing: Explorations in the Mircostructure of Cognition,* Rumelhart, D. E.; McClelland, J.L., MIT Press, Cambridge, 1980, pp. 151-193.

Sch92. Schröder, M. Knowledge Based Analysis of Radiology Reports Using Conceptual Graphs. In *Conceptual Structures: Theory and Implementation - Proc .7th Workshop, Las Cruces, New Mexico*, Pfeiffer, H. D.; Nagle, T.E., Springer Verlag, Lecture Notes in Artificial Intelligence, Berlin, 1992, pp. 293-302.

Sch93. Schröder, M. *Erwartungsgestützte Analyse medizinischer Befundungstexte - Ein wissensbasiertes Modell zur Sprachverarbeitung*, Ph.D. dissertation, infix, St. Augustin, 1993.

Sie94. Siegel, E.V. and McKeown, K.R. Emergent Linguistic Rules from Inducing Decision Trees: Disambiguating Discourse Clue Words. In *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle, 1994, pp. 820-826.

Sil92. Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., and Hirschberg, J. Tobi: A standard for labeling english prosody. In *International Conference on Spoken Language Processing*, 1992.

Ste93. Stein, A. and Thiel, U. A Conversational Model of Multimodal Interaction in Information Systems. In *Proc. of Annual Meeting of the American Association of Artificial Intelligence*, 1993, pp. 283-288.

Str95. Strom, V. Detection of accents, phrase boundaries, and sentence modality in German. In *Proc. Eurospeech*, Vol3, 1995, pp. 2039-2041.

Sun94. Sundin, U. Assignment and Scheduling. In *CommonKADS Library for Expertise Modelling - Reusable Problem Solving Components*, Breuker, J.; Van de Velde, W., IOS Press, Amsterdam, 1994, pp. 231-264.

Uhm91. Uhmann, S. *Fokusphonologie: Eine Analyse deutscher Intonationskonturen im Rahmen der nichtlinearen Phonologie*, Niemeyer, Tübingen , Linguistische Arbeiten(1991).

Wac86. Wachtel, T. Pragmatic sensitivity in NL interfaces and the structure of conversation. In *11th Int. Conf. on Computational Linguistics, Bonn (COLING86)*, 1986, pp. 35-41.

War96. Ward, W. Dialog Management in the CMU Spoken Language Systems Toolkit. In *Proc. Twente Workshop on Language Technology, Dialogue Management in NL-Systems*, LuperFoy, Nijholt, Z., Univ. Twente, Enschede, 1996, pp. 133-138.

Web95. Weber, V. and Wermter, S. Towards Learning Semantics of Spontaneous Dialog Utterances in a Hybrid Framework. In *??*, 1995.

Woo84. Woolf, B. and McDonald, D.D. Context-dependent Transitions in Tutoring Discourse. In *Proc. Annual Meeting of the American Association of AI*, 1984, pp. 355-361.

You94. Young, M. and Moore, J.D. Does Discourse Planning Require a Special-Purpose Planner?. In *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle, 1994.

APPENDIX A: OVERLAP MATRIX TABLE

APPENDIX B: DIALOGUE ACT HIERARCHY FROM THE VM PROJECT

```
dialogue act
     request_suggest
          request_suggest_date
          request_suggest_location
          request_suggest_duration
     accept
          accept_date
          accept_duration
          accept_location
     give_reason
     clarify
          clarify_query
          clarify_answer
          clarify_state
     confirm
     deliberate
          deliberate_explicit
          deliberate_implicit
     garbage
     suggest
          suggest_exclude
               suggest_exclude_date
               suggest_exclude_duration
               suggest_exclude_location
          suggest_support
               suggest_support_date
               suggest_support_duration
               sugest_support_location
     feedback
     digress
          digress_scenarion
          refer_to_setting
     introduce
          introduce_name
          introduce_react
          introduce_position
     init
          init_date
          init_duration
          init_location
     motivate
          motivate_appointment
     reject
          reject_date
          reject_location
          reject_duration
     thank
          thank_init
          thank_react
     request_comment
          request_comment_date
          request_comment_duration
          request_comment_location
```