

Tilburg University

Extracting Information from Spoken User Input

Lendvai, P.K.

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

[Link to publication in Tilburg University Research Portal](#)

Citation for published version (APA):
Lendvai, P. K. (2004). *Extracting Information from Spoken User Input: A Machine Learning Approach*. [n.n.].

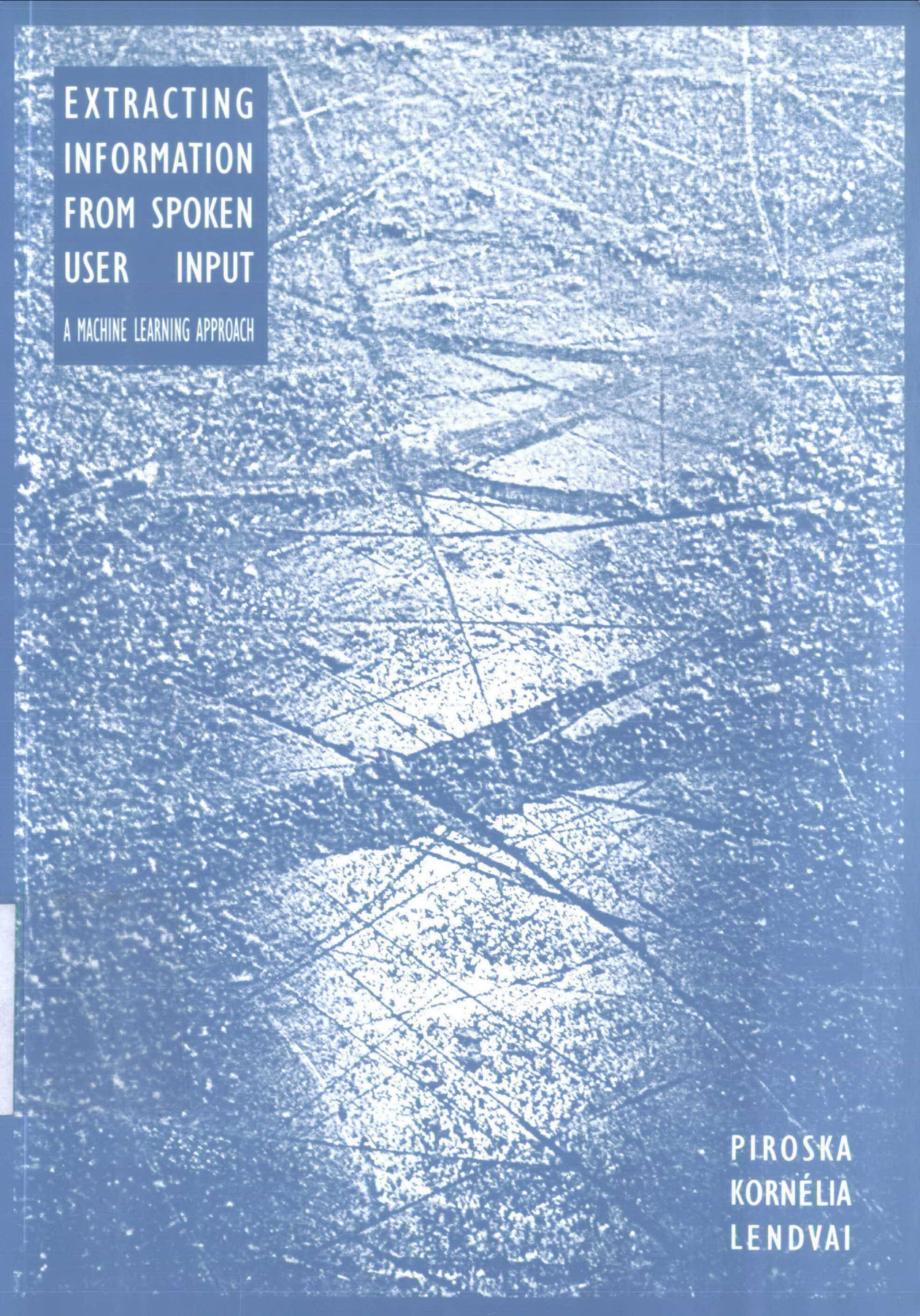
General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



EXTRACTING INFORMATION FROM SPOKEN USER INPUT

A MACHINE LEARNING APPROACH

PIROSKA
KORNÉLIA
LENDVAI



PIROSKA KORNÉLIA LENDVAI

EXTRACTING INFORMATION FROM SPOKEN USER INPUT

A MACHINE LEARNING APPROACH

The project of this thesis was funded by SOBU (Samenwerkingsorgaan Brabantse Universiteiten;
Organisation for cooperation between universities in the Brabant region)

© 2004 Piroska Kornélia Lendvai

ISBN 90-9018874-6

Printed in Enschede

Typeset in L^AT_EX

Cover: Yubileyny, St. Petersburg 2004



Extracting Information from Spoken User Input

A Machine Learning Approach

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
prof. dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen ten overstaan van
een door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op maandag 20 december 2004 om 10.15 uur

door

Piroska Kornélia Lendvai

geboren op 24 december 1972
te Bonyhád, Hongarije

Promotores: Prof. dr. W.P.M. Daelemans
Prof. dr. H.C. Bunt
Copromotores: Dr. A.P.J. van den Bosch
Dr. E.J. Krahmer

...

Presenter: You have a new theory about the brontosaurus...

Anne Elk: Can I just say here Chris for one moment that I have a new theory about the brontosaurus?

Presenter: Uh... exactly. What is it?

Anne Elk: Where?

Presenter: No — no, what is your theory?

Anne Elk: What is my theory?

Presenter: Yes!

Anne Elk: What is my theory that it is? Yes — well, you may well ask what is my theory.

Presenter: I *am* asking.

Anne Elk: And well you may. Yes, my word, you may well ask what it is, this theory of mine. Well, this theory, that I have, that is to say, which is mine — is mine.

Presenter: I know it's yours! What is it?

Anne Elk: Where? Oh, what is my theory? Ah! My theory, that I have, follows the lines that I am about to relate...

Presenter: Oh, God...

Anne Elk: The theory, by Anne Elk...

Presenter: Right...

Anne Elk: [clears throat] This theory, which belongs to me, is as follows — [clears throat] This is how it goes — [more throat clearing] The next thing that I am about to say is my theory — [clears throat] Ready? The theory, by Anne Elk, brackets, miss, brackets. My theory is along the following lines...

Presenter: God!

Anne Elk: All brontosaurususes are thin at one end, much, *much* thicker in the middle, and then thin again at the far end. That is the theory that I have, and which is mine, and what it is, too.

Presenter: That's it, is it?

Anne Elk: Right, Chris.

Presenter: Well, Anne, this theory of yours seems to have hit the nail right on the head...

Anne Elk: And it's mine.

...

(From: *Monty Python's Brontosaurus Theory Sketch*)

Acknowledgements

I would like to express my thanks towards those who helped me in the past three and a half years while I was engaged in the process of bringing this thesis into existence.

I feel honoured that Walter Daelemans was willing to be my PhD advisor (promotor). Next to the substantive comments made on my study, the sophisticated knowledge base developed by his pioneering research in language technology influenced me a lot. I am thankful to Harry Bunt for encouraging me to apply for a ‘Learning to Communicate’ project at Tilburg University. His critical remarks, especially on prefinal versions of the manuscript, enabled me to considerably strengthen theoretical aspects of my work.

I am most indebted to my daily supervisors, Antal van den Bosch and Emiel Krahmer, for their dedicated support and guidance through all the stages of my project, exhibiting striking patience, excellent mentorship, and being a cheerful company day by day. They demonstrated an unbelievable amount of creative thought and willingness to consult me on various aspects of research, intellectual interests, and my stay in the Netherlands. I am especially grateful for the immediate, essential comments, tireless advocacy on rephrasing, and ideas received from Emiel and Antal during writing the thesis text. This work is the product of our close cooperation, which I enjoyed a lot.

I am glad for having the possibility to conduct research with Marc Swerts (Tilburg and Antwerp Universities), Laura Mărușter (Eindhoven and Tilburg Universities), and Sander Canisius (Tilburg University); this dissertation shows the impact of our joint work. Jacques Terken has been kind to act as my SOBU supervisor at the Eindhoven University of Technology and to provide useful comments on the manuscript. Acoustic data processing was done courtesy of Leo Vogten (Eindhoven University). Special thanks to Antal for the WPS program and a number of data processing scripts, to Jan Kooistra for software support, to Emiel for standardising the Dutch summary, and to the authors of the TIMBL manual.

It has been a pleasure to work in the inspiring environment of the Department of Computational Linguistics and the Induction of Linguistic Knowledge research group. Thanks to all colleagues, in particular to Ielka van der Sluis, who made the past years comfortable and fun: Anne Adriaensen, Bertjan Busser, Elias Thijssen, Els van Loon, Erik Tjong Kim Sang, Erwin Marsi, Hans Paijmans, Iris Hendrickx, Jakub Zavrel, Jeroen Geertzen, Ko van der Sloot, Martin Reynaert, Menno van Zaanen, Olga van Herwijnen, Paul Vogt, Reinhard Muskens, Roser Morante, Sabine Buchholz, Yann Girard, and the friendly people of the Faculty of Arts.

I am pleased that Ielka and Jan undertook the responsible task of being paranymphs,

providing trusted help and company in moments quite different from watching odd films, going to retro-concerts, performing deep aesthetic analysis of travel photos, and the like.

My deepest thanks to Yevgen Rudenko for standing by me in all moments, as well as to our families, for all precious emotions and cultural heritage.

The incomplete list of friends who have been keeping up my spirit during this period includes Adrien Haraszti, Anne-Marie van den Bosch, Arthur and Barbara Zhuravlov, Bea Nemes, Bernadett Kárász, Boris Yakshov and Galina Pronicheva, the Bagry family, Edit Gaál, Eszter Zákányi, Gergely Thuróczi, Miklós Urbán, Levente Bejdek, Nomi Vereckei, Olga Vybornova, Paul Meijer, Péter Simon, Tamás Biró, the Van der Sluis family, Zsófi Fekete, Zsolt Müller, Zsolt Varga, and Zseby Zoltán Wojnischek.

Especially on this day I would like to convey my respect and emphasised affection towards the art created by Evgeni Plushenko, as well as to the numerous (for some reason USSR-related) actors, directors, writers, musicians, and other performing artists, who impressed me every single day. Thank you for providing essential motivation for going on.

Tilburg, 3 November 2004.

Contents

1	Introduction	1
1.1	The complexity of interpreting user input in spoken dialogue systems	1
1.2	Machine learning for extracting information from spoken user input	2
1.3	Research objectives	3
1.3.1	A robust approach	6
1.3.2	Detecting task-related acts	7
1.3.3	Detecting information units	8
1.3.4	Detecting forward-pointing problems	9
1.3.5	Detecting backward-pointing problems	10
1.4	Overview	12
2	Computational Interpretation of Spoken User Input	13
2.1	Natural language understanding in spoken dialogue systems	13
2.2	Analysis levels in interpreting spoken user input	16
2.2.1	Task-related acts	17
2.2.2	Information units	18
2.2.3	Forward-pointing problems	20
2.2.4	Backward-pointing problems	21
2.3	Potential information sources for interpretation	22
2.3.1	Cues in analysing task-related acts	23
2.3.2	Cues in analysing information units	23
2.3.3	Cues in analysing forward-pointing problems	24
2.3.4	Cues in analysing backward-pointing problems	25
2.4	Summary	26
3	Machine Learning as a Research Environment	27
3.1	Algorithm choice	28
3.1.1	Memory-based learning	29
3.1.2	Rule induction	33
3.2	Experimental methodology	36
3.2.1	Algorithm parameter optimisation	37
3.3	Summary	39

Chapter 1

Introduction

1.1 The complexity of interpreting user input in spoken dialogue systems

Spoken dialogue systems (SDSs) are developed to assist people at controlling devices and at accessing various computer-based services. When human users interact with a SDS, a specific type of communication takes place that is referred to as task-oriented dialogue. In task-oriented dialogues the dialogue partners want to reach some common goal, one that represents the purpose of the utilised device or service. Our study focuses on SDSs that are information-providing systems. In such SDSs the common goal is to transfer information from the system to the user. SDSs of this kind can also be seen as speech interfaces to databases, enabled by a successful interaction to perform a database search: the database is consulted and information is retrieved by the system when enough query constraints are obtained from the input supplied by the user. The query constraints are pieces of information that are inferred from what the user says during the dialogue. In other words, interaction with the SDS proceeds via a series of dialogue exchanges, i.e., pairs of system and user turns, which lead to a computational state where the database query can be performed. When the query result is delivered to the user, the goal of the interaction is fulfilled.

A crucial subprocess of the interaction is thus that the dialogue system infers the content of user turns. This takes considerable effort; at least three major factors contribute to the complexity of such automatic interpretation. One factor is that the spoken material may contain noise. Apart from environmental and channel-related auditory noise, linguistic noise may also be present in spoken input: ungrammatical linguistic constructions are frequently uttered by people, and the presence of so-called disfluent elements such as stuttering, repetitions, and filled pauses, which do not belong to the intended informational content of the utterance, is not uncommon. In addition, the results of automatic speech recognition (ASR) implemented in a SDS are often incorrect, especially when the ASR engine has to operate on large domains. Errors in SDS-internal measurements can also occur, and may lead to noise in the material from which in-

formation needs to be extracted by the SDS. Additionally, noise has been found to be difficult to automatically distinguish from linguistic subregularities and exceptions (cf. [Daelemans et al. 1999, Rotaru and Litman 2003]).

The second factor accounting for complexity in interpreting user input is that in a task-oriented dialogue a user turn is typically some concise utterance that amalgamates manifold communicative aspects. [Traum 2003] identifies three inherent levels of questions and answers in human-machine communication: (i) the performance level of dialogue acts, (ii) the semantic level of basic values, and (iii) the interactional level of the conversation. For example, a typical user reply to an information-demanding system prompt (i.e., the machine's utterance, e.g. 'How may I help you?') can be considered to simultaneously perform the acts of information providing, supplying the particular pieces of information that were requested, and giving feedback on how the interaction is progressing (e.g., 'I would like to know about recreational activities in Tilburg.'). [Krahmer et al. 2001b] find that a positive feedback (i.e., signalling that the communication proceeds without problems) is often represented by a zero element in the utterance, that is, the user will usually not say explicitly that the interaction progresses well.

The third factor explaining why it is not trivial to infer the content of a user turn is that language technology employed to automatically extract this content is error-prone: substantial research has been carried out on the complex task of user understanding, but present applications still seem to require innovative enhancements to allow for successful human-machine communication on a more general scale. This calls for devising robust techniques that work with extensive coverage of spoken language phenomena and sufficient precision at the same time (cf. [Maynard et al. 2002, He and Young 2004]).

1.2 Machine learning for extracting information from spoken user input

In recent years there has been an increased interest in using statistical and machine learning approaches for the processing of user utterances in spoken dialogue systems. Dialogue act classification is an example for which this approach has been relatively successful. The goal of this task is to determine what the underlying intention of an utterance is (e.g., suggest, request, reject, etc.). Various techniques have been used for this purpose, including data-driven language models [Reithinger and Maier 1995], maximum entropy estimations [Choi et al. 1999], mixed stochastic techniques [Stolcke et al. 2000], transformation-based learning [Samuel et al. 1998b], and others. For processing and understanding the units of information that represent the content of spoken user utterances, statistical techniques have also proven their usefulness, either in combination with rule-based grammars (e.g. [Cettolo et al. 1996, Van Noord et al. 1999, Wahlster 2000, Cattoni et al. 2001]) or without them (for example [Allen et al. 1996, Nakano et al. 1999]).

Another task for which machine learning approaches have been applied is automatic problem detection. Given the frequent occurrences of communication problems between users and systems due to misrecognitions, erroneous linguistic processing, incorrect assumptions, and the like, it is important to detect problems in the interaction as soon as possible, or even try to anticipate them (cf. [Hirschberg et al. 2000, Litman et al. 2000,

Walker et al. 2000a, Hirschberg et al. 2004]). Various researchers have also shown that users signal communication problems when they become aware of them, and that it is possible to pinpoint utterances that reveal that the user acquired knowledge (perhaps not even fully consciously) about a communication problem (cf. [Hirschberg et al. 2001, Van den Bosch et al. 2001]). Such turns are sometimes referred to as awareness sites, a term which we will also use in our study.

Interpreting the acts performed and the information units supplied by the user, predicting, as well as identifying communication problems are all highly relevant tasks in processing user input in SDSs. Still, none of the studies in the literature addresses these issues in combination. Such a combined approach would establish a complex interpretation module for SDSs, extracting information about semantic aspects (such as the content of the user's utterance) and pragmatic aspects (the performed act, source of communication problems, feedback about the status of the dialogue) of the user input.

1.3 Research objectives

In this study we propose an architecture for a module that performs shallow analysis of user input in a SDS and provides a complex interpretation of user turns. We refer to the interpretation process as 'shallow' since no deep linguistic analysis is performed on the user input in order to infer the interpretation, and the material utilised by the module is obtained by simple means from the speech recogniser and the dialogue manager of the SDS. The output produced by the module is a four-level representation of the user turn, consisting of the following components:

- the performed basic task-related act(s),
- the information unit type(s) for which information was provided, in our study corresponding to the slots of the query to be completed,
- whether the turn is the source of communication problems,
- whether the turn exhibits user awareness of communication problems.

Figure 1.1 shows the interpretation module in a schematic SDS architecture. After the user input is supplied, it is processed by the ASR. The output of the ASR is fed into the language interpretation module, of which shallow interpretation forms a submodule. The shallow interpretation module receives input from the dialogue manager module as well. The dialogue manager (DM) module is typically the central coordinating unit of a SDS, responsible for maintaining the interaction by incorporating the content of the user input, and designing an adequate response strategy to that user input (for details see for example [Flycht-Eriksson 1999, Traum and Larsson 2003, Popescu-Belis et al. 2003]).

The next step in the process described in Figure 1.1 is that the shallow interpretation module extracts the above pieces of information based on the material received from the ASR and the DM, whereby a four-level interpretation of the user turn is obtained. If performed accurately, arguably, such a complex interpretation is able to improve language processing in a dialogue system in many ways. Apart from facilitating full understanding

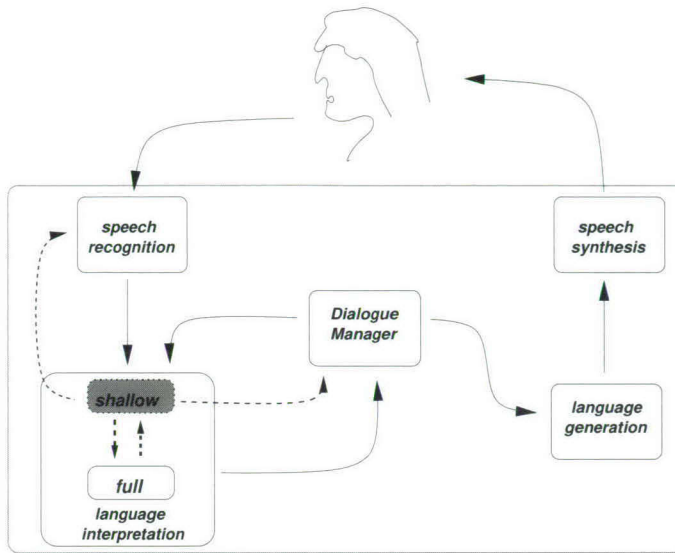


Figure 1.1: The shallow interpretation module (indicated by the dark box, situated in a full language interpretation module) in a possible SDS architecture. The dashed arrows symbolise potential connections between the shallow interpretation module and other modules of the SDS.

of the input, the resulting interpretation can be fed back to the speech recognition and the dialogue manager of the SDS that can utilise this information in a number of ways. For example, knowledge about the information unit types supplied in the user turn may enable the speech recogniser to be more confident about some hypothetical analysis of the utterance (cf. [Ringger and Allen 1997, Stolcke et al. 1998b, Zechner and Waibel 1998]). Likewise, from the obtained interpretation the DM may receive an indication that the user is signalling a problem, or that the user input is likely to be erroneously processed. This would enable the DM to adapt to the given situation, for example by changing the recognition engine, or by switching to a different error recovery or confirmation strategy (cf. e.g. [Hirschberg et al. 2004], and the references therein).

Arguably, by broadening the module we could additionally aim at extracting the actual values the user provides in the turn in case slot-filling activity is detected. However, it is not among the goals of our study to cover this issue.

The present work aims to be an interdisciplinary study: we integrate the components of the proposed shallow interpretation module in a machine learning framework. The learning task in this framework involves simultaneous task-related act and information unit type classification, as well as bidirectional problem detection. Corresponding to the four-level interpretation, the learning tasks in the module are the following:

- identify basic task-related act(s),
- identify the information unit type(s), i.e., query slot(s), for which information is provided (if any),
- identify forward-pointing problems, i.e., whether the turn is a source of miscommunication,
- identify backward-pointing problems, i.e., whether the turn exhibits user awareness of miscommunication.

Arguably, generating such a combined pragmatic-semantic interpretation is a difficult task since there are many ways in which an input may contain these different components. Natural language phenomena are often claimed to be ambiguous, since they yield various ways in which the spoken input may be interpreted. In addition, some of the components will be difficult to identify, e.g., whether a user turn indicates that the user is accepting a system error rather than that the user is providing positive feedback, or whether the user turn is likely to be erroneously processed or not.

In particular, our goal is to investigate the following research issues in our study:

- (i) to what extent certain machine learning techniques can be used for shallow interpretation of user turns in spoken dialogue systems,
- (ii) whether the complex learning task of four-level interpretation can be optimised by decomposing it to subtasks, and
- (iii) whether filtering noise from spoken input on the basis of higher-level linguistic information leads to improved learning performance on the shallow interpretation task.

Corresponding to (i), we train two supervised machine learning algorithms to extract information in terms of the four-level interpretation from user turns. This can be seen as a disambiguation task applied to spoken language material: the learning algorithms need to assign one complex interpretation to each user turn. [Daelemans et al. 1997] claim that complex tasks in natural language processing may be decomposed as sequential or parallel subtasks. Therefore, corresponding to (ii), we also test whether decomposing the complex four-level interpretation task into subtasks is more optimal for the extraction of pragmatic-semantic information from user input. Finally, corresponding to (iii), we devise techniques that attempt to block noise (such as syntactically or lexically incorrect or superfluous words that may have a negative effect on the interpretation task) from the algorithms. We use the method of automatic filtering to remove from our data (a) disfluent words, (b) syntactically less dominant words, and (c) words that may carry less informational value in the given human-machine interaction. We observe whether filtering the user input by these means yields improvement over using unfiltered data in the shallow interpretation task.

The goal of performing all learning experiments by two machine learning algorithms is to introduce a broader technical scope to our investigation: the two algorithms are representatives of different branches of supervised learning techniques, namely of memory-based learning and of rule induction. We train the algorithms on a large set of labelled

examples derived from the OVIS corpus of spoken human-machine dialogues with a Dutch train travel information system [Boves et al. 1995]. Information used by the algorithms comes from different sources, and is obtained by means that are affordable in most dialogue systems. We train the memory-based learner and the rule induction learner under identical conditions, and report on the experimental results of testing their performance on the shallow interpretation task.

1.3.1 A robust approach

The proposed shallow interpretation module aims to be robust in three respects, namely:

- to cope with noise in spoken input and in the shallow representation of such input,
- to account for multi-layeredness in the input content, and
- to deploy adequate machine learning techniques that form the core of the module.

To design a robust technical approach, we deal with noisiness on several levels. We attempt to design learning experiments in a way that tolerates approximative, erroneous, and hypothetical measurements in the data representing the spoken input, since the data comes from possibly imperfect measurements and hypotheses of the SDS itself (e.g., the ASR module). [He and Young 2004] claim that a spoken language understanding system should be able “to correctly interpret the meaning of an utterance even when faced with recognition errors”. Additionally, the filtering techniques indicated above are another attempt to devise mechanisms that compensate for noise both in the spoken input (i.e., the words uttered) and its representation in the SDS (e.g., the ASR hypotheses).

At the same time, we also try to automatically learn whether certain types of user input can be identified as problem sources that themselves introduce noise into the interaction with a SDS. Moreover, problem detection is attempted without carrying out a fine-grained typology of the occurring problems. Rather, two main groups of phenomena are defined and learnt: forward-pointing problems (i.e., problem source), and backward-pointing problems (i.e., feedback on the communicative situation).

In order to account for multi-layeredness in the input content, we extract information related to the pragmatic and semantic levels of the user input: on the pragmatic level task-related acts, problem source, and problem awareness are detected, on the semantic level the supplied information unit types are identified (if any). We hypothesise that identifying a few simple categories on the pragmatic and syntactic level yields robustness: for example, we identify that a user is supplying information in the given turn, as well as the query slot(s) to which this information corresponds, but it is not determined how the input globally influences the interaction, neither the functions the user intends to perform by such input (i.e., to correct something, to assert, or to agree, etc.), nor the way the content of the current input relates to the content of the previous input (i.e., whether the input contains repeated information, etc.), and so on. Rather, the user utterances are projected into basic supercategories of actions in the task domain (sometimes referred to as domain actions, cf. [Cattoni et al. 2001]), by which we aim to ensure applicability and transferability of the approach.

Shallow interpretation is conceptualised as a classification task, and our third goal in devising a robust approach is to design adequate machine learning techniques for optimal performance on this task. The techniques aim at attaining high classifier performance at a relatively low cost: the machine learners utilise information that is easily obtainable from the SDS, and that is represented in the experiments in a shallow way. No higher-level linguistic information, which is often computationally expensive to obtain, is used in the learning experiments. Even the filtering approaches, which attempt to implicitly incorporate higher-level linguistic information in the SI task, primarily draw on shallow, generally applicable machine-learning-based approaches.

The design of the shallow interpretation module is hypothesised to result in robust performance, whereby our goal is to develop a general method for shallow interpretation of user input by establishing a straightforward approach, implying that its successful transportation to a new domain of task-oriented human-machine interaction would involve the adjustment of the set of interpretation classes, and re-training on dialogue data from that domain.

Below we explain the significance of the four components of the shallow interpretation module in more detail.

1.3.2 Detecting task-related acts

The linguistic term ‘dialogue act’ refers to both general and specific types of intentions of the speaker that are manifested in and conveyed by the utterance of the speaker. The speaker’s intention in an utterance is largely formed by and is dependent on the situation in which it takes place. Since dialogue acts reflect the relationship between utterances and context-dependent communicative functions, dialogue acts are pragmatic in nature.

The discipline of computational pragmatics is concerned, among others, with the automatic detection and processing of dialogue acts (see for example [Bunt and Black 2000]), either in order to discover the underlying mechanisms of natural language dialogue in general, or to utilise these in natural language processing applications (see for example [Bunt 1989]). It is not trivial to infer what kind of dialogue act is being performed in a given utterance, even in a dialogue that takes place in a more restricted, for example task-oriented way. As described earlier, this is partly related to the fact that the speaker’s intentions within a turn are typically manifold; and more than one communicative intention may be expressed by one speaker turn. For example, in interacting with a SDS that provides information about recreational activities, the imaginary but plausible user turn ‘I did not say biking, I said hiking’ can be seen to simultaneously convey rejection, correction, information providing, repetition, and so forth. [Bunt 2001] suggests that it is beneficial for the utilisation of dialogue acts in practical applications to “consider an utterance as multifunctional rather than as (functionally) ambiguous”, which we also pursue in the present work.

A wide-branching taxonomy of dialogue acts exists in the literature (cf. for example [Bunt 2001, Popescu-Belis et al. 2003]), opening up many choices on how fine-grained dialogue acts may be defined in an actual interaction model. If the goal is to examine subtle communicative processes, it is probably useful to define many fine-grained categories of dialogue acts. However, we hypothesise that for a shallow interpretation module it suffices

to define a limited set of simple actions that a user may execute in interacting with an information-providing SDS, to which we refer as task-related acts, and to perform robust pragmatic analysis of user input on the basis of such task-related acts.

Note that certain members of task-related acts may pertain to classical dialogue acts, whereas others may be of a different type. We emphasise that our study deliberately does not concern the full level of dialogue acts (i.e., the established notions of all-purpose, as well as specific categories describing user intentions), but solely the pragmatic level of task-related acts which are carried out by users interacting with a SDS. Nonetheless, as we show later in more detail, our set of task-related acts aims to represent general notions, scalable to other types of dialogue as well.

Even if we restrict the automatic detection of user acts to those of task-related acts, the difficulty of automatic identification of these acts remains. One factor adding to this difficulty is that a user may digress from schematic anticipations in his or her reply to a system prompt: for example, the expectation that an information-demanding prompt will be followed by an information-providing answer does not apply to all situations, especially when speech- and language processing of the previous input has not been perfect. People may in such cases react with a range of utterance types. Consider for example the interaction with a train travel information system given in Figure 1.2. The SDS in this interaction prompts the user for values of slots it needs to fill in order to retrieve a particular train connection from a database. (The dialogue is sampled from the OVIS corpus, which we introduce later. Utterances are translated from Dutch; the original transcriptions are shown in Figure 1 of the Appendix.)

In the first exchange of the interaction the system prompts for departure and arrival station names, but the user fills only the departure slot, which is an action not uncommon in human-machine interaction. The system incorrectly thinks the user answered both slots, and proceeds by prompting the user for the next slot it requires (i.e., travel time). The user becomes aware of the system error from the prompt in the second system turn (S2), because information understood from the first input (U1) is implicitly verified by the SDS there. The user immediately signals that there is a communication problem: this is done by notifying the system that it has made an error, and not providing information for the required slot of departure time. This input again leads to misrecognition (see S3) since the system expected date and time information, but instead it heard the word ‘error’ (and perhaps this word is not in its vocabulary). In turn U4 the user changes his strategy and supplies the information that has been incorrectly confirmed. Unfortunately the user hangs up the telephone after this turn, perhaps because he had no more patience to continue the interaction.

1.3.3 Detecting information units

While task-related acts are pragmatic in nature, the information units that are related to the content of a turn concern the semantic level of the user input. Traditionally, in task-oriented dialogue such information units are the factual values entered by the user, which exist independently of the general context of the dialogue. Alternatively, and in our study, the supercategories to which certain groups of these factual values refer to can be considered as information unit types. Such supercategories are in other words the query

Turn	Utterance
S1	From which station to which station do you want to travel?
U1	From Amsterdam.
S2	When do you want to travel from Almelo to Amsterdam Central Station?
U2	Error.
S3	I'm sorry I did not understand you. Could you repeat when you want to travel from Almelo to Amsterdam Central Station?
U3	Go back, it's incorrect.
S4	I'm sorry, again I did not understand you. Could you say when you want to travel from Almelo to Amsterdam Central Station?
U4	I want to go from Amsterdam to Emmen.

Figure 1.2: User reactions to system error in a train timetable SDS (OVIS, dialogue nr. 002/005).

slots that are filled in when a user provides factual values. Identifying which slots are being filled can in itself be of practical value in task-oriented dialogue, for example to ascertain that a value that may be supplied for more than one slots (e.g., for both the departure and the arrival station name) is assigned to the right slot.

Again, the difficulty in extracting such information from the user turn is manifold. In the first place, speech recognition is a main source of problems, since incorrect recognition can put the process of inferring treated slots or slot values on the wrong track. Additionally, the values entered by the user are often difficult to recognise due to limitations in typical ASR vocabularies, especially since these values can form an infinite set in some domains. For example, in a train travel SDS a large number of station names and time indications need to be recognised, whereas in the recreational activities domain the user may name some lesser known sports type or geographical area that is not in the vocabulary of the ASR. In these cases it is difficult to extract the actual values provided for the slots.

Moreover, as mentioned above, in case of communication problems users tend to become confused and either not fill the demanded slots (see the turns U2 and U3 in Figure 1.2), or fill other slots than the system prompted for (see turn U4 in Figure 1.2). Another frequent phenomenon is that the user is providing more, or less information than was solicited by the corresponding system prompt (see turn U1 in Figure 1.2).

1.3.4 Detecting forward-pointing problems

In studies dealing with human-machine interaction, assessment of SDS performance is often based on two measures: on word accuracy, i.e., the percentage of words correctly recognised by the SDS, and concept accuracy, i.e., the percentage of semantic concepts correctly recognised (cf. [Boros et al. 1996]). In our study it is the lack of full concept accuracy in processing the user's turn that is regarded as a communication problem (also called miscommunication). Below we motivate why and how we attempt robust detection of miscommunication between the human user and the SDS.

Problems that ‘point forward’ are ones that originate in the current turn of the dialogue, and will have consequences in the following turn. Typically, these are cases when an utterance is erroneously processed (due to e.g., speech recognition flaws and incorrect language understanding, an issue that we are going to cover later), or the prompt generated in reaction to it is improper; typically, it requires practical insight into a given SDS to decide whether the former or the latter is the problem source in a given case. The user turns U1, U2, and U3 in Figure 1.2 are examples of a forward-pointing communication problem, because they lead to extracting incorrect values from the user input (in the case of U1), or to extracting nothing from the user input (in the case of U2 and U3).

Identifying whether the current user utterance will cause problems is supposedly difficult, since it is not straightforward to understand what makes an input improper in the forward-pointing dimension. This component not only has to cover technical issues that pose problems to the given dialogue system itself (such as its inability to cope with hyper-articulated speech, dialects, out-of-vocabulary words, or noisy input), but also problems that are due to cognitive misunderstandings between the two parties, such as assumptions and presuppositions, as well as unforeseen circumstances, for example that a user gets distracted by something, and so on. Yet another difficulty of automatically detecting forward-pointing problems is that the machine learning algorithm has less information available for learning this task, since it cannot yet rely on the user’s feedback.

In sum, the task of identifying forward-pointing problems consists of spotting problems that originate in the current turn, resulting in conceptual inaccuracy in the system. Detecting forward-pointing problems is useful since it enables the dialogue manager to expect what types of user input are going to be well or badly processed. Obtaining such knowledge is important in order to correctly reject the recognition hypothesis of potentially badly received turns, and to be more confident about having understood other turns correctly [Hirschberg et al. 2004]. At the same time, identifying user input that could potentially put the interaction at risk would enable the dialogue manager to adapt its strategy to a more optimal one [Litman and Pan 1999, Walker et al. 2000a, Walker et al. 2000b]. For example, if a certain type of user’s turns are poorly recognised, the system could switch to a very explicit prompting strategy, or could re-prompt for the input and try to recognise it using a differently trained ASR [Hirschberg et al. 2004].

1.3.5 Detecting backward-pointing problems

Giving feedback is an essential mechanism of dialogue. To comply with the requirements of communication, the information exchanged by the dialogue partners needs to be grounded, i.e., established by acknowledgement from time to time (cf. [Traum 1994, Traum and Heeman 1997]). Grounding can be seen as the management of communication in order to reach mutual understanding. Providing feedback is one of the ways by which grounding operates, requiring that the partners provide feedback on how successful the information exchange was. Grounding can be seen as an action, the function of which is the management of the interaction.

Feedback is given by each conversational partner in a dialogue: in human-machine communication the machine too should return information to the user on how well the input is received. In SDS this is mostly realised via implicit verification prompts or

explicit verification prompts. Implicit verification prompts present to the user what was understood from the previous turn, and at the same time prompt for new information concerning unfilled slots. Turns S2, S3, and S4 in Figure 1.2 are implicit verifications of the (incorrect) departure station and the destination station values. When the user notices from these prompts that the system misunderstood him, making corrections is often difficult, since the SDS is asking for new information already. Users are generally puzzled in such cases, not knowing how to correct and supply information at the same time [Weegels 2000]. Note that [Krahmer et al. 2001b] find that signals concerning information grounding can either be positive ('go on') or negative ('go back'), where "negative cues are comparatively marked, as if the speaker wants to devote additional effort to make the other aware of the apparent communication problem ([Swerts et al. 1998])".

Just like humans may signal with a zero element that communication progresses as intended, SDSs may also simply proceed when they assume having understood everything correctly. The system turns S2, S3, and S4 in Figure 1.2 illustrate that, with respect to awareness in communication problems, SDSs can be in two states when processing user input: they either assume having obtained the correct processing of the user input (which assumption might or might not be correct; e.g., in S2 this is incorrect), and continue the dialogue in due order, or they assume that the user turn could not be correctly processed (which again might or might not be the case). In the latter case the system typically produces a clarification prompt, requesting the user to re-enter his input. For examples on how and why these system states can emerge, see [Streit 2003].

Typically, certain prompt types reveal that the system realises it has interpretation problems. Meta-prompts ('Try saying a short sentence'), apology ('I'm sorry I did not understand you'), repeated prompts, and prompts asking the user to repeat information all mark that the system is not confident enough in the processing results of the previous input. Obviously, the important part of problem detection is to point out cases when the system was incorrectly confident in some interpretation, which implies that it will also be detected when the system was correctly confident in some interpretation.

It is important to note that giving feedback is traditionally regarded as a dialogue act. However, we do not treat the full diversity of feedback phenomena in this study (for details see for example [Bunt 2001]). Rather, we focus on the — from the point of view of human-machine communication — important phenomenon of awareness in communication problems. We refer to the detection of this phenomenon as the detection of backward-pointing problems. In sum, the task of identifying backward-pointing problems consists of spotting turns in which the user became aware of the system's incorrect processing of the input. If aware sites are detected, they can provide an important cue for the system about the user noticing communication problems (of which the system might not yet be aware), so that the SDS can launch some error recovery strategy on time.

We hypothesise that it is important to distinguish problems with respect to the time line of their effect (i.e., forward- vs backward-pointing problems), because in this way a two-fold approach is designed to problem detection in SDS. As certain utterances are unproblematic in the current turn (i.e., in the forward-pointing dimension) but at the same time reflect awareness of problems that occurred in the previous turn (i.e., in the backward-pointing dimension), different problem categories can be assigned to the properties (i.e., the words, the intonation, the situational context, etc.) of a turn. By differentiating these

two tasks based on the direction of their effect we can reuse research material in a unified but dual-perspective way for error detection, enabling classification of subtle processes taking place within a user turn.

1.4 Overview

The structure of our study is the following. Chapter 2 discusses our four components in shallow interpretation by surveying previous work in the field of automatic processing of spoken input. We touch upon the issues of data annotation, as well as the information sources employed in machine-learning-based research. In Chapter 3 we introduce the discipline of machine learning and describe the two learning algorithms we work with. Our experimental methodology, as well as the general experimental set-up are explained.

Chapter 4 starts with introducing our research material, the OVIS corpus. We describe the corpus annotation and the information we employ in our machine learning experiments. Subsequently, the results of the learning experiments on the complex shallow interpretation task are presented. We provide an analysis of the obtained results at the end of the chapter.

In Chapter 5 we attempt to optimise learning performance on the shallow interpretation task. This is carried out by the method of information partitioning. A systematic search is conducted for the optimal class and feature group composition for each component of the shallow interpretation task (i.e., of the task-related acts, information units, forward-pointing problems, and backward-pointing problems). We provide qualitative and quantitative analysis of the experiments per component.

In Chapter 6 we conduct information filtering. We test machine learning-based, general filtering techniques on our data, aiming at eliminating material from the user input that may interfere with the shallow interpretation task. Three filtering techniques are applied to the task design optimised in Chapter 5. We compare the performance of the machine learning algorithms on the filtered and the unfiltered input. We present the conclusions of our research on shallow interpretation in Chapter 7.

Chapter 2

Computational Interpretation of Spoken User Input

The current chapter outlines some important aspects of computational processing of spoken user input. We discuss previous work related to shallow interpretation (SI), pointing out similarities and differences between work done in this area by other researchers, and our approach. The survey elaborates on the issue of annotating spoken dialogue corpora for learning tasks in SI. We examine what components, present in our four-level SI approach, are treated in other studies, and what attributes machine learners use in those works.

2.1 Natural language understanding in spoken dialogue systems

In order to infer the content of user input, often a language processing module is implemented in SDSs. Computational processing of natural language aims to model language so that computer programs can analyse language material on various levels. From the scientific point of view the emphasis in natural language processing (NLP) lies in creating a computational theory of language comprehension and generation. However, in practical applications this mainly comes down to providing solutions for the automatic processing of certain linguistic aspects of natural language utterances, by “methods that can work on raw text as it exists in the real world” [Manning and Schutze 1999].

NLP may draw on many different disciplines in discovering and modelling regularities of language, whether of a structural or a cognitive nature. [Jackson and Moulinier 2002] differentiate empirical NLP from symbolic in the sense that, in order to construct a model of language, empirical NLP “looks for patterns and associations, some of which may not correspond to purely syntactic or semantic relationships”. Indeed, our approach to SI can be seen as a direct mapping of a bulk of natural language material to linguistically cross-categorical concepts that incorporate four dimensions that are pragmatic-semantic

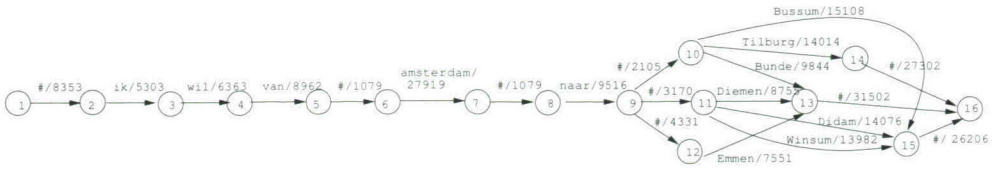


Figure 2.1: Word graph of the user input in turn U4 of Figure 1.2 ‘ik wil van Amsterdam naar Emmen’ (*I want to go from Amsterdam to Emmen*). Hash marks stand for pauses, the confidence score of each word hypothesis is given after the slash.

in nature. As stated in the previous chapter, our goal is to assign to user turns in a SDS a representation that incorporates task-related act(s), information unit(s), forward- and backward-pointing problems. Our approach is in line with [Eisele and Ziegler-Eisele 2002] who claim that “some [language] technologies cannot be assigned to one specific [linguistic] level, because they serve a more generic purpose”, and pinpoint the treatment of noise in the input as being such a purpose.

Natural language understanding (NLU) focuses on the comprehension part of NLP. Understanding human speech technically consists of two parts, speech processing and language processing, both making use of some kind of language modelling, traditionally in the form of a lexicon and a grammar. Statistical methods are widely used in NLU as these have proved to be simple and successful, drawing on n -gram distributions of linguistic units (phonemes, words, etc.) in the user input.

SPEECH PROCESSING In the first part of the NLU process, methods of speech technology are applied to analyse various acoustic-phonetic parameters of the speech signal in the form of amplitude, frequency, energy and possibly other measures. Based on these measures and a language model employed in the ASR, the speech recogniser produces a list of hypothetical sequences of words corresponding to the speech signal. The ASR’s hypotheses of a user utterance in this way consist of an n -best list of word strings. This output is often combined in a lattice, which is a directed acyclic graph in which the nodes are time points and the arcs are word hypotheses. Figure 2.1 shows this word graph for the input of user turn U3 in Figure 1.2. It can be observed that the first part of this turn (‘I want to go from Amsterdam to’) is processed by the ASR without any branching in the graph (i.e., only one word string is hypothesised), whereas concerning the arrival station name six different hypothesised tokens are provided. A lot of branching in this part of the graph indicates that the ASR had difficulties with recognising the arrival station name.

Each hypothesised word in the word graph is assigned a score (corresponding to the number after the slash in the figure) that represents a certain confidence of the ASR in recognising that word at that position of the input. These confidence scores are derived from the speech signal and the language model. The best path of words is often selected from the word graph based on the recognition confidences. At the end of the recognition process the ASR yields a hypothetical transcription of the user input, typically consisting of one string of words (i.e., a 1-best word list). Confidence scores are furthermore often used in error detection (cf. [Litman et al. 1999, Walker et al. 2000b,

Litman et al. 2001]), although they turned out to not be fully utilisable since often there is no reliable correlation between a high confidence score and a correct recognition result [Hirschberg et al. 2004]; [Litman et al. 2000, Hirschberg et al. 2004] found that prosodic properties of the user input more reliably indicated speech recognition problems than confidence scores alone. For a detailed explanation on speech recognition for user interfaces see for example [Balentine et al. 1999].

LANGUAGE PROCESSING Methods for processing the linguistic structure of the ASR output can range from statistical to knowledge-based. Closely depending on the application's goal, the key task of language understanding in SDSs is to relate the processed input to the slots that need to be filled. In state-of-the-art NLU systems often heuristic techniques are implemented when it comes to interpreting user input, such as word- or concept-spotting (cf. for example [Aust et al. 1995, Allen et al. 1996]). The goal of concept spotting is to process the input for values that satisfy the slots in the system query, for example by searching for station names in the input. This technique fails in many cases when non-standard answers are provided by the users, for example when certain slot values are being corrected or rejected.

An effective solution for robust understanding may be the combination of statistical and knowledge-based techniques. For instance, [Cettolo et al. 1996] claim that the domain knowledge needed for understanding should be obtained in two ways: from the data itself, and from the expertise of the designer of an understanding module. Likewise, [Rayner and Hockey 2003] devise an interpretation architecture that combines data-driven and rule-based approaches and find that the hand-crafted rules serve as a back-off mechanism to which interpretation can retreat in case the data-driven method becomes unreliable (mainly due to data sparseness). Hybrid methods show their usefulness for understanding spoken input in speech-to-speech translation applications as well [Cattoni et al. 2001, Wahlster 2000]. The number of actual computational approaches to implementing NLU tools is vast; for an overview we refer to [Manning and Schutze 1999, Jurafsky and Martin 2000, Mitkov 2003]. No matter the actual approach taken, linguistic analysis of user input is supposed to yield a content-related representation of the input.

Empirical approaches to analysis rely on training data, and weight alternative analyses of strings based on some method that draws, e.g., on frequency counts, generated probabilities, rules, etc. The method used in our study is classification of natural language data, a bottom-up method for creating a model by identifying patterns in the data. One advantage of a bottom-up approach is that it can be domain or language independent to some extent, so that the method used for one language is transportable to other languages via re-training on the new language.

Traditionally, there are several processing subtasks in analysing spoken input, which are organised in a cascaded fashion, so that output of one module serves as input to subsequent modules. The layers of the cascade depend on the desired goal and the fine-grainedness of the computational analysis required by the actual SDS. Besides sequential modularisation it is possible to have more complex solutions used for the speech and the language processing parts, enabling these to directly influence each other's performance: the more information is received from components of the processing cycle, the more confident a certain interpretation of an utterance can be (see e.g. [Allen et al. 1996,

Zechner and Waibel 1998, Nakano et al. 1999, He and Young 2004]). Alternatively, parallel interpretation of different processing levels can make applications more robust, for example by making processing less prone to errors [Heeman 1998, Uszkoreit 2002]. Recently, researchers also began to devise applications whose goal is not to produce a transcribed word string, but to transform the speech signal into a representation of the main intentions of the speaker. This can be seen as a direct mapping from speech to dialogue act. Aspects of the work of [Nakano et al. 1999] could be considered as being such an attempt.

The current study shares its main line with these non-sequential approaches to the processing of user input, since we use properties of the ASR output and the dialogue manager to interpret user turns on several levels simultaneously. Nonetheless, we model a stand-alone NLU system, since our module has no access to the internal processes within the ASR and DM modules of a dialogue system. This situation often occurs when NLP modules are being developed for SDSs, since typically the various modules of a SDS are designed and deployed by different project teams.

2.2 Analysis levels in interpreting spoken user input

In the previous section we situated SI (shallow interpretation) of user input in the field of NLP. In the current section we give a survey on how data are collected and annotated to enable research on components of SI. An essential prerequisite of empirical research is the availability of (large collections of) material, in our case of spoken dialogue. Spoken dialogue corpora are built according to a number of design criteria that may depend on specific research aims: they may contain samples representative of conversational topic, diverse levels of situation spontaneity, speech register, dialectal language use, speaker gender, and the like. In other cases a corpus contains quite specific material, e.g., consisting solely of interactions with a given application. An important aspect of speech corpora is that besides the transcribed dialogue they contain audio material as well.

Typically, to enable research on the collected material, corpora are enriched with extra information on certain phenomena (again, depending on the research aims): the speech (transcriptions) are analysed and annotated, either manually or semi-automatically. Mark-up may be assigned to various levels of segmentation (word-, phrase-, sentence-, utterance level, etc.). This allows for examining patterns of the annotated categories, for developing rules that describe aspects of language use, and other types of empirical research.

Experts have created a number of international mark-up standards for corpus-based research: these are guidelines for orthographically transcribing spoken language, and to use annotation schemes for labelling (cf. [Gibbon et al. 1997]). The standards allow for more consistency in empirical research across different groups of scientists, providing guidance in many aspects of linguistic mark-up, as well as a starting point for creating one's own labelling scheme (as in our case). One of the broadest annotation standards to be mentioned is the MATE framework [Dybkaer and Bernsen 2000]. MATE was designed after reviewing more than 60 existing annotation schemes, encoding levels of prosody, (morpho-)syntax, co-reference, dialogue acts, communication problems, and cross-level issues, with the aim of developing a standard framework for annotating spoken dialogue corpora at multiple levels. For a thorough survey of dialogue data and annotation we refer

to [Popescu-Belis et al. 2003].

It is important to see that regardless of the standardised use of annotation, inconsistencies often occur in data labelling. This is on the one hand due to different perceptions of cross-categorical concepts (situated in different context). Inter-annotator agreement scores serve to reflect the level of consistency in the labelling of a corpus, cf. for example [DiEugenio and Glass 2004]. On the other hand, annotation inconsistencies also occur due to errors during the labelling process, since semi-automatic annotation is often used for large corpora. When evaluating corpus-based research results it has to be noted that inconsistency in mark-up may introduce a certain level of noise into the material. Another issue in data-oriented research is the amount of material available for exploration. It has been the goal of many empirical studies to find out in what way the scaling of training material contributes to optimal results; concerning NLP tasks see for example [Banko and Brill 2001, Curran and Osborne 2002, Van den Bosch and Buchholz 2002] and their references.

In the remainder of this section we look at how components of SI (the task-related acts as well as traditional dialogue acts, the slots and other information units, the source of communication problems, and awareness of communication problems) are annotated in speech corpora.

2.2.1 Task-related acts

The definition of task-related acts can be regarded as a nontraditional issue. Since it draws on the traditional notion of dialogue acts, in the current subsection we survey research pertaining to dialogue acts. The dialogue act (DA) of an utterance reflects the main intention(s) conveyed by the speaker in that utterance. Since DAs are typically defined and investigated on various levels of grain size, it has been found that segmentation of a user turn into smaller units is crucial for correctly identifying DAs (cf. [Traum and Heeman 1997, Finke et al. 1998, Nakano et al. 1999, Reithinger and Engel 2000, Cattoni et al. 2001]); a process which is however not trivially executable by automatic approaches (cf. e.g. [Stolcke et al. 1998b]). Annotation schemes for labelling DAs are typically very complex as they aim at capturing all types of actions that occur in dialogue; sometimes DA annotation even incorporates semantic concepts (cf. [He and Young 2004]).

A commonly used annotation scheme for communicative actions is DAMSL (Dialog Act Mark-up in Several Layers, [Allen and Core 1997]). The label set of DAMSL is designed to capture the multiple functions within speaker turns by marking turns along four orthogonal dimensions that reflect their purpose and role in the dialogue: communicative status (marking whether the turn is intelligible), information level (characterising the content of the turn on a meta-level), forward-looking communicative function (characterising the effect of a turn on the subsequent turn), and backward-looking communicative function (indicating how the turn relates to the previous turn). DAMSL is a deliberately simple but robust tag set. It is emphasised by the designers of the scheme that some turns can be multi-dimensional in a complex way, for which guidelines are offered that restrict the co-occurrence of certain labels.

Below we present the label supersets that belong to each dialogue dimension in DAMSL. Note that each superset includes more refined subcategories, those being the actual DAMSL

annotation labels. This indicates that the annotation scheme contains many fine-grained (nonetheless intended as all-purpose) categories of user intentions. For example, the category AGREEMENT includes the labels ACCEPT, ACCEPT-PART, REJECT, REJECT-PART, HOLD, and MAYBE.

- Communicative status: UNINTERPRETABLE, ABANDONED, SELF-TALK
- Information level: TASK ('doing the task'), TASK-MANAGEMENT ('talking about the task'), COMMUNICATION-MANAGEMENT ('maintaining the communication'), OTHER-LEVEL
- Forward-looking communicative function: STATEMENT, ASSERT, REASSERT, OTHER-STATEMENT, INFLUENCING-ADDRESSEE-FUTURE-ACTION, OPEN-OPTION, ACTION-DIRECTIVE, INFO-REQUEST, COMMITTING-SPEAKER-FUTURE-ACTION, OFFER, COMMIT, CONVENTIONAL, OPENING, CLOSING, EXPLICIT-PERFORMATIVE, EXCLAMATION, OTHER-FORWARD-FUNCTION
- Backward-looking communicative function: AGREEMENT, UNDERSTANDING, ANSWER, INFORMATION-RELATIONS

In the current work we similarly assign interpretations to whole user turns. Our aim in using DAs is to point out the main, task-related, pragmatic act exhibited by the user turn, which we call the task-related act (TRA). Since the goal is to carry out an abstract characterisation of the user turn by the TRAs, some of the categories in the set of TRAs are defined on the basis of DAs, whereas others stand for nontraditional types of user actions. It is important to see that TRAs concern only the information level of the user input (see the second superset in DAMSL). Our TRA labels can be regarded to pertain to the following information level supercategories in DAMSL:

- TASK (i.e., slot-filling in the SDS)
- TASK-MANAGEMENT (i.e., answering to meta-questions of the SDS)
- OTHER-LEVEL (i.e., providing confusing or irrelevant information to the SDS).

We are going to elaborate on our annotation scheme for TRAs in Section 4.2.

2.2.2 Information units

In the NLU module of a dialogue system usually a semantic parser is deployed that transforms the user's utterance into a formal semantic representation or a semantic frame. [Cettolo et al. 1996] explain that a semantic frame includes a frame type, which represents the main goal of the query (e.g., retrieving a train connection), and the slots, representing the constraints the query has to satisfy (e.g., origin, destination, etc.). For example, the

sentence ‘I want to travel from Amsterdam to Tilburg on the fourth of February’ might be translated into the following frame (cf. [Veldhuijzen van Zanten et al. 1999]):

$$\left[\begin{array}{ll} \text{destination} & \text{tilburg} \\ \text{origin} & \text{amsterdam} \\ \text{month} & \text{february} \\ \text{day} & 4 \end{array} \right] \quad (2.1)$$

The standard formalism for building such semantic structures is the grammar formalism of head-driven phrase structure grammar (HPSG) that employs typed feature structures [Pollard and Sag 1987, Pollard and Sag 1994]. Other, classical formalisms combining syntactic and semantic information are Montague grammar [Montague 1974] and generalised phrase structure grammar (GPSG) [Gazdar et al. 1985].

The semantic representation of an utterance may also be set out in the form of a propositional expression, since the semantic content of an utterance is traditionally computed according to a schematic notion of meaning, called the logical form, onto which the entered values are mapped (cf. [Allen 1995]). An example for this notation can be

`has_departure_time: [date, time= [day:4]]`

for the utterance segment ‘[travel] on the fourth’ (cf. [Reithinger and Engel 2000]). Semantic parsers are traditionally built using hand-crafted semantic grammar rules (cf. e.g. [He and Young 2004]), which may be combined with grammatical parsers (see for example [Van den Berg et al. 1994, Allen et al. 1996]). Further examples include the Verbmobil corpus in which the propositional content of utterances is converted into a modified form of HPSG-like semantics, using a domain description language that unifies several discourse representation structures [Bos et al. 1996], and a grammar formalism described in [Bonnema et al. 1997] where each word or phrase is associated with a feature structure, in which both syntactic and semantic information is represented in a combined way. Typically, these formalisms have to generate a semantic expression used to update the dialogue state in a SDS.

We observe that, similar to the annotation of DAs that are pragmatic in nature, annotation of information units in utterances takes place on various levels of detail in the literature. On the one hand, such content-related labels are of various levels of structural fine-grainedness themselves (cf. e.g. the embedded structure in the above notation), and on the other, we observe that the scope of utterance segmentation in labelling is also at variance; i.e., labels are assigned on the level of turn, phrase, word, and so on. For example, semantic roles may be assigned to syntactic constituents of a turn (cf. [Allen 1995]), and/or the contents of the turn can be annotated for the meaning they carry. [Weber and Wermter 1996] label each word in a corpus of interactions at a railway counter according to basic, task-related semantic categories such as LOCATION, DESTINATION, TIME, and the like. [Reithinger and Engel 2000] extract the contents of a turn from bigger sub-turn segments. [Rayner and Hockey 2003] define a set of semantic atoms that represent primitive domain concepts, as well as values of these concepts, specifying the set of legitimate combinations among these.

In our study information unit types are labelled on the turn level. The labels concern exclusively the task-related slots for which information is supplied by the user (more details on our annotation of slots will be provided in Section 4.2). Our aim is to identify the query concepts for which information is entered by the user (e.g., items similar to the ones in the left column in the expression shown in 2.1), without identifying the particular values associated with these concepts (e.g., the items in the right column in expression 2.1).

Studies dealing with semantic representation often either aim at detecting the full, deep semantic structure of some input, or keep their research at the level of DAs. In the latter cases, it is often observable that DA categories unify both pragmatic and semantic information from a domain (e.g., [He and Young 2004]), which may lead to many low-frequency labels. Our shallow approach attempts to eliminate such a skewed label distribution by defining labels that account for general pragmatic-semantic information types.

2.2.3 Forward-pointing problems

General mark-up of communication problems can be found in few works: research often focuses on some subgroup of communication problems since it is difficult to address a general class of problems. For example, in the MATE annotation scheme the labelling of communication problems proceeds in a detailed way whereby problems are “tagged as types of violation of the guidelines for cooperative spoken dialogue”. Such an encoding is however a non-trivial task to accomplish, and the designers explicitly state that it is difficult to analyse utterances correctly in order to “determine which guidelines they violate and how”; for example, a user supplying the time of the travel by saying ‘at 9 o’clock’ may count as violation of the cooperativity guideline that prescribes to avoid ambiguity, since not all parameters of the travel time, namely morning or evening, are fully stated by the user in this input.

[Aberdeen et al. 2001] describe a method for detecting errors in task-based human–computer dialogues by automatically deriving them from fine-grained semantic tags. This suggests that the components of our shallow understanding module such as the semantics-related information types and the pragmatics-related communication problems may be closely related to each other. Investigating whether such relations can be automatically discovered forms one of our research issues, since such information may play an important role in the class label design of the SI task.

Annotation of forward-pointing problems in the literature is diverse: we observe again that the annotation of what counts as a problem source defines a range of phenomena. Moreover, the level of segmentation at which these problems are annotated is of several grain sizes. [Litman et al. 1999] perform automatic detection of poor speech recognition at the dialogue level, tagging complete dialogues as featuring good or bad ASR performance. [Walker et al. 2000a] likewise tag whole dialogues as exhibiting task success or task failure, the latter consisting of either the user hanging up on the system, or a human operator interrupting the conversation, or an incorrect query retrieval by the system. [Litman and Pan 1999] identify sequences of dialogue turns as featuring good or bad ASR performance, whereas [Litman et al. 2000] annotate single dialogue turns as featuring good

or bad ASR performance.

In the study of [Walker et al. 2000b] dialogue turns are identified either as causing NLU errors or as being correctly understood. Two error classes are distinguished in this work: mismatch and partial match between the user input and what the system understood from it. The partial match category denotes cases when the user input's pragmatic-semantic aspect (defined as the act of referring to a task in the domain) is correctly recognised, but the deep semantic aspect (i.e., the actual slot value entered by the user) is misrecognised. [Hirschberg et al. 1999] annotate correctly and incorrectly recognised utterances scored by hand for semantic accuracy, while [Hirschberg et al. 2000, Hirschberg et al. 2004] identify ASR misrecognitions in terms of concept accuracy and word error rate. [Kamm et al. 1998] characterise user utterances in terms of recognition scores and ASR rejections.

The forward-pointing problem concept of [Van den Bosch et al. 2001] overlaps fully with that of the current study, since it served as a pilot study of certain issues of the present research. In the work of [Van den Bosch et al. 2001] user utterances are assigned PROBLEM or NO PROBLEM labels on the turn level, depending on whether these originate a communication problem on the conceptual level or not. Recall that our primary concern in problem detection is to discover whether the system attains perfect concept accuracy or not; in the latter case we talk about a communication problem.

2.2.4 Backward-pointing problems

Examining previous literature on backward-pointing problems in dialogue, we see that [Krahmer et al. 1999] investigate user turns in terms of their providing 'go back' vs 'go on' signals to the system, whereas [Levow 1998] and [Litman et al. 2001] both annotate problems in terms of system misrecognitions (specified as "erroneous system grounding" in [Litman et al. 2001]), or system rejections, aiming at distinguishing user reactions to them automatically. Problem annotation in [Van den Bosch et al. 2001], as noted above, lies fully in line with that of our study: backward-pointing problems are defined as the system's conceptual misinterpretation of the user input provided in the previous turn, which the user notices in the current turn.

In surveying the literature on annotating awareness of communication problems we observe that this phenomenon is often marked by labels representing dialogue acts. Namely, some of the back-channeling DA types defined in DA taxonomies correspond to what is researched as awareness sites. Such labels may indicate (indirectly) that one of the parties is having a difficulty in the conversation, and are typically called REJECT ('Well, no.'), NO-ANSWER ('No.'), SIGNAL-NON-UNDERSTANDING ('Excuse me?'), APOLOGY ('I'm sorry.') e.g. in the merged Switchboard-DAMSL encoding, cf. [Jurafsky et al. 1997]. Of course, not all of the utterances receiving the above tags signal problems, thus the decision about problem signalling requires close investigation of the whole dialogue. Our approach to detecting backward-pointing problems is to collapse the relevant subset of such utterances into a general backward-pointing problem category, instead of many fine-grained backward-pointing problem categories.

2.3 Potential information sources for interpretation

In the previous section we described the ways spoken corpus material is annotated on components of SI. In this section we pay attention to how other studies treat information sources available from speech corpora for learning these components. We examine what particular pieces of information are utilised for detecting DAs, information units, problem source, and problem awareness in user turns of task-oriented human-machine interaction.

The largest unit examined in this study is the speaker turn, which may consist of one or more utterances, or, characteristically of information-seeking dialogues, only of an elliptical (i.e., incomplete on some linguistic level) phrase. We aim at exploiting a wide range of contextual properties for the identification of the interpretation. Observing the overwhelming and successful utilisation of (word) n -gram sequences in NLU tasks, it is straightforward to assume that the main cue in discovering patterns in language is context. Context is aptly defined in [Bunt 2001] as “the totality of conditions that may influence the understanding and generation of communicative behaviour”, and [Allen et al. 1996] indeed suggest that the extensive use of context enhances robustness in NLU.

We henceforth refer to a user turn that needs to be interpreted as the ‘focus turn’. A focus turn’s context consists of a large number of attributes, such as the words contained by the focus turn and the preceding turns, the intonation with which these are uttered, the time span during which they are uttered, the prompt upon which the input follows, and so on. It is an empirical question which attributes are useful for automatically learning to interpret a user turn. The choice of utilising one or another property necessarily depends on what sources and types of information are regarded to be relevant for the underlying task either intuitively, or based on previous work. In empirical research the attributes used in assigning some representation to a user turn are often simply selected on grounds of their supposed predictive power towards the component(s) the representation contains.

In studies that treat components of SI, contextual attributes, also called cues, are employed with a wide range of grain size that range from primitive to sophisticated. Primitive cues are typically simple representations of whether a condition is true or false for an attribute, e.g., whether a certain word is present or absent in the focus turn. Sophisticated cues can be high-level linguistic concepts (e.g., syntactic information, semantic information) or meta-level concepts such as the identity of slots that have been treated in the interaction up to the current point, or the kind of grammar the ASR used in processing a given input. Naturally, the cues differ also in the effort that has to be made to obtain them: some are easily extractable in real time from various modules of the SDS (for example utterance duration), others are often computationally more expensive to obtain (for example syntactic information).

Attributes of the actual spoken and textual material in the turns constitute the perceivable and measurable context of dialogue turns. Besides, cognitive types of context are also present in a dialogue situation, such as world knowledge, beliefs, social obligations, and the like (cf. [Bunt 2000]). Such cognitive phenomena are difficult to optimally define and infer in a NLU system, often making it expensive to build a SDS when such context is implemented in it extensively. Moreover, cognitive context is mostly of use for the DM modules of the system that traditionally need to perform reasoning.

[Bunt 2001] argues that context can be optimally utilised only in case it is defined such

that it is “both sufficiently powerful to form an adequate basis and sufficiently restricted to be manageable”. Our research is an attempt to use context in a restricted but powerful way, unifying usefulness and low cost of contextual attributes when these are used in machine learning of SI. Before explaining our motivation and actual selection of cues (described in Section 4.3), the remainder of the current chapter surveys what attributes other studies utilise for analysing SI-related aspects of user input.

2.3.1 Cues in analysing task-related acts

The work of [Samuel et al. 1998a] provides a good overview of machine learning approaches to the computation of DAs. It notes that the attributes used in all surveyed studies include the *dialogue act labels of the preceding utterances*, since dialogue structure information, provided by DA sequences, is supposed to be predictive of the identity of the next DA. An important difference between such approaches and our work is that we do not use the computed TRAs of user input in the detection of the focus turn’s TRA, as this could accumulate error in the learning task (cf. [Qu et al. 1997]).

It is also observed by [Samuel et al. 1998a] that “some systems utilized basic features of the current utterance: *specific words* found in the utterance, the utterance’s *length* (number of words), and the *speaker direction* (who is talking to whom)”. Moreover, *lexical cues* are also often extracted from utterances in order to identify DAs [Samuel et al. 1998b, Choi et al. 1999, Keizer 2003], since for example the presence of the token ‘yes’ can indicate an AFFIRMATIVE DA, or the presence of ‘from’ may be predictive of INFO-PROVIDING, etc. *Prosodic properties* are utilised e. g. in [Jurafsky et al. 1996, Stolcke et al. 1998a, Taylor et al. 1998, Shriberg et al. 1998, Shriberg et al. 2001]. Prosody may play a supportive or disambiguative role in classifying one or another DA label. Evidence for this provided by [Stolcke et al. 1998a] is that a YES/NO QUESTION that is in statement form (i.e., includes no wh-inversion) is typically marked by a sentence-final rise of the voice pitch. At the same time, [Beun 1989] finds that in 20% of the cases when a question is posed in the form of a statement, no sentence-final pitch raise can be observed. The two contradictory findings indicate that some contextual cues that are found useful on some data set may not always generalise to other data sets, especially when research is conducted with non-robust methods involving a small data set.

Other widely used attributes in learning DAs include the *micro-syntax* of an utterance: verb tense, the presence of wh-inversion, auxiliary verbs, subject type, and the like, as well as punctuation marks, etc. may point to certain types of DAs [Jurafsky et al. 1996, Choi et al. 1999, Keizer 2003]. For instance, wh-inversion in some turn might indicate that the turn is an INFO-REQUEST, whereas in the opposite case the turn might be a STATEMENT, and so on. For further comparisons we refer to [Popescu-Belis et al. 2003] who provide useful pointers to a large number of studies on automatic dialogue act tagging.

2.3.2 Cues in analysing information units

[Traum 2003] emphasises that for understanding answers to questions properties of the *local dialogue structure* are needed. It is a common technique to first detect the DA of an utterance, and subsequently in a separate step identify information units, i.e., se-

mantic information, since the latter is often regarded as the argument of the DA (cf. [Reithinger and Engel 2000, Cattoni et al. 2001]). [He and Young 2004] however first use a semantic parser to process the output of the ASR, re-scoring the word graph n-best output, and subsequently identify the DA of the *most confident string*.

Typically, *statistical*, as well as *lexical properties of the ASR output* and those of *syntactic analysis* are widely utilised for performing semantic analysis (cf. for example [Van den Berg et al. 1994, Bonnema et al. 1997, Rayner and Hockey 2003]).

For an overview of knowledge sources used in computational interpretation of information units in the user input see [Flycht-Eriksson 1999]. This study concludes that knowledge sources utilised by SDSs are often not clearly separable from the actual dialogue model implemented in a system; the employed cues are often inherent to the implemented dialogue model, which yields limited reusability of such approaches.

2.3.3 Cues in analysing forward-pointing problems

Researchers have utilised a variety of contextual properties for identifying forward-pointing problems for SDS. Many of these are simple, such as the *lexical output of the ASR module* of the SDS [Walker et al. 2000b, Van den Bosch et al. 2001], as well as *prompt history* [Walker et al. 2000b, Van den Bosch et al. 2001]. Others use additional *system-internal information* that represents prompting strategy, or the NLP grammar implemented in the system [Hirschberg et al. 1999, Walker et al. 2000a, Walker et al. 2000b], as well as automatically extractable *acoustic cues* [Litman et al. 1999, Walker et al. 2000a], and confidence scores output by either the ASR module [Litman et al. 1999] or the dialogue manager [Walker et al. 2000b].

[Hirose 1995, Hirschberg et al. 1999, Hirschberg et al. 2000, Litman et al. 2000] find evidence that *prosodic properties* of user input are predictive of forward-pointing problems in SDSs. It was found that utterances produced with marked prosodic settings are typically prone to error [Oviatt et al. 1996, Swerts et al. 2000] – presumably because general-purpose recognisers are not trained to deal with a speaking style which differs critically from the ‘average’ speaking style on which these recognisers are trained. Detecting *hyperarticulation* (louder and higher voice, and slower speech rate) might therefore be a good way to spot forward-pointing problems. However, for some SDSs hyperarticulation is shown to cause no recognition problems [Batliner et al. 2003, Goldberg et al. 2003]. On top of this, some users are simply less well recognised than others, and research has found a number of prosodic properties distinguishing these people from others (cf. [Hirschberg et al. 1999, Hirschberg et al. 2000]).

Sophisticated attributes that need to be manually annotated are also employed in some studies: these mainly concern semantic content relating to the interaction. Such dialogue attributes represent *inconsistency* between system prompt and user reply, *topic shifts*, *salience-coverage*, and the like [Hirschberg et al. 1999, Walker et al. 2000b]. The effect of the user being an *experienced* user or a novice one is probably one considerable factor in this respect, making user-modelling an important part of SDS design. Low recognition scores and ASR rejections are reported to occur more often in the case of novice users than in the case of expert users (cf. [Kamm et al. 1998]). The *age* and the *gender* of a user may likewise contribute to recognition success or failure (see [Walker et al. 2000a, Privat 2003]),

since some voice types associated with these factors are better recognised than other voice types.

Speaking style is another dominant factor that can determine ASR success (cf. e.g. [Weintraub et al. 1996]). Speaking style can cause diversity with respect to prosodic, lexical and syntactic patterns in speech. Depending on the purpose of a given SDS, the domain of interaction in actual systems is mostly a quite limited one, which restricts speaking style to a to-the-point task-oriented conversation. However, there are also applications that allow for more spontaneous dialogue, necessitating large-vocabulary recognisers.

2.3.4 Cues in analysing backward-pointing problems

Arguably, communication problems and user reactions to them very much depend on the dialogue situation in which they occur. For instance, [Litman and Pan 1999] and [Swerts et al. 2000] have shown that some *dialogue strategies*, like user-initiated interactions, lead to more errors than others, and that, accordingly, some *system prompt types* are more likely to trigger misrecognitions than others. Research by [Krahmer et al. 1999] has brought to light that users may react markedly differently to errors occurring in explicit versus implicit *verification of information*. Furthermore, it turned out that the *speaking style* of users' first corrections of system errors is different from that of corrections that occur in a chain of corrections [Swerts et al. 2000].

The reason to investigate prosody for the purpose of error detection is motivated by the fact that it functions well as a cue to problems in human-human interactions (see e.g. [Shimojima et al. 1998]). Consequently, if it would be possible to automatically locate places in the dialogue where speakers switch to a special prosodic style, they can become indicative of errors. An important cue in spotting backward-pointing errors may be that in response to the system's processing errors people may sometimes react with a *hyperarticulate* speaking style. This tendency occurs widely when people are confronted with communication problems in interacting with other people, and these findings appear to generalise to human-machine interactions as well [Shriberg et al. 1992, Oviatt et al. 1998]. Additionally, the *wording*, *syntax*, *duration*, etc. properties, of such reactions can be markedly different from answers to non-problem-revealing prompts, see for example [Krahmer et al. 2001b]. Therefore, researchers have also started to explore whether prosody may be useful as a resource for error detection (see e.g. [Levow 1998, Litman et al. 2001]).

The aware turn of the user supplies important cues for detecting problems that originate in the previous turn. Studies that aim at spotting aware user turns in SDSs make extensive use of both primitive and complex cues. Primitive cues include *confidence scores* in the ASR module of the system [Litman et al. 2001], *lexical output of the ASR* module of the SDS [Van den Bosch et al. 2001], the *amount of slots filled* [Krahmer et al. 1999], *dialogue history* [Litman et al. 2001], as well as the presence of certain *lexical attributes* in the user input [Krahmer et al. 1999, Litman et al. 2001], and the presence of *repeated* lexical items [Hirschberg et al. 2001]. High-level features involve aspects of *syntax* in the user answer (utterance length, word order) [Krahmer et al. 2001b]. Attributes of the *preceding turns* [Litman et al. 2001], and *experimental parameters* and aspects of the underlying *ASR grammar* [Litman et al. 2001] are also often employed in automatic detection of user

awareness of communication problems.

2.4 Summary

In this chapter we have surveyed previous work on processing components of SI (task-related acts that pertain to dialogue acts, as well as semantic information units, backward- and forward-pointing communication problems) in two respects: how these are annotated in corpora, and what information is utilised in their automatic detection. We emphasised the differences between the approach of previous research and that of the current study: contrary to most of the surveyed work, we deliberately define general categories of task-related acts, information unit types, and communication problems. Additionally, our study utilises unsophisticated, low-level cues in the classification of the SI components, keeping the approach shallow, thus, supposedly, robust.

We conclude that our approach is more complex in its goals than most of the surveyed work, since we attempt the detection of a four-level representation of pragmatic and semantic aspects of user input to a SDSs. We hypothesise that this is a difficult task, however, we believe that predicting such a complex representation for new utterances improves natural language understanding in human-machine communication.

Chapter 3

Machine Learning as a Research Environment

The current study uses machine learning as a research environment for developing and testing modules that perform shallow interpretation of user turns in spoken dialogue systems. In this chapter we introduce the general empirical set-up in which shallow interpretation takes place. In the first section of the chapter we describe the algorithms employed: a memory-based ‘lazy’ learner, and an ‘eager’ learner, a rule induction algorithm. In the second section we pay attention to two methodological issues of machine learning that play an important role in our research, namely the evaluation of algorithm performance, and algorithm parameter selection.

Machine learning (ML), a research area within the discipline artificial intelligence (AI), provides an algorithmic approach to model a phenomenon by estimating its parameters on the basis of examples and to improve the performance of predicting new instances of that phenomenon. In case the learning algorithm is trained on examples that are labelled in terms of classes that collectively describe the phenomenon, we speak of supervised learning. In the current study supervised learning techniques are used; in this way we can make good use of our labelled corpus data.

In order to learn the model, a supervised ML algorithm processes the examples which typically consist of fixed-length feature vectors containing attributes, i.e., features, of the phenomenon in the form of conjuncted variable values, as well as the class that represents a distinctive category of the phenomenon. If the learning task is to classify the examples, the algorithm, called classifier, learns a function that maps the examples’ features to the set of classes. ML is a cornerstone of AI since learning algorithms (also called learners) are able to extract knowledge from the examples they are supplied with, and to improve with experience, which are primary characteristics of intelligence.

The training of an algorithm is the process during which knowledge is gained from the examples so that the algorithm becomes able to map the features to the set of predefined classes. Assessment of the adequacy of the learnt model takes place by testing the

algorithm on how well it classifies unseen examples (called test instances). Test instances consist of a feature vector but not the class, which needs to be assigned by the algorithm. The working principle of inductive ML is that a model that converges to the target function on the learning examples will do so on similar, previously unseen test instances. For an introduction to the theory of ML, primary ML algorithms, and examples of practical applications to real-world problems, cf. [Mitchell 1997].

Many studies have investigated the extent to which ML-based empirical methods can be utilised in natural language processing, and it is generally claimed that linguistic issues can be (re)formulated as learning tasks; see for example the collection of [Wermter et al. 1996], as well as [Daelemans et al. 1997], and their references. [Daelemans 1995] discusses that all computational NLP problems can be formulated either as a disambiguation task or a segmentation task: when performing classification of natural language data, the class symbols represent linguistic categories, respectively boundaries between linguistic units.

3.1 Algorithm choice

It is still an open issue which ML techniques are the most suitable for which NLP tasks. An important point to make is that the working principles, i.e., the bias of ML algorithms differ largely in terms of “what can be represented as an induced hypothesis, and how the search for a hypothesis is heuristically guided” [Daelemans and Hoste 2002]. For example, the knowledge acquired through learning can be largely different among classifiers in terms of comprehensibility, re-usability, and storage. As a result, the algorithm choice may depend on the research purposes: for example, one algorithm is more suitable for gaining a compact model of some phenomenon, another is for modelling low-frequency or irregular examples in a domain, a third is for utilising feature independence, and so on. On investigating a number of existing supervised classification ML methods on benchmark NLP tasks see for example [Márquez 2000, Zavrel et al. 2000].

At the same time, the recent work of [Daelemans and Hoste 2002] has provided empirical evidence that “interaction between algorithm parameter settings and feature selection within a single algorithm often accounts for a higher variation in results than differences between different algorithms or information sources” (the latter referring to the employed features). The fact that, irrespective of the learning task and an algorithm’s bias, performance differences between different algorithms might be of a much smaller scale than those by the same learner under differing conditions, suggests that for certain NLP tasks it might not matter significantly which algorithm is employed, given identical experimental conditions, and sufficient data. Contrary to this, [Rotaru and Litman 2003] have found that learning algorithms with a different bias can produce significantly different performance depending on several factors such as the task, the number of features, and the type of features.

In our study two ML algorithms with a different bias are trained and tested on all tasks. Our primary goal is not to plot a competition between ML algorithms, but, as set out in Chapter 1, to ensure the validity of our investigation as well as its generalisability to other ML algorithms, by applying two different biases to the same classification tasks. The two algorithms used by us, namely a memory-based learning algorithm, IB1,

and a rule induction algorithm, RIPPER, are representatives of different classes of machine learners: [Daelemans et al. 1997] regard memory-based learning and rule induction as extremes in terms of the amount of effort invested in learning and the technique of knowledge representation. Below we discuss the technical details pertaining to IB1 and RIPPER.

3.1.1 Memory-based learning

The IB1 algorithm is a memory-based learning algorithm, a descendant of the k -nearest neighbour (k -NN) classifier [Fix and Hodges 1951, Cover and Hart 1967, Aha et al. 1991, Cost and Salzberg 1993]. Memory-based learning is a type of ‘lazy’ learning, because the classifier simply stores a representation of all training examples in memory, without abstracting away from individual instances during the learning process. This stands in contrast with our other learner which is typically referred to as an ‘eager’ learner, see Section 3.1.2.

Memory-based learning algorithms classify new instances by looking for the most similar (i.e., ‘nearest’) examples in memory and extrapolating from their class the new instance’s class. Taking the classical k -NN approach to classification entails that when k is set to 1, IB1’s strategy is to return the class of the immediate nearest neighbour. The nearest neighbour is searched for among the training examples that are stored in memory: it is the memory example that has the least difference, according to a similarity metric, with the test instance. Memory-based learning algorithms thus do not search for a target function that covers all examples, but for a local function that is based on the examples closest to the focus (i.e., test) instance. Research found that due to its bias a memory-based classifier may yield more precise results than classifiers that discard low-frequency items from the induced knowledge model, in case these low-frequency items constitute exceptions that re-occur in test data [Daelemans et al. 1999].

The TiMBL software package [Daelemans et al. 2003] incorporates a variety of memory-based pattern classification algorithms, among others the IB1 algorithm (the default in TiMBL). We employ IB1 in the TiMBL package version 5.0.0, and henceforth refer to it as the memory-based learner, MBL. The classification procedure by MBL has two subprocesses: learning, during which examples are stored in memory, and classification, during which k -NN examples are found and subsequently the class is extrapolated from k -NNs to the new instance.

For classification of a test instance Y , the set of k memory examples closest to Y is determined. The amount of nearest neighbours defines the amount of memory examples that are used to extrapolate the class of the test instance. In MBL classification takes place by searching for the nearest distances (instead of nearest neighbours). Search for the nearest distances implies that when the same distance is measured for more than one neighbour, these are regarded as being equally similar to the test instance. In this way, if for example k is set to 1, the number of examples from which the class is extrapolated may be more than 1, since several nearest neighbours may occur at the same nearest distance. We will however refer to nearest distances throughout this study with the term ‘nearest neighbours’.

MBL computes the distance between a memory example X and the test instance Y for each feature according to some metric $\Delta(X, Y)$, so that the distance of X and Y is defined

as the sum of the differences between the features:

$$\Delta(X, Y) = \sum_{i=1}^n w_i \delta(x_i, y_i), \quad (3.1)$$

where n is the number of features, δ is the distance per feature — since x_i is the (value of the) i th feature in X and y_i is the i th feature in Y —, and w_i is a weight marking the i th feature's importance in the task.

The kernel of this distance function in our study can be one of the following four metrics:

- Overlap metric
- Numeric metric
- Modified Value Difference Metric (MVDM)
- Jeffrey divergence metric.

OVERLAP The overlap metric computes the distance function according to the following formula:

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i. \end{cases} \quad (3.2)$$

NUMERIC In case the feature values are numeric, and the feature is declared as numeric, the distance is computed by MBL as

$$\delta(x_i, y_i) = |(\frac{x_i - y_i}{max_i - min_i})|, \quad (3.3)$$

where max_i and min_i are the maximum value and the minimum value of the i th feature. The calculation of the numeric distance ensures that numerical feature values will be treated appropriately when computing their distance, i.e., will not be treated as symbols but as real numbers.

MVDM The Overlap distance metric regards values either as identical, or as different. However, in many symbolic tasks there is a graded dissimilarity of feature values: for example, the filling of the departure station slot may be more similar to the filling of the arrival station slot than to answering a yes/no question.

The MVDM distance metric assigns to each pair of values of a particular feature an index representing the distance between the values: the similarity of the values of a feature is determined by looking at the co-occurrence of values with target classes [Cost and Salzberg 1993]. For the distance between two values v_1 and v_2 of a feature the difference is computed as the conditional distribution of the classes C_i for these values:

$$\delta(v_1, v_2) = \sum_{i=1}^n |P(C_i|v_1) - P(C_i|v_2)|. \quad (3.4)$$

where n is the number of classes, and v_1 and v_2 are values of x (i.e., the distance is calculated for all value pairs of all examples in the training data). [Daelemans et al. 2003] warn that data sparseness may result in unwanted effects when the MVDM metric is used: if there are feature values occurring only a few times or once in the whole data set, and always with the same class, MVDM will regard those as identical, whereas if they occur with two different classes, their distance will be maximal. In such cases it is preferable to use the Overlap metric instead of MVDM. TIMBL offers such a back-off from MVDM to Overlap through a frequency threshold which is activated when one or both of a pair of matched values occur fewer times in the training data than this threshold.

JEFFREY DIVERGENCE Finally, the Jeffrey divergence metric computes the distance between class distributions of two values of a feature. Jeffrey divergence works similarly to MVDM, but instead of computing a geometrical distance between two class distribution vectors it uses a logarithm term:

$$\delta(v_1, v_2) = \sum_{i=1}^n (P(C_i|v_1) \log \frac{P(C_i|v_1)}{m} + P(C_i|v_2) \log \frac{P(C_i|v_2)}{m}), \quad (3.5)$$

where m is computed as

$$m = \frac{P(C_i|v_1) + P(C_i|v_2)}{2}. \quad (3.6)$$

Compared to MVDM, Jeffrey divergence assigns larger distances to value pairs of which the class distributions are more orthogonal so that zero probabilities become more marked, making Jeffrey divergence more robust on sparse data [Daelemans et al. 2003]. As with MVDM, it is possible to set a frequency threshold to back-off from Jeffrey divergence to the Overlap metric.

As yet another option of the distance function, it is possible to rank the features according to their estimated importance in the classification. This is done by assigning weights (represented by w_i in Equation 3.1) to the features, which is computed by a feature-weighting metric. Weighting features in k -NN by their classification prediction strength implies that examples are regarded as more similar to each other when they share more of the higher-weighted features. The weighting function used in the distance function can be one of the following:

- No weighting, all features have the same importance
- Information Gain weighting
- Gain Ratio weighting
- Chi-squared (χ^2) weighting

- Shared variance weighting.

INFORMATION GAIN (IG) IG is an information-theoretic metric, measured by computing the difference in *entropy* between the situations with and without knowledge of the value of the feature concerned. Entropy is a numerical measure of informativity, measuring uniformity in the representation of information [Shannon and Weaver 1949]. The entropy H is computed by estimating probabilities of class labels from relative frequencies in the training data:

$$H(C) = - \sum_{c \in C} P(c) \log_2 P(c), \quad (3.7)$$

where C is the set of class labels. The **IG** of feature i is thus measured as

$$IG_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v), \quad (3.8)$$

where V_i is the set of values for feature i . This entails that the more uniform the probability distribution of classes, the greater the entropy, i.e. uncertainty, in the task. IG tends to assign high weights to features with a lot of values, which may have an unfavourable effect on classification.

GAIN RATIO (GR) GR feature weighting aims at balancing this effect out via normalising the IG feature weight by the entropy of the feature value. It is an information-theoretic heuristic established by [Quinlan 1993]. To compute the GR of a feature, its IG is calculated and normalised for features with different numbers of values. GR is thus IG divided by si_i , standing for split info which equals to the entropy of the feature value:

$$GR_i = \frac{IG_i}{si_i}, \quad (3.9)$$

where

$$si_i = - \sum_{v \in V_i} P(v) \log_2 P(v). \quad (3.10)$$

χ^2 WEIGHTING The GR measure may still show an unwanted bias towards features with more values, since the GR statistic is not corrected for the number of degrees of freedom in the contingency table of classes and values; the χ^2 weighting metric may correct this [Daelemans et al. 2003]. The χ^2 statistic is computed by the following formula:

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})^2}{E_{ij}} \quad (3.11)$$

where O_{ij} is the observed number of cases with value v_i in class c_j , and E_{ij} is the expected number of cases which should be in cell (v_i, c_j) in the contingency table, if the hypothesis

that no predictive association exists between feature and class is true.

SHARED VARIANCE WEIGHTING Another method to correct for the degrees of freedom is to use the shared variance measure:

$$SV_i = \frac{\chi_i^2}{N \times (\min(|C|, |V_i|) - 1)} \quad (3.12)$$

where $|C|$ and $|V_i|$ are respectively the number of classes and the number of values of feature i , and N is the number of instances. For more details on feature weighting in the IB1 algorithm we refer to [Aha 1998, Daelemans et al. 2003].

CLASS VOTING Finally, in the process of extrapolating the class from k -NNs to the new instance, it is possible to define the prominence of each NN in voting for the new class. The weight of a NN is computed as a function of the NN's distance from the test instance, allowing closer NNs to have a more prominent vote in the classification. Class voting may take place on the basis of class majority (in this case all neighbours are assigned the same weight), as well as on the basis of linear, inversed, and exponentially-decayed distance weighting.

To summarise, MBL has the following important working parameters:

- number of nearest neighbours used for extrapolation (default: 1)
- distance metric (default: overlap)
- feature weighting metric (default: GR)
- class voting of the nearest neighbours (default: majority class voting).

MBL provides no direct explanation of its classification output, which is a disadvantage for understanding the results obtained. In order to point out problematic cases for the learner, classifier-internal logs can be examined that record the parameter use and its effect on the test data: by observing the NNs it is possible to determine on which basis MBL extrapolated the class, providing an indirect explanation for the decision.

3.1.2 Rule induction

In contrast to the 'lazy' learner MBL, our other classifier is an 'eager' learning algorithm, RIPPER [Cohen 1995], used in version 2.5. RIPPER is a rule induction algorithm that is designed to be fast and efficient even on noisy datasets. The original rule induction algorithm on which it draws is described in [Fürnkranz and Widmer 1994]. The bias of rule induction is to discover regularities in the data and represent those by the simplest possible rule set. The induced rule set is then used to classify new instances.

The most dominant type of rule induction algorithm is the sequential covering algorithm, of which RIPPER is one variant. The kernel of the classical sequential covering

algorithm is to incrementally build up a set of rules that collectively cover all positive examples (cf. [Clark and Niblett 1989, Michalski et al. 1986]). The algorithm learns one rule at a time, after which the examples covered by this rule are removed from the training set, and a new rule is learnt. Each rule states that satisfying a condition (which, depending on the employed grammar, may be a conjunction of conditions, or a conjunction of disjunctions, etc.) implies membership in a particular class: **if** *<feature test>* **then** *<class>*. In each iteration rules are built up greedily. The algorithm starts with the most general rule that covers all instances. Then a set of extensions is generated in which conditions on attribute-value pairs are added to the initial rule. These extensions are evaluated by some measure, such as accuracy of prediction (i.e., the number of correct classifications) over the instances covered by the conditions, or entropy. The goal is to find a rule that has high accuracy, but not necessarily high coverage [Mitchell 1997].

This routine is repeated until some stopping criterion is reached, for example that all training examples are covered, or the performance of the rules does not improve any more on some test part of the training data. The set of rules is then ordered according to some criterion (e.g., from low coverage to high coverage). Note that this may influence effectivity, i.e., learning time, as when a new instance needs to be classified, the rule set is traversed from top to bottom to search for the first rule that fires.

The approach of rule induction to classification is also related to that of decision tree learning (see [Quinlan 1986, Quinlan 1993]) as both aim at discovering patterns in data by some heuristics, for example based on accuracy or coverage metrics. However, in contrast with decision tree learning that generates an embedded tree structure in a parallel fashion, rule induction algorithms generate ordered lists of rules sequentially.

Since the sequential covering algorithm does not backtrack (a property that is also inherent in decision trees), it is not guaranteed to find the smallest or best set of rules [Mitchell 1997]. Therefore, and also to prevent overfitting of the rules (meaning that they would fit the training data well but the test data less well), pruning is often used as post-processing the learnt rule set (again, similarly to decision tree learning) [Fürnkranz 1997].

The drawback of rule induction algorithms, described in [Cohen 1995], is that they cannot work optimally if the sample size is small. However, they are very powerful and fast general learning tools, and can reach very good results, especially when the number of classes in the data is small. An advantage of rule induction algorithms is that by reading the induced rules it is possible to interpret the generated output, which provides an explanation about the model's estimated parameters. RIPPER is often used in studies that deal with problem detection in human-machine communication (cf. [Litman et al. 2000, Walker et al. 2000b, Litman et al. 2001, Hirschberg et al. 2004]).

Below we describe the kernel of our rule induction algorithm, RIPPER, to which we henceforth refer as the rule induction learner, RI. RI starts learning by separating the training set in two. On the basis of one part it induces rules, maximising coverage and accuracy for each rule, where the employed heuristic is to minimise entropy in the data set by each rule induced. According to RI's default rule grammar, the condition part of each rule may consist of one or more conjoined feature value tests. The heuristic used by RI for growing the rules is to add a test on a feature value if using that condition results in more accurate segmentation of the data, which is estimated by the IG function. When the induced rules classify instances in the test part below a certain threshold, they

are not stored. Rules are induced per class; by default their ordering proceeds from low-frequency classes to high-frequency ones, leaving the most frequent class as the default rule (which is generally beneficial for the size of the rule set). The stopping criterion RI uses draws on the idea of the minimum description length [Rissanen 1978]. The minimum description length principle states that the length with which regularities in the data are described corresponds to the success of discovering those regularities, so that regularity in the data can be used to compress it, i.e. to describe it using fewer symbols than needed to describe the data literally [Grünwald et al. 1998]. RI uses this as a heuristic to decide if the induced rule set needs to be pruned. It post-prunes the generated rules by the reduced error pruning technique [Cohen 1995].

When classifying a new instance, the rule set developed by RI is traversed from top to bottom. As soon as a rule fires (i.e., its feature-value test conditions match with those of the test instance), the class of the rule is returned and the traversal through the list is stopped.

In sum, RI amalgamates entropy minimisation in the form of sequential covering and generation of rules in accordance with the minimal description length, as well as some heuristics, such as pruning. Below we list the parameters most significant for our study in RI:

- amount of learning examples to be minimally covered by each rule (default: 2)
- hypothesis simplification (simplify more/less, default: simplify less, i.e., multiply coding cost of theory by 0.5)
- negative tests on feature values allowed or disallowed (default: disallowed)
- number of optimisation rounds on the induced rule set (default: 2)
- class ordering (default: order by increasing frequency)
- loss ratio of false positives/false negatives (default: 1)
- expect data noisy/non-noisy (default: expect noisy).

Hypothesis simplification in RI is possible by setting the coding cost of the theory generated by a grammar based on the minimum description length principle. Negative tests on feature values enable to assign a class on the basis of an instance not having a certain feature value. Optimisation rounds are carried out per rule set per class, for example by merging similar rules into a more general, pruned rule. Setting the loss ratio manipulates a cost function in RI that determines the trade-off between the false positives (i.e., instances falsely selected for a particular class, see below) and the false negatives (i.e., instances falsely unselected for a particular class, see below) of the rules, thereby determining the importance of the type of misclassification for a class.

Additionally, RI also allows for declaring features as numeric. This option enables segmenting the data by rules with ‘smaller_than’/ ‘larger_than’ conditions (i.e., discretisation) at numeric value boundaries, which might be beneficial for the classification task as opposed to ‘has_value’ treatment of numeric features (which is for example the default in MBL).

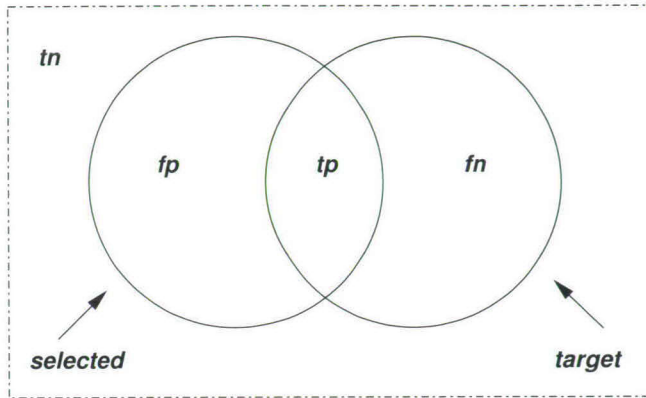


Figure 3.1: A diagram facilitating the illustration of our evaluative measures, reproduced from [Jurafsky and Martin 2000]. ‘tn’ denotes true negatives, ‘tp’ denotes true positives, ‘fn’ denotes false negatives, ‘fp’ denotes false positives.

3.2 Experimental methodology

In our general experimental set-up, training and testing is done by 10-fold cross-validation. The data are randomly split into ten partitions, each roughly the same size. This is carried out by means of dialogue-based partitioning, thereby ensuring that no material from the same dialogue could be part of the training and the test set. Learning experiments are conducted so that each partition acts as a test set once, while nine-tenth of the data serves as training material. n -fold cross-validation (CV) is a generally accepted method in the ML community for conducting and presenting performance measurements on some task, see for example [Weiss and Kulikowski 1991]. 10-fold CV allows for estimating classification performance on previously unseen material, measuring the prediction strength of a learner.

In our study the performance of learners will be evaluated according to four measures. To illustrate how these are computed, consider Figure 3.1 that we reproduced from [Jurafsky and Martin 2000]. The target set contains the instances that need to be assigned a particular class, the selected set contains the instances to which the learner assigned a particular class. Based on the diagram, a contingency table can be made, which we present in Table 3.1. Based on the diagram and the contingency table, the evaluative measure accuracy can be seen to represent the percentage of correctly classified test instances:

$$Acc = \frac{tp + tn}{tn + fp + tp + fn}. \quad (3.13)$$

The other three measures, precision, recall, and F-score, are common measures of performance in information retrieval, and are somewhat newer in NLP. Precision is the ratio of correctly classified instances in a class to the total number of instances identified as

	Target	Non-target
Selected	tp	fp
Not selected	fn	tn

Table 3.1: Contingency table of evaluating a classifier’s prediction in terms of target and non-target classes. ‘tn’ denotes the amount of true negatives, ‘tp’ denotes the amount of true positives, ‘fn’ denotes the amount of false negatives, ‘fp’ denotes the amount of false positives.

members of the class:

$$Pre = \frac{tp}{tp + fp}. \quad (3.14)$$

Recall is the ratio of correctly identified instances in a class to the total number of instances in the class:

$$Rec = \frac{tp}{tp + fn}. \quad (3.15)$$

The F-score metric represents the harmonic mean of precision and recall, of which we employ the unweighted variant, defined as

$$F = \frac{2PreRec}{Pre + Rec}. \quad (3.16)$$

In evaluating a classifier’s performance the F-score value (on a particular class, or proportionally computed based on all classes) is often more informative than predictive accuracy: accuracy can be opaquely biased to the majority class(es), whereas the F-score characterises the rate of precision and recall for the prediction of the target, penalising for disharmonic divergence between precision and recall [Van Rijsbergen 1979].

For evaluative purposes often a baseline learning approach is conducted on the data. The baseline strategy typically employs some straightforward heuristic, for example to assign the majority class label of the training instances to all test instances. Baseline learners in this study will be defined per learning task.

3.2.1 Algorithm parameter optimisation

Both MBL and RI have parameters that bias their performance. It is unknown beforehand which (combination of) parameter settings yield the best generalisation performance on some task. Research, among others by [Daelemans and Hoste 2002], shows that it is often not optimal to use the default parameter settings of a learning algorithm, whereas it is beneficial to tune the settings to suit the type of data and task better. Ideally one would

want to tune algorithm parameters automatically: such a procedure however contains a search problem for finding optimal parameter settings given a particular data set and a particular task, where the search space to be explored can be large since it consists of all possible combinations of parameters.

The method of wrapped progressive sampling (WPS, [Van den Bosch 2004]) offers a solution to this, involving a heuristic search algorithm. The procedure implemented in WPS includes finding a set of optimised algorithmic parameters for a range of machine learning algorithms ('classifier wrapping', [Kohavi and John 1997]), combined with progressive sampling of training data [Provost et al. 1999], by testing decreasing amounts of setting combinations on increasing amounts of training data. The approach is claimed to search the space of parameter setting possibilities considerably more thoroughly than economic search heuristics such as Monte Carlo sampling (e.g., [Samuel et al. 1998a]).

WPS makes an estimation of optimal parameter settings by performing experiments on the training material itself (since it is not allowed to use test material to make that estimation): parameter setting combinations are tested on a random 20% of the training data. The best performing settings, evaluated by performance accuracy, are retained and re-run in another round of testing, already on a larger amount of data. This routine is iterated over a growing amount of data, producing a list of accuracies from which badly-performing setting combinations are discarded. The process is iterated until only one parameter setting is left, or, in case several settings remain, either the default setting is returned if it is among the settings, or a random selection is made from the settings.

The method is reported in [Van den Bosch 2004] to show little improvement on benchmark tasks for algorithms that offer few parameter variations, but may yield marked improvements for algorithms offering many possible parameter combinations. We tested the effect of WPS in [Lendvai et al. 2003], where the performance of MBL on classifying disfluent language phenomena was found to increase from 95.7% to 97.0% accuracy and from 72.3 to 80.0 F-score when optimised algorithm parameters (combined with an attenuation technique, see Section 6.1.1.3) were used instead of the defaults. For more details on WPS we refer to [Van den Bosch 2004].

We employ WPS version 1.0 throughout the experiments in the current study, unless stated otherwise. This entails that for both classifiers a learning process will consist of two parts per data partition: parameter search, and the experiment itself, in which the parameter setting estimated as optimal by WPS is applied to the full 90% training set, and is tested on the yet unseen 10% test set.

For MBL the following metrics are optimised in WPS, testing with the indicated values (in total 925 setting combinations):

- number of NNs used for extrapolation: 1, 3, 5, 7, 9, 11, 13, 15, 19, 25, 35
- distance metric: Overlap, MVD, Jeffrey divergence, the latter two with frequency thresholds 1 and 2
- feature weighting: no weighting, IG, GR, χ^2 , shared variance weighting
- NN weighting for class voting: majority class voting, linearly-inversed distance

weighting, inverse distance weighting, exponential-decay distance weighting with α set to 1, 2, or 4.

For the RI algorithm the metrics and values tested in WPS are the following (in total 648 setting combinations):

- amount of learning instances to be minimally covered by each rule: 1, 2, 5, 10, 20, 50
- multiply coding cost of a hypothesis with 0.5, 1.0, 2.0
- negative tests on the feature attributes: allowed, disallowed
- number of optimisation rounds on induced rule set: 0, 1, 2
- class ordering: increasing frequency, decreasing frequency
- loss ratio of false-positives/false negatives: 0.5, 1.0, 2.0
- expect noisy data, expect non-noisy data.

It is important to emphasise two points concerning the use of WPS in our study. First, we expect that the resulting optimised parameter settings are often going to differ per experiment (i.e., per data partition); such a tendency simply indicates that the data are heterogenous across the data sets, which is common knowledge, and entails that the parameter settings may not be reused on new data, since they do not fully generalise. It is exactly this eventual diversity in the resulting optimal parameters that enables us to estimate the generalisation performance of the method of optimising algorithm parameters, preventing to report on results produced by an overfitted set of parameters. At the same time, even if there is variation in the selected parameters for different cross-validations, it is possible to decide on a final setting to be used for new test data, for example by conducting parameter search on the complete data set, or by reusing the settings that are found optimal for the majority of the cross-validations.

Second, it is not the primary aim of the current study to measure the gain WPS adds to the shallow interpretation method, rather, we use WPS as a tool that is incorporated as a plug-in in both classifiers. Therefore, we report on the resulting optimised algorithm parameter settings only in case these provide non-trivial insight into the nature of the SI task.

3.3 Summary

In this chapter the methodology employed in our study was described. In Section 3.1 the issue of algorithm choice was explained, after we gave a description of the two classifiers used throughout the study. We emphasised that memory-based learning and rule induction are

regarded as extremes in terms of their learning effort and classification effort, and it is unclear, given the contradictory findings of [Daelemans et al. 1999, Daelemans and Hoste 2002] and [Rotaru and Litman 2003] whether it can be expected that the performance of memory-based learning and rule induction is going to be different on our task.

In Section 3.2 we provided the details of the general experimental set-up used in this work: we perform 10-fold CV combined with wrapped progressive sampling in the experiments, so that MBL and RI are trained and tested under identical conditions. Classification performance is measured and evaluated on the basis of four measurements: accuracy, precision, recall, and F-score. We pay most attention to the figures characterising the F-score obtained on the task, since this figure represents an informative aggregate of classification precision and classification recall.

Chapter 4

Shallow Interpretation Module: Data, Experiments, and Results

In this chapter we present the architecture of the SI module, as well as the results of applying ML algorithms to the complex task of interpreting spoken user turns. After introducing our research material, the OVIS corpus, we explain the process of annotating user turns in the corpus according to the four components in SI (Section 4.2). We then describe the design of the ML experiments. We explain the class label design for our task in Section 4.2. Subsequently we describe the features and give an account of how they are obtained in Section 4.3. The chapter is completed by reporting on the results of ML experiments, performed both with MBL and RI (Section 4.4.2). We summarise the experimental outcomes in Section 4.5.

4.1 OVIS

4.1.1 The OVIS system

In this section we introduce our research material, collected from interactions with the OVIS dialogue system. ‘OVIS’ is an acronym for ‘Openbaar Vervoer Informatie Systeem’ (Public Transport Information System). The blueprint for the OVIS experimental spoken dialogue system for Dutch was based on the German Philips automatic train timetable information system [Aust et al. 1995]. OVIS was developed in the Dutch national research project ‘Language and Speech Technology’ which ran from 1995 to 2000, funded by the Dutch Organisation for Scientific Research (NWO). The modules of the OVIS system were developed by different groups at different sites. The project’s goal was to develop a telephone-interfaced speaker-independent SDS that travellers could call by telephone for enquiries on train connections in the Netherlands. The SDS had three development versions. OVIS1 was implemented in 1995. For speech processing a word-transition-based, statistical language model was implemented in the ASR. For language processing two alternative NLP modules were developed: a data-oriented (i.e., probabilistic), and a

grammar-oriented (i.e., rule-based) parsing module. The goal of OVIS1 was to build a demonstrator via which realistic human-machine dialogues could be acquired, to facilitate further development of the OVIS system [Boves et al. 1995].

In later phases of the project two more versions were developed of the SDS. In OVIS2 the ASR was made adaptive to enable interaction between the ASR, NLP, and DM modules. The major difference between OVIS2 and the subsequent installment, OVIS3, was in the capability and quality of the system components [Boves et al. 1995]. For details on the components of the OVIS architecture see the following literature: [Strik et al. 1997] on speech recognition, [Van Noord et al. 1999] and [Bonnema et al. 1997] on the NLP components, [Theune 2003] on language generation, and [Veldhuijzen van Zanten 1998] on the dialogue manager of the system. Furthermore, [Veldhuijzen van Zanten et al. 1999] provide an evaluation of the NLP components of OVIS2.

The OVIS1 system prompted the user for four slot values in order to retrieve information from an on-line database of train timetables. In particular, the user needed to provide the departure and arrival station names, as well as the date and time (i.e., hour and minute, and, if needed, time of the day) of either the departure or the arrival. The strategy employed was primarily system-initiative, however, it allowed the user to provide unsolicited information in reply to any of the prompts. The OVIS1 system always gave feedback to the user on what it had understood from the user input, by means of either implicit or explicit verification prompts. This entailed that users could (in principle) always become aware of communication problems from the following system question. The system was able to retrieve more than one connection, from which it presented the first suitable one, asking the user afterwards if she wanted to hear the remaining connections as well. Dialogues could in principle continue (i.e. a new query could be started) after some connection was retrieved from the database.

4.1.2 The OVIS corpus

During the development of the OVIS1 demonstrator a corpus of dialogue transcriptions with syntactic and semantic annotations was compiled. This material consisted of transcribed dialogues that were sampled from a range of telephone calls of test users of OVIS. The test users knew that they were subjects in an experiment and were requested to call the system regularly. Figure 4.1 is an illustration of the type of dialogues contained in this corpus. We indicate the implicit or explicit verification prompts in the figure. We provide here the English translation of the system and the user turns; the original Dutch transcription is given in Figure 2 in the Appendix.

The material used in the present study consists of 441 transcribed dialogues coming from the corpus of user interactions with OVIS1. We henceforth refer to this material as the OVIS corpus. Audio files containing the recorded user input, the transcribed, semantically annotated user utterances, as well as the ASR output containing word graphs were available to us for research. The dialogues amount to 3,738 pairs of system and user turns. We had no access to user identification numbers, therefore it cannot be stated to what extent these dialogues involve different speakers. By listening to the acoustic material however, we can infer that the vast majority of the dialogues are conducted with different users. The number of turn pairs in the dialogues range from 2 to 29, the average

Turn	Utterance	Ver. type
S1	Good morning. This is the automatic information system of public transportation. This system provides information exclusively about train travels from a National Railways station to a National Railways station. From which station to which station do you want to travel?	
U1	I want to go from Voorschoten to The Hague Central station.	
S2	When do you want to travel from Voorschoten to Utrecht Central station?	IMPL
U2	I want to go to The Hague Central station.	
S3	When do you want to travel from Voorschoten to The Hague Central station?	IMPL
U3	The day after tomorrow.	
S4	At what time do you want to travel the day after tomorrow?	IMPL
U4	Somewhere in the evening.	
S5	So you want to travel between eighteen and twenty-four hours?	EXPL
U5	Yes.	
S6	I am sorry I did not understand you. Could you repeat whether you want to depart between eighteen and twenty-four hours?	EXPL
U6	Yes.	
S7	There are five connections. This is the earliest: (...). Do you want me to repeat the connection?	
U7	Yes, repeat it.	
S8	I am sorry I did not understand you. Could you say again if you want me to repeat the connection?	
U8	Yes.	
S9	I repeat the connection. (...) Do you want me to repeat the connection?	
U9	No.	
S10	I am sorry I did not understand you. Could you say again if you want me to repeat the connection?	
U10	No, don't repeat.	
S11	I am sorry but again I did not understand you. Could you say again if you want me to repeat the connection?	
U11	No.	
S12	Do you want to have information about the next connection?	
U12	No.	
S13	Do you want to have information about another connection?	
U13	No.	
S14	I am sorry I did not understand you. Could you say again if you want to have information about another connection?	
U14	No, I don't want to know another connection.	

Figure 4.1: English translation of dialogue nr. 001/007 sampled from the OVIS corpus. The original Dutch transcription is provided in Figure 2 in the Appendix. System prompts are marked for verification type, where IMPL stands for implicit verification and EXPL for explicit verification.

number of turn pairs is 8. The number of words uttered in a user turn ranges from 1 to 15, the average number of words being 3. 43.2% of the turns are inaccurately recognised by the system, which seems to be somewhat higher than the misrecognition rate reported for other systems (e.g. in [Barkhuysen et al. 2005, Hirschberg et al. 2004]) who claim that 30-32% of misrecognised user utterances is generally representative of speaker-independent SDSs in real life settings. For details on the linguistic aspects of user turns in the OVIS corpus see [Van Noord et al. 1996].

The following sections focus on how the corpus material was made processible for machine learning experiments. This includes the labelling of the four components of SI in the user turns, as well as the extraction of features that the learners draw on in classification. The class label, i.e., the output side of the SI module is explained first.

4.2 Class label design

The four SI components in each user turn are labelled in terms of four sets of simple and straightforward labels. The labelling process is carried out automatically, since it was possible to draw on two earlier annotations of the OVIS corpus: [Veldhuijzen van Zanten 1996] and [Van den Bosch et al. 2001]. Manual annotation in terms of the label sets would have also been possible simply by observing the transcribed dialogues. Below we describe in detail the inventory of our labels.

4.2.1 Task-related act labels

The first label set consists of labels that represent the TRA in a user turn. As indicated earlier, these labels stand for basic acts on the information level of the dialogue that a user may perform in information-seeking dialogues, conducted with a SDS of a dominantly system-initiative prompting strategy. Our approach to defining the TRAs is to consider task-oriented communication as consisting primarily of transitions between a small number of distinguishable states [Feinman 1997] that on the user's part correspond to basic answer types (modelled e.g. in [Levin et al. 2000]). We find that five labels are sufficient to represent the basic task-related acts in the OVIS corpus. These user TRAs are as follows:

- s ('slot-filling'), provide information with respect to the query (e.g. 'from Amsterdam to Tilburg')
- Y, give an answer that expresses affirmative input in the given dialogue context (e.g. 'yes', 'that's right', 'indeed', 'please do', etc.)
- N, give a negative answer (e.g. 'no thanks', 'it's not necessary', 'go back', 'this is incorrect', etc.)
- A, accept incorrectly verified information (e.g., by not signalling a system error)
- NSTD, give a non-standard reply (e.g., to remain silent, to provide a fully irrelevant input).

These task-related acts incorporate a lot of the traditionally established dialogue acts: for example, *s* can often be seen as pertaining to one of the categories STATEMENT, ASSERT, REASSERT, etc. (in the Forward-looking communicative function of the dialogue) defined in DAMSL (cf. Section 2.2.1), and also to the categories AGREEMENT, UNDERSTANDING, or ANSWER (in the Backward-looking communicative function in DAMSL). However, we simply denote by *s* that on the Information level of the dialogue the user is ‘doing the task’ (quoting the definition of DAMSL), i.e., is filling the required slots.

Likewise, *Y* and *N* denote input in which the user is ‘talking about the task’ (in terms of affirmation and negation) with the system, since he or she answers yes/no questions or meta-questions with an affirmative or a negative act. Traditionally, most of the *Y* and *N* utterances could be regarded as ASSERT (note that this category incorporates the DAs ACCEPT, ACCEPT-PART, REJECT, REJECT-PART, HOLD, and MAYBE, cf. Section 2.2.1), or RE-ASSERT in the Forward-looking communicative function of the dialogue defined in DAMSL, and as UNDERSTANDING, ANSWER, or INFORMATION-RELATIONS in the Backward-looking communicative function, and so on.

Note that the *N* act does not always indicate that the user rejects a system verification; *N* can also be a simple negative answer (e.g., a refusal) to a yes/no question such as ‘Do you want to know another connection?’. Likewise, a *Y* act does not always indicate that the user confirms the system prompt as being correct: observe in Table 4.1 that part of the turns labelled as acceptance (i.e., *A*) co-occur with the affirmative (i.e., *Y*) TRA. Since the user’s conducting a *Y* or an *N* TRA might be an important indication that the interaction is unproblematic, respectively problematic, we regard it useful to explicitly learn whether the user input exhibits these acts (combined with other TRAs, if applicable).

Even more so (and partly concerning these labels), as by observing the interactions in our corpus we conclude that the ASR of this system is often unable to confidently recognise input containing ‘yes’ and ‘no’, even if these are contained in the word graph. Furthermore, [Hockey et al. 1997, Krahmer et al. 2001a] show that answering a yes/no question without including ‘yes’ or ‘no’ overtly may still communicate a clear ‘yes’ or ‘no’ meaning — we will find out whether our approach can capture this phenomenon. At the same time, ‘yes’ and ‘no’ are lexical items that are often used as cues in automatic DA classification (for example in [Keizer 2003], cf. Section 2.3.1), and it is important to investigate to what extent the presence of these items can be automatically detected.

It may happen in human-machine interactions that users accept incorrect system verifications. Acceptance may take place by explicitly uttering ‘yes’, but also by simply not saying explicitly to the SDS that there is a problem (e.g., by objecting to the verified information). Since the words uttered in an acceptance turn, and sometimes in a non-standard input, are often identical to those in a regular (i.e., *s*) input, it will be important to see whether it is possible to find other, not necessarily verbal cues (to be explained in Section 4.3) to detect acceptance.

In many cases more than one TRA label can be assigned to a user turn. Next to acceptance, another example is the user input ‘No, I want to depart at 11 in the evening.’, in our annotation *N;S*: the user employs both negation and slot-filling within the same turn. Combinations of the basic TRA labels yield four more labels (*A;Y* *A;S* *Y;S* *N;S*), totalling to nine TRA labels for our data.

TRA Label	Occurrence
S	2,033
N	693
Y	516
N;S	177
A;S	153
NSTD	81
A;Y	66
Y;S	19

Table 4.1: Occurrence of task-related act labels in the OVIS corpus, sorted by frequency.

Table 4.1 shows the frequency of TRA labels in the OVIS corpus. The most frequent TRA label is s: 2,033 turns are labelled to exhibit only slot-filling activity. There are 693 turns that are labelled as negative TRAs, and 516 turns that are labelled as affirmative TRAs. The remaining labels occur as indicated in the table. Note that acceptance of system errors occurs 153 times in combination with s and 66 times with y, but never in isolation.

The fact that we define the pragmatic acts of the user only on the Information level of the dialogue is related to our practical goals: by incorporating TRAs in the SI module we hope to interpret basic, task-oriented notions in the input of a user who interacts with a limited-domain SDS, rather than to set up a framework of dialogue act definitions, which is already extensively attempted (cf. the survey in Section 2.2.1), or to reason about the intentions of the user and its effects on the interaction, which is typically taken care of by the DM module of the SDS. Our hypothesis is that defining TRAs in an unsophisticated way is more robust with respect to the end result of the SI module than a fine-grained approach, and also more optimal for the portability of the approach.

Therefore, the TRAs in our work represent a limited set of answer types in task-oriented dialogue with a system-initiative SDS (i.e., the user only supplies answers to the system prompts), without being concerned about intentions behind utterances, or effects of utterances. We assume that these categories are able to reflect core information level actions taken in interacting with an information-providing SDS in such a way that detecting these is sufficient to contribute to interpreting the user input in human-machine interactions.

4.2.2 Information unit labels

Our second label set consists of shallow semantic labels that concern the task-related information units (i.e., the slots) for which information is supplied by the user. These are the following:

- v ('vertrek', departure station)
- A ('aankomst', destination station)

- D (day_of_travel)
- T (time_of_day_of_travel, i.e., morning, afternoon, evening)
- H (hour_and_minute_of_travel).

Note that these are the slots in the travel query the system needs to fill in order to perform the database search, therefore these labels will always co-occur with a slot-filling activity (defined by an s label, Section 4.2.1).

Two further labels are added to this set for technical reasons. In case no slots are treated in the turn, the label

- VOID

is inserted into the annotation. This applies to all cases when the user does not perform slot-filling, for example when he or she answers a yes/no question. The index

- @

marks that the D, T, or H value entered refers to the arrival part of the travel, and not to the departure, which is the default for the above slots. If, for example, to the prompt ‘When do you want to travel from Amsterdam to Tilburg?’ a user said ‘I want to arrive in Tilburg at eight in the evening.’, the slot labelling is ATH@, whereas if the user turn is ‘At eight in the evening.’, the slot labelling is TH. ‘@’ occurs 88 times in user turns in the corpus, distributed among various combinations of the D, T, and H labels (which are not among the most frequent labels shown in Table 4.2).

Our seven slot labels may also combine with each other, as often more than one slot is being filled in a turn, for example because the system may also ask for multiple slots simultaneously. This happens typically in reply to the opening prompt (‘From which station to which station do you want to travel?’) that prompts for V and A simultaneously. The number of unique slot labels totals to 30. Table 4.2 shows the ten most frequent slot labels in the OVIS corpus. The most frequently occurring slot label is VOID (1,356 times), whereas 14 labels are assigned to less than 10 turns in the data.

It is important to note that user turns such as ‘I want to travel in the evening.’ and ‘I want to know the last connection.’ are both labelled as T. Similarly, if the user says ‘I want to travel now.’, this is regarded as slot filling of H. Likewise, if the user says ‘I want to know the return trip.’, the turn is labelled as slot filling of V and A. In this way (and similarly to the TRA labels) our classifiers will be forced to learn cases where several types of wording (including relative time references and elliptic constructions) refer to the same concept. Another characteristic of our labelling scheme is that slots that are being corrected by the user are marked only in case information is (re-)entered for those. For example, the turn ‘I did not say Amsterdam.’ is labelled VOID, whereas ‘Not to Amsterdam but to Amersfoort.’ is labelled as A (i.e., arrival station slot).

4.2.3 Forward-pointing problem labels

For annotating communication problems two labels suffice: ‘problem’ and ‘no problem’. These two labels mark communication problems in the forward-pointing dimension. We

Slot Label	Occurrence
VOID	1,356
VA	917
D	453
H	262
V	225
A	109
TH	90
DT	57
DTH	53
DH	53

Table 4.2: Occurrence of the ten most frequent slot labels in the OVIS corpus.

label each user turn as problematic (PROB) if it gave rise to incorrect system reactions, or OK if it did not. The majority of user turns are unproblematic in the forward-pointing dimension: 2,125 turns are annotated as OK, whereas 1,613 turns are annotated as PROB.

4.2.4 Backward-pointing problem labels

Our fourth label set annotates the backward-pointing dimension of communication problems. The PROB label here is associated with user utterances that follow a question–answer pair in which the user’s answer caused some communication problem, for instance (and most often) because it was misrecognised. Therefore, the PROB label of the fourth SI component identifies the point at which the user became aware of the communication problem, since he or she has just heard a system prompt not in accordance with information provided in the previous (or earlier) exchanges in the dialogue. As above, the label OK is used to annotate cases when no communication problems occur in the backward-pointing dimension.

The majority of the user turns are unproblematic in the backward-pointing dimension: 2,125 turns are annotated as OK, whereas 1,613 turns are annotated as PROB. The identical distribution of labels in the forward- and backward-pointing dimensions is due to the fact that all user turns that are labelled as problem source yield an incorrect system response, which, due to the verification strategy of this system, can in principle be noticed by the user from the following prompt. Note that such a principled difference between the two problem dimensions may not have the same ‘trivial’ label distribution in other SDSs: if the verification strategy of a SDS is not always immediately verifying information, then users may not become aware of communication problems from the immediately following system prompt.

4.2.5 Global class label

The global class label design in our machine learning experiments is the following. Each user turn is assigned one label consisting of the four components in SI: user turns are

Global Label	Occurrence
N_VOID_OK_OK	399
Y_VOID_OK_OK	372
S_VA_PROB_OK	296
S_VA_PROB_PROB	222
S_VA_OK_OK	197
S_D_OK_OK	163
N_VOID_OK_PROB	133
S_VA_OK_PROB	129
N_VOID_PROB_OK	101
S_V_PROB_PROB	99

Table 4.3: Occurrence of the ten most frequent global class labels in the OVIS corpus.

represented as a combination of task-related acts, slots, forward-, and backward-pointing problems. This means that one symbol incorporates all four components of the SI. The labels of the four components are concatenated so that the general format of the class assigned to a user turn is

TASK-RELATED ACT_SLOT_FORWARD-POINTING PROB_BACKWARD-POINTING PROB.

The number of different labels composed of the four components is 148 in the OVIS material. This means that our classifiers will learn to classify 148 different classes in the data. Table 4.3 shows the ten most frequent global class labels in the OVIS corpus. The most frequently occurring slot label is N_VOID_OK_OK (399 times). This means that the most common user reply in the OVIS corpus is a negative answer (possibly to a yes/no system question), which is going to be processed correctly by the system, and which does not show awareness of communication problems. The second most frequent class label is an affirmative answer which is going to be processed correctly by the system, and which does not show awareness of communication problems (Y_VOID_OK_OK). The third most frequent class label marks filling the slots for departure and arrival, showing no awareness of communication problems (e.g., this might be the first user turn), that is going to be erroneously processed.

Figure 4.2 shows the annotation of an example dialogue from the OVIS corpus. The user turns in this figure are annotated according to the four components of the SI task. System prompts are analysed in terms of TRAs and slots in a similar fashion, to be described in Section 4.3.1.

4.3 Feature design

We utilise a number of cues in our machine learning experiments for learning the combined, four-layered SI class. In this section the input side of the SI module is described: how feature values are extracted from the corpus data and how they are preprocessed. The primary hypothesis underlying our study is that shallow interpretation correlates with

Turn	Utterance	Annotation
S1	Good evening. This is the automatic information system of public transportation. This system provides information exclusively about train travels from a National Railways station to a National Railways station. From which station to which station do you want to travel?	Q_VA
U1	I need to go from Schiphol to Nijmegen on Tuesday next week.	S_VAD_PROB_OK
S2	From where to where would you like to travel on Tuesday twelve December?	Q_VA;I_D
U2	From Schiphol to Nijmegen.	S_VA_OK_PROB
S3	At what time do you want to travel from Schiphol to Nijmegen?	Q_H;I_VA
U3	Around quarter past eleven in the evening.	S_TH_PROB_OK
S4	So you want to leave around eleven thirty-eight in the morning?	E_TH
U4	No, in the evening.	N;S_T_PROB_PROB
S5	Please correct me.	M
U5	No, I would like to leave at eleven fifteen in the evening.	N;S_TH_PROB_PROB
S6	So you want to leave at twenty-three thirty-eight?	E_H
U6	Yes, approximately.	A;Y_PROB_PROB
S7	I'm sorry I didn't understand you. Could you repeat if you want to leave at twenty-three thirty-eight.	RE_H
U7	At twenty-three fifteen.	S_H_OK_PROB
S8	So you want to leave at twenty-three fifteen?	E_H
U8	Yes. [hangs up]	Y_VOID_OK_OK

Figure 4.2: Dialogue nr. 004/005 sampled from the OVIS corpus. The original Dutch transcription is provided in Figure 3 of the Appendix. User turns are annotated by our label set on the four components of SI: TRA, slot, forward-, and backward-pointing problem, respectively. System prompts are analysed in terms of TRAs and slots (to be explained in Section 4.3.1).

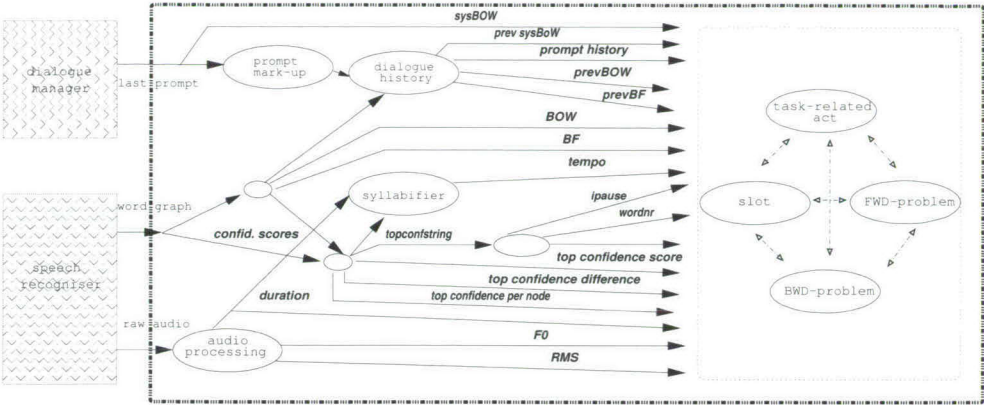


Figure 4.3: The architecture of the SI module.

shallow properties of the user input and its context that can be automatically obtained from the SDS. We turn such properties into a vector of features. The features are deliberately shallow: no explicit linguistic processing is needed for obtaining them. These simple features are obtained from the dialogue system and the audio material recorded by it. The values associated to these shallow features for each turn are employed in the experiments either in their raw form, or are computed via straightforward procedures.

Figure 4.3 shows the architecture of the SI module: feature values are obtained from the dialogue manager and speech recogniser modules of the system. They are then either directly used (illustrated by arrows leading directly to the back-end of the module where shallow interpretation takes place), or they are processed by simple procedures (indicated by the oval nodes), which are explained below. Note that for graphical reasons the various F0 and RMS features are represented by two single arrows, without enumerating all measures (see the bottom section of Table 4.4). In the remainder of this section we explain the employed features, their processing, and how they are represented in the feature vector.

Table 4.4 lists the employed features according to their origin: whether they come directly from the dialogue manager (DM) or the speech recogniser (ASR) of the system, or whether they come from prosodic processing of the audio recording of the user input made by the ASR (Prosody). Since some of these features are represented by more than one bit in the feature vector, the resulting feature vector of each user turn consists of 2,482 items. Note that some of our features may encode certain pieces of information in a (partly) redundant way, for example the bag-of words representation of the system prompts, and the prompt history feature.

4.3.1 Source: Dialogue manager

In this section we enumerate the features obtained from the DM of the system. The dialogue manager component of the SDS prompts the user to enter slot-filling information. The logged history of system prompts in the DM component contains the order of

prompts as they were produced by the system. From the DM we used the words that are present in the current and the previous system prompt. These are turned into two 467-bit unstructured bag-of-words (BOW) vectors. Each bit in the BOW stands for a word that occurred at least once in the prompts (in total 467 words), indicating whether a word is present ('1') in the current/previous system prompt or not ('0'). In case the turn is the first turn of a dialogue, the BOW vector of the previous system prompt is empty, and therefore consists of zeroes.

Furthermore, we encode system prompts in terms of a set of structured labels. We represent the prompts in the feature vector in a similar fashion as user turns are labelled: in terms of TRAs and slots. Basic TRAs in prompts include the following in this particular SDS (their mark-up between brackets):

- asking a question (Q)
- performing explicit verification (E)
- performing implicit verification (the simultaneous occurrence of a question and a verification: Q;I)
- repeating a prompt (R)
- asking a meta-question (M)
- offering travel advice (final result, FR).

Note that these labels overlap extensively with traditional dialogue acts (cf. e.g. the DAMSL encoding described in [Allen and Core 1997] and treated in Section 2.2.1, as well as the Switchboard-DAMSL encoding [Jurafsky et al. 1997]), differing in this respect from the TRAs that are used to represent classes of user input in pragmatic terms. It is important to see that the labels of system prompts are only employed as feature values in the classification task, and never form part of a class label.

The prompted slots partly overlap with those defined for user turns (see Section 4.2.2). These are the following: the departure and arrival stations (V and A, respectively), the corresponding day, time of day (i.e., morning, noon or night) and hour (standing also for hour and minute) of the departure (D, T, and H, respectively). The time slots may be prompted for simultaneously by the system ('wanneer'; *when*, Q_DTH), or in isolation (e.g., 'hoe laat'; *at what time*, Q_H). Additional prompted slots mark cases when the system asks whether the user wants to have the travel advice repeated (repeat connection, Q_RC), or whether the user wants to have information about another connection (Q_OC), the next connection (Q_NXC), and so on. In case the user enters information about the arrival time, time of day, or day (instead of the default departure time, time of day, or day), the system is able to verify slots with respect to the arrival data, which we again mark with the '@' sign. The dialogue in Figure 4.2 shows prompt analyses along these lines.

To represent dialogue history, we extract the current and the nine previous system prompt types. This can be seen as a (partial) representation of the dialogue history: a sequence of ten prompts form a set that is usually large enough to contain all the prompts the system posed to the user up to the focus turn. Furthermore, as the given

Aspect	Feature
DM: prompt DM: lexical	<ul style="list-style-type: none"> ▷ sequence of last 10 prompt types (<i>prompt history</i>) ▷ bag-of-words of current prompt (<i>sysBOW</i>) ▷ bag-of-words of previous prompt (<i>prev sysBOW</i>)
ASR: confidence ASR: branching ASR: lexical	<ul style="list-style-type: none"> ▷ highest summed confidence score in current word graph (<i>topconf</i>) ▷ highest summed confidence score normalised by number of nodes in path (<i>topconfpernode</i>) ▷ score difference between most confident and second-most confident path in current word graph (<i>topconfdiff</i>) ▷ branching factor in the word graph of current utterance (<i>BF</i>) ▷ branching factor in the word graph of previous utterance (<i>prevBF</i>) ▷ bag-of-words of current user turn (<i>BOW</i>) ▷ bag-of-words of previous user turn (<i>prevBOW</i>) ▷ word string in most confident path in current word graph (<i>topconf-string</i>) ▷ length of most confident string (<i>wordnr</i>)
Prosody: pitch Prosody: loudness Prosody: duration Prosody: speech rate	<ul style="list-style-type: none"> ▷ maximum F0 (<i>F0max</i>) ▷ minimum F0 (<i>f0min</i>) ▷ position of maximum F0 (<i>F0maxpos</i>) ▷ position of minimum F0 (<i>F0minpos</i>) ▷ mean F0 (<i>F0mean</i>) ▷ standard deviation of mean F0 (<i>F0stdev</i>) ▷ maximum energy (<i>RMSmax</i>) ▷ position of maximum RMS (<i>RMSmaxpos</i>) ▷ mean RMS (<i>RMSmean</i>) ▷ standard deviation of mean RMS (<i>RMSstdev</i>) ▷ duration of turn (<i>dur</i>) ▷ duration of initial pause (<i>ipause</i>) ▷ tempo (<i>tempo</i>)

Table 4.4: Overview of the employed features.

SDS employs an immediate verification strategy, it is very unlikely that a prompt history of more than ten steps could contain relevant information with respect to the focus turn (cf. [Koeling 2002] on the contribution of limited dialogue history to NLP tasks). In case the dialogue up to the focus turn contains less than nine previous prompts, the remaining, non-existing prompts are marked by a special null symbol in the feature vector.

Note that the prompt history encodes various pieces of information in an implicit (therefore shallow) way: for example, the slots for which the system thinks it has acquired the correct value, the number of times a prompt is repeated in the course of the interaction, recurrent prompt sequence patterns that possibly indicate dialogue substructures, and so on.

Prompt structure	Occurrence
Q_VA	555
Q_DTH;I_VA	359
FR;Q_RC	354
RQ_VA	270
Q_OC	244
Q_H;I_D	224
RQ_DTH;RI_VA	130
Q_V;I_A	118
RQ_V;RI_A	107
E_D	96
E_H	85

Table 4.5: Occurrence of the ten most frequent system prompt structures in the OVIS corpus.

4.3.1.1 Co-occurrence of turn pairs

After representing the system turns according to this scheme, we performed a simple analysis on our data counting co-occurrences of labelled system prompts and the corresponding user answers. Note that the figures in this analysis pertain to the specific OVIS application; however, the scale of occurrences may be indicative for other SDS as well. The primary aim of this subsection is to describe the OVIS corpus in more detail, not to present general findings about human-machine task-oriented dialogues.

The number of different system prompts in the OVIS corpus is in total 94. The ten most frequent structures are shown in Table 4.5. The most frequent structure of system prompts is Q_VA ('From which station to which station do you want to travel?', 555 cases). We see that asking for the time of the travel and simultaneously verifying the departure and arrival stations is the second most frequently asked prompt in this SDS (Q_DTH;I_VA, 359 cases), whereas the third most frequently occurring prompt is the one providing the user with travel advice and asking whether the user would like to have this repeated (FR;Q_RC). The fourth most common prompt is to repeatedly ask for the departure and arrival stations. Further down the list are prompts such as asking the user whether to start a new query ('Do you want to know another connection?', Q_OC), asking for the time of travel, while simultaneously verifying the day of travel (Q_H;I_D), and so forth.

Given this information, it is interesting to examine the patterns formed by adjacent pairs of system prompts and user responses. We find that there are 708 different pair combinations between system and user turns in the corpus; this illustrates well that there is no obvious mapping from a system prompt to a particular user reaction, indicating that modelling prompt and answer correspondence in a classification task is probably quite ambitious.

In order to illustrate how these pairs combine, we compiled Table 4.6 that shows the ten most frequent pairs of system prompt and user replies. It turns out that the most common system turn – user turn pair is Q_VA – S_VA_PROB_OK: the system asks for the slots of

Prompt structure	User turn class label	Occurrence
Q_VA	S_VA_PROB_OK	290
FR;Q_RC	N_VOID_OK_OK	230
Q_VA	S_VA_OK_OK	193
RQ_VA	S_VA_PROB_PROB	156
Q_OC	N_VOID_OK_OK	134
Q_DTH;I_VA	S_D_OK_OK	115
RQ_VA	S_VA_OK_PROB	83
Q_OC	Y_VOID_OK_OK	65
FR;Q_RC	N_VOID_PROB_OK	55
Q_DTH;I_VA	S_D_PROB_OK	52

Table 4.6: Occurrence of the ten most frequent pairs of system prompt and user reply in the OVIS corpus.

departure and arrival station and the user fills these slots which are erroneously processed by the system, whereas the user turn exhibits no awareness of previous communication problems — probably because the input occurs at the first turn of a dialogue. This prompt and reply combination is present 290 times in the corpus. It is a noteworthy finding that most frequently a user answer to the opening prompt in this SDS is erroneously processed.

The second most common turn combination is FR;Q_RC – N_VOID_OK_OK with 230 occurrences: presenting the travel advice to the user, and asking whether to repeat the connection, to which the user answers with negation that is going to be correctly processed by the system, and that exhibits no awareness of a previous communication problem. The third most frequent pair is Q_VA – S_VA_OK_OK in the corpus (193 times), which represents yet another user reaction type to the opening prompt: the user provides the departure and arrival station names, showing no awareness of previous communication problems, and this slot-filling is correctly recognised.

It is noteworthy that in reply to the system question about travel time ('when'), combined with implicitly verifying the departure and arrival stations most users provide only the travel day (in unproblematic context, i.e. Q_DTH;I_VA – S_D_OK_OK, 115 times, line 5 of the table). Expecting such a user answer is usually not trivial in SDSs that are designed on the basis of hand-made rules. This suggests that data-driven approaches to SDSs may be useful, among others to account for potential deficiencies of hand-crafted rules in such applications (cf. [Rayner and Hockey 2003]).

Likewise, it is noteworthy that to the system prompt 'From which station to which station do you want to travel?' 22 different answer types can be found in the OVIS corpus. In 290 cases these are S_VA_PROB_OK, in 193 cases S_VA_OK_OK, 13 times S_VA_PROB_PROB, etc. Note that in five cases the users answered to this prompt by filling in the date of the travel (S_D_OK_OK). The ten most frequent user reaction types to the opening prompt are shown in Table 4.7. This illustrates our earlier note on the fact that the opening system turn elicits user input that is a problem source in the majority of the cases in the corpus. What seems to aggravate this situation is that if the user reply to the opening prompt is

Prompt structure	User turn class label	Occurrence
Q_VA	S_VA_PROB_OK	290
	S_VA_OK_OK	193
	S_VA_PROB_PROB	13
	S_VAD_PROB_OK	11
	S_VA_OK_PROB	8
	S_V/A_PROB_OK	7
	S_D_OK_OK	5
	S_V_PROB_OK	4
	S_D_PROB_OK	4
	S_V_OK_PROB	3

Table 4.7: Occurrence of the ten most frequent user reaction types to the opening prompt in the OVIS corpus.

Prompt structure	User turn class label	Occurrence
RQ_VA	S_VA_PROB_PROB	156
	S_VA_OK_PROB	83
	S_V_PROB_PROB	11
	S_V_OK_PROB	7
	S_V/A_OK_PROB	2
	S_A_PROB_PROB	2
	N_VOID_PROB_PROB	2

Table 4.8: Occurrence of the seven most frequent user reaction types to the repeated opening prompt in the OVIS corpus.

not confidently recognised and the SDS repeats this prompt, 14 different answer types are received from users, and the majority of those are again incorrectly recognised. The seven most frequent user reaction types to a repeated opening prompt are shown in Table 4.8, the remaining seven answer types occur only once in the corpus. Some user reactions are labelled `S_V/A_OK_PROB`. In these cases the user provides only one station name to the system, and it is impossible to determine even in context whether it is the departure or the arrival station name. Such user input occurs 16 times in the corpus.

It is also intriguing to look at the various ways how users react to explicit verification prompts. For example, to the system's explicitly verifying the day of travel (`E_D`) 16 different user replies are annotated in the corpus. 52 times `Y_VOID_OK_OK` replies are received, 17 times `Y_VOID_PROB_OK` replies; Further answer types to this prompt include `N;S_D_OK_PROB` (5 times), `A;Y_VOID_OK_PROB` (4 times), etc. The ten most frequent user reaction types to the explicit verification prompt of the day of travel are shown in Table 4.9.

In [Lendvai and Maruster 2003] we conducted further research on the OVIS material (note that our corpus contained 442 dialogues in this study instead of 441). We applied process mining techniques to the data in order to discover relations between system prompts and user answers. The method analysed global interaction processes taking place with this particular SDS, inducing an interaction model underlying these dialogues in the form of a dependency/frequency graph. In this graph it was possible to specify those system prompts during the various stages of the dialogue that received more problematic than non-problematic user input.

We reproduce the interaction model discovered by this method in Figure 4.4. The frequency counts of user input types are given in the node labels. Note that only the TRA, slot, and backward-pointing problem ('pr') components are marked in the user labels. Forward-pointing problems are represented by the arcs that lead to problematic input. These are printed in bold to differentiate them from unproblematic forward-pointing relations. Frequency counts are given for the arcs as well. A dialogue progresses from the top of the graph, starting with the opening system prompt (label `Q_VA`) to the bottom. Two end state labels are present in the graph, `OK_END` for successfully ending dialogues, and `PR_END` for unsuccessfully ending dialogues where no query result was provided to the user. Since this graph represents a simplified interaction model based on a subset of the dialogues, the frequency counts differ from the ones reported in the current study. The graph illustrates well that even for a relatively simple dialogue strategy (i.e., prompting for a limited number of slots) the user action space can be relatively large, and mostly not trivially predictable.

4.3.2 Source: Speech recogniser

The ASR component of OVIS produced a number of features that can potentially be useful for SI. In particular, the material available to us contains the word graph associated to each user turn in the corpus. The word graph is often easily accessible from internal recognition logs in most SDSs. It has several properties that could be of potential use for extracting pragmatic-semantic information (cf. Sections 2.1 and 2.3). Apart from the words that it contains, the sequencing of these can also be extracted from the lattice (if

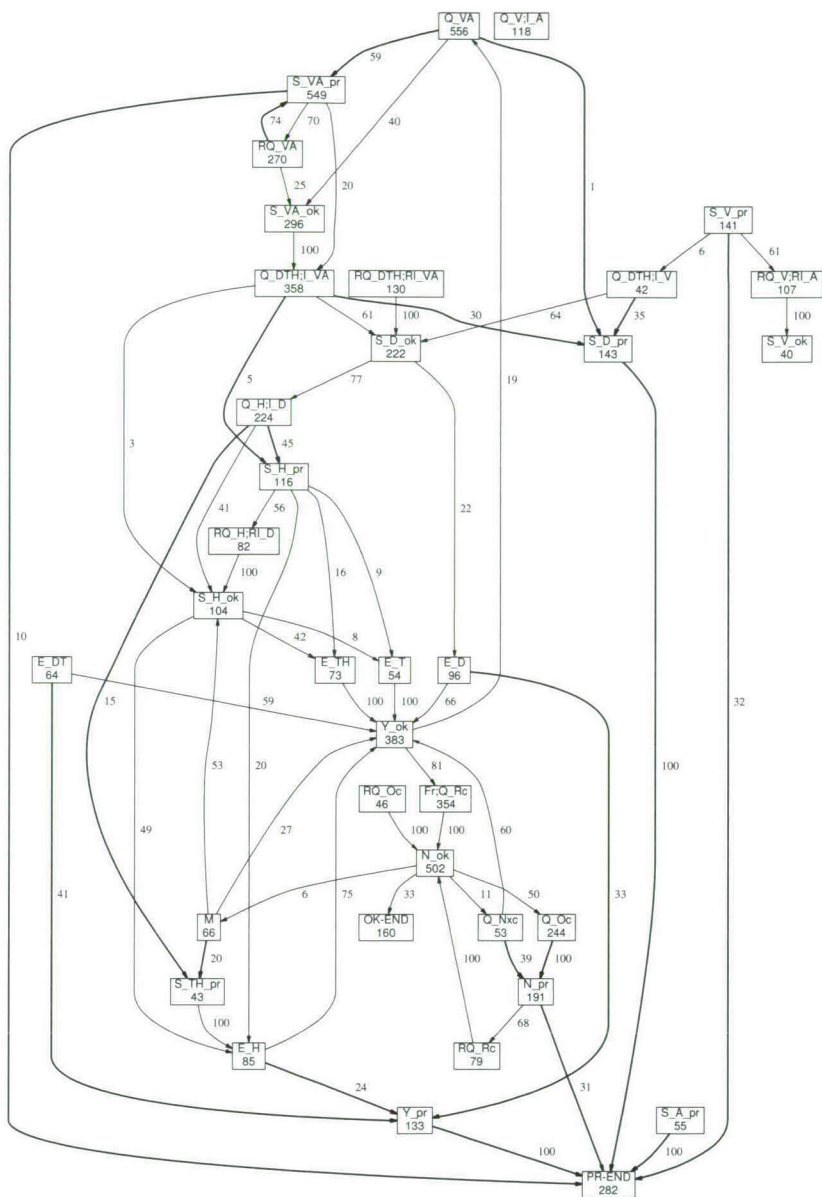


Figure 4.4: Simplified interaction model induced from the OVIS corpus by process discovery techniques described in [Lendvai and Maruster 2003]. Arcs leading to problematic input in reply to a system prompt are printed in bold.

Prompt structure	User turn class label	Occurrence
E_D	Y_VOID_OK_OK	52
	Y_VOID_PROB_OK	17
	N;S_D_OK_PROB	5
	N_VOID_PROB_PROB	4
	A;Y_VOID_OK_PROB	4
	S_D_PROB_PROB	2
	N_VOID_OK_PROB	2
	N;S_D_PROB_PROB	2
	Y_VOID_PROB_PROB	1
	Y_VOID_OK_PROB	1

Table 4.9: Occurrence of the ten most frequent user reaction types to the explicit verification prompt of the day of travel in the OVIS corpus.

the graph is turned into n -best paths). The confidence scores for each recognised word can likewise be obtained from the word graph. Structural properties of the word graph can also be computed, for example the thickness of the graph, represented by the number of simultaneous arcs in it, or the degree to which the arcs branch off, and so on.

For our study from each word graph we extract the recognised words (including the potentially incorrect ones) and encode these as two 759-bit, unstructured BOW vectors. The 759 bits represent all words that occurred at least once in the word graphs (i.e., forming the set of words recognised in the corpus). In each BOW vector we indicate whether a word is present in the corresponding word graph ('1') or not ('0'). Note that the BOW representation of the speech recognition results yields a larger set of material to exploit, than only the highest-ranked recognition result. At the same time, the way in which this complex information is represented in our study is utterly shallow (i.e., unsophisticated), since no information pertaining to (syntactic) structure, word form, or frequency is represented in the BOW.

We represent the recognition results of the previous user turn as well, since these may contain potential cues for the prediction of one or another SI component. For example, overlapping word hypotheses in the current and the previous user turn may signal that the user re-entered some slot value because it was incorrectly recognised, and so on. If the focus turn is the first turn of the dialogue, the previous word graph is empty.

From the word graph we furthermore extract the degree of branching both in the current and the previous word graphs, measured by the amount of branch-offs from the nodes. No branching means there is only one path in the graph. This branching factor characterises the degree of confusion in the word graph by indicating the average branching per node in the graph; much branching may be an indication of system uncertainty or noisy user input. Note that if the focus turn is the first turn of the dialogue, the previous branching factor is undefined, marked by a special character in the feature vector.

As illustrated in Figure 4.3, the confidence measurements of the ASR are also converted into features: we sum the confidence scores over the word transitions in the lattice, and

the path in the lattice that has the highest summed confidence score is used to create three separate features: we use the highest summed confidence score in the word graph itself, the concatenated string of words in the most confident path, as well as the number of words in the most confident path. Moreover, we compute the confidence score difference between the most confident and second-most confident path. Confidence scores are often used in classification of user turns, especially for error correction (cf. Sections 2.3.3 and 2.3.4) although these system-internal measurements do not always provide reliable cues (cf. Section 2.1).

4.3.3 Source: Speech prosody

Our third information source for the SI task is formed by prosodic attributes of the spoken user input. We incorporate prosodic features in the feature vector since those have been reported to function well for problem detection purposes (cf. Sections 2.3.3 and 2.3.4). From the digital audio recordings of the OVIS system we automatically extracted a number of measurements using the GIPOS software package. GIPOS generates and manipulates waveforms, spectrograms, and other forms of speech data [Vogten and Gigi 1998]. The processing yielded the following measurements.

The pitch of the user's voice is measured in terms of F0, i.e., fundamental frequency. The method used to determine F0 is Hermes' method of sub-harmonic summation (cf. [Hermes 1988]), combined with dynamic programming to smooth the F0 contour and remove any possible pitch measuring errors. Six features are computed that characterise pitch: the value of the maximum and the minimum F0 in the turn, the mean F0 and its standard deviation, as well as the position of the maximum and minimum F0 in the time line of the signal.

Loudness is measured in terms of RMS, i.e., the root mean square amplitude of the acoustic wave signal. Four features are extracted concerning RMS: the value of the maximum RMS in the turn (the minimum RMS is always zero), the time position of the maximum RMS, as well as the mean RMS and its standard deviation. The duration of the utterance is defined in seconds, and is automatically measured from initial silence to final silence in the recording of the user input.

From the word graph we furthermore extract the duration of the initial pause, again on the basis of the most confidently recognised string. The initial pause is measured in frames as the length of the silence that precedes the beginning of the speech signal. This feature may cue the degree of hesitation of the user in responding, cf. [Krahmer et al. 2001a].

From the word graph yet another prosodic feature, speech tempo, is computed. The speech tempo of the turn corresponds to the number of uttered syllables per second. The memory-based syllabifier tool of [Busser 1998] is used to automatically compute the number of syllables in the most confidently recognised string (recall that this may often only partially overlap with the actual utterance). The ratio of the number of syllables and the duration of the turn (in seconds) yields the tempo feature. Naturally, other simple (automatic) methods can also be used for approximating the number of syllables in a string of words, for example counting the number of vowel strings in it (combined with rules specifying vowel combinations in the given language).

Note that most of our features represent the data in a noisy way, since they are derived

on basis of potentially corrupted measurements. For example, in calculating tempo both the basis of the computation (i.e., the noisy most confidently recognised word string), and the tool that computed the amount of syllables (i.e., the imperfect syllabifier) add to the discrepancy between the original speech rate and our tempo feature. We hope however that the computed features represent the main tendencies in the focus turn's context, supplying sufficient information for robust extraction of SI components. Our choice in the current study is not to perform active selection of features but simply gather a large, assumed-to-be-comprehensive feature set. This is motivated by findings of e.g. [Batliner et al. 1999] who employ a large number of features in a prosodic processing task and find that the effort needed for determining an optimal feature set does not pay off in classification performance.

Note that the OVIS audio material is incomplete for a number of dialogue turns caused by technical conditions during the collection process: 108 turns are missing from the recordings, meaning that for these turns we have no prosodic feature values. Since the employed learning algorithms are able to handle missing values, we mark the prosodic feature values of these turns by a question mark in the feature vector.

The current study aims to gain insight into the relative importance of prosody in SI-related analysis of user turns in SDSs. Therefore, below we provide more insight into our data through descriptive statistics of the gained prosodic attributes.

4.3.3.1 Descriptive statistics

Our goal in this section is to find out whether prosodic tendencies observed in other, primarily American English systems (cf. Sections 2.3.3 and 2.3.4), also hold for our data. Therefore, we computed a series of basic statistics over the prosodic features measured over all user utterances, distinguishing between the problematic and non-problematic instances according to the backward-pointing-problem labelling. We performed an independent samples test on these pairs of means to check whether the differences between them are of statistical significance. Note that since we do not have information about user identity in the corpus, this analysis is carried out by collapsing all prosodic measurements into a pool, entailing that the findings reflect general prosodic tendencies across speakers, which might not hold for individual speakers. This distinction seems to be important, since [Hirschberg et al. 2004] come to the conclusion that "relative differences in speakers' prosodic values, not deviation from some 'acceptable' range, distinguishes recognition failures from successful recognitions".

Our tests reveal that, in line with observations reported in the literature, problematic turns differ significantly in important aspects from unproblematic turns. Table 4.10 highlights a selection of the most significant outcomes. Averaged over the complete data set the figures indicate that turns that feature awareness of communication problems tend to be significantly longer, have higher pitch maxima and pitch means than unproblematic user turns. Aware utterances furthermore exhibit significantly more energy (i.e., are louder) and a faster speech tempo than utterances signalling unproblematic grounding.

It is worth noting that in a certain sense this bird's-eye-view on the data is misleading. A closer look reveals that the scale of difference between the prosodic means is highly dependent on the kind of system prompt to which the given user responds. To examine

Feature	Difference in means	Signif.
F0 Max (Hz)	10.9	*
F0 Mean (Hz)	6.7	**
RMS Max	95.8	
RMS Mean	16.4	*
Ipause (frames)	0.3	
Duration (s)	0.4	**
Tempo (syll/s)	0.2	**

Table 4.10: Statistical comparison of prosodic means in the backward-pointing task (i.e., mean of turns showing backward-pointing problems minus mean of turns showing no backward-pointing problems). ‘*’ denotes outcomes of an independent samples test with $p < .05$ significance; ‘**’ denotes $p < .01$ significance.

the prosodic behaviour of users in reply to certain system prompts, we create prompt super-categories. The Explicit verification group (E) covers all prompts with a label ‘E_’, irrespective of what slot types are being confirmed by the system. Likewise, Implicit verification (I) groups all prompts in which the second part of the label contains ‘;I_’, irrespective of what slot types are being asked for or being confirmed by the system. The prompt type Open question (O) stands for system questions asking for slot information, without verifying other slots (‘Q_’). Yes/no question (Y) stands for all prompts that require a confirmation from the user; such prompts are typically given by the system at the end of the interaction (e.g., ‘Do you want to know another connection?’, Q_OC). We group the prosodic characteristics of user turns according to the prompt super-categories. We refer to this grouping of (properties of) user turns as ‘informed data splitting’. The information used to split the data is the prompt super-categories, to which we refer as prompt types.

Table 4.11 highlights the comparison of prosodic properties on the data split according to the four most frequent prompt types. The table compares the differences in the means for problematic minus unproblematic turns according to the most recently asked system prompt in the backward-pointing dimension. Figures for the remaining prompt types (meta-prompt, repeated prompts) are not included in the table; they occur less frequently and produce more unreliable outcomes. Typically, other F0 and RMS measurements correlate with certain prompt types as well, for example a high F0 maximum often is accompanied by a high F0 mean measurement.

If we compare the value differences of the means for problematic turns according to the most recently given system prompt, we find that some values deviate strongly from the overall average (displayed in Table 4.10). The differences in means vary across prompt types: the independent samples test reveals that the scales of the differences between means of problematic and unproblematic turns depend on the system prompt given in the most recent system turn. An obvious example is duration (see the corresponding row of Table 4.11). The difference in the utterance duration of aware of problems/unaware of problems turns is generally smaller after an implicit verification prompt than after an explicit verification prompt, or after a yes/no question. This means that aware and

Feature	Difference in means			
	Data split according to last system prompt type			
	E (474)	I (966)	O (591)	Y (665)
F0 Max (<i>Hz</i>)	26.7 **	6.4	36.2 **	52.0 *
F0 Mean (<i>Hz</i>)	9.1 *	4.4	13.7 *	33.9 *
RMS Max	1202.9 *	-213.6	-700.9	179.3
RMS Mean	113.3 **	7.8	-57.6 *	48.8
Ipause (<i>frames</i>)	-1.3	-0.9	5.4 **	1.7
Duration (<i>s</i>)	0.9 **	0.5 **	-0.2	0.8 **
Tempo (<i>syll/s</i>)	0.5 **	2.3	-0.5 **	0.6 *

Table 4.11: Statistical comparison of differences in prosodic means in the backward-pointing dimension per prompt types. Statistics for split data according to last system prompt type are shown for four system prompt types, the number of cases covered given between brackets. ‘*’ denotes outcomes of an independent samples test with $p < .05$ significance; ‘**’ denotes $p < .01$ significance.

unaware turns are of a more similar duration after I than after e.g. E or Y. Observing the RMS values we find similar subtleties. For instance, a user’s answer following an implicit verification of misunderstood information does not tend to be spoken significantly louder (since the subtraction produces a negative value), as one would expect in consequence of hyperarticulation (found in [Hirschberg et al. 2000, Oviatt et al. 1998]). Judged by the outcomes of the independent samples test, characteristics of some of the prosodic attributes are in accordance with findings concerning hyperarticulate speech, but others are clearly not, when distinguishing according to the actual prompt type.

What follows from these findings is that the type of system prompt may be relevant for detecting (backward-pointing) problems, and perhaps for the other components of SI as well. The fact that statistically significant differences exist between means does not entail that such differences are useful for the automatic detection of (components of) the SI label of user turns. Moreover, the statistical findings on turn type co-occurrences presented in Section 4.3.1 showed that, when the SI label is composed of four components, the most recent prompt can be followed by many types of (low frequency) user turns, and it is doubtful whether the correlations are strong enough to be utilised in ML. In Chapter 5 we will investigate in more detail to what extent these prosodic features contribute to components of SI when employed in a machine-learned classification task. Additionally, in [Lendvai et al. 2002a] we present statistics with respect to the forward-pointing problem dimension as well, and describe the potential use of informed data splitting for ML-based problem detection purposes. We now present the experimental outcomes on the SI task using all extracted features.

4.4 Results

4.4.1 Baseline

One possible baseline strategy for predicting the four-layered class label is to always predict the majority class tag. The most frequent label among the 3,738 user utterances in the corpus is `N_VOID_OK_OK` (the user gives a negative answer but it does not signal a problem, and it is correctly processed, e.g., ‘No, thank you.’). This label occurs 399 times in the corpus, and the strategy of always predicting this label yields 10.7% accuracy. This simple majority-class baseline is very low, and since it has no recall on the majority of other class labels, it is not informative for evaluation.

Given that certain types of user input are much more likely to follow certain types of system prompts (cf. Section 4.3.1), a better baseline, directly computable from the data, is to predict user input classes on the basis of the most recently asked system prompt. Always guessing the class occurring most frequently in response to the last system prompt type (averaged over the 90% training sets, in the same 10-fold partitions as used by the learners) produces a baseline of 41.9% accuracy. This simple strategy provides us with a sharp baseline, which we consider more relevant in assessing the performance of our learners than the majority-class baseline. We call this informed baseline as ‘prompt baseline’.

The detailed scores, including precision, recall, and F-score on the four components of the task, are given in the top section of Table 4.12. The prompt baseline reaches a 78.7 F-score on predicting the task-related act performed in the user turn, an F-score of 77.8 on predicting the types of filled slots, an F-score of 55.3 on the detection of forward-pointing communication problems, and an F-score of 81.3 on the detection of backward-pointing communication problems. The diversity among the classification results per partition is characterised by the standard deviation figures.

4.4.2 Performance on the complex SI task

In two series of experiments we train MBL and RI on all features to classify user turns in terms of the four components of SI. Below we describe the results obtained by the two learners, and compare those to the prompt baseline results, and to each other. Table 4.12 displays the performance of the two learners on the shallow interpretation task. Note that the overall accuracy score is not regarded as the most informative evaluative metric about the performance of the learners on the SI task; it is a complex measurement that is computed by proportionally weighting correct vs incorrect classifications of the individual SI components. Furthermore, as explained in Section 3.2, accuracy can sometimes in itself be uninformative about the actual performance on the task. Since the overall accuracy measurement is nonetheless often indicated in research, for practical reasons we also report on this score, besides the more informative precision, recall, and F-score.

Looking at overall accuracy, MBL attains 49.3%, and RI 45.5%. The difference between the accuracy of the prompt baseline and that of MBL is statistically significant in a paired t -test ($t = 7.3$, $p < 0.01$), likewise, the difference between the accuracy of the prompt baseline and that of RI is statistically significant ($t = 5.0$, $p < 0.01$). The optimised MBL

algorithm outperforms the optimised RI algorithm in classification accuracy, and their difference is statistically significant in a paired t -test ($t = 3.8, p < 0.01$).

If we look at the detailed sub-measures (accuracy, precision, recall, and F-score) indicated per component for each classifier, we see that in general MBL performs better than the baseline learner, whereas RI performs worse, or the same, as the baseline. MBL improves over the baseline and over RI by a broad margin in all but one aspect: on the forward-pointing problem component: 55 points of F-score is the maximum that can be attained on this subtask for all three learners, no matter the classification strategy.

The component for which both MBL and RI achieve the highest F-score is the task-related act label (89 and 80.9, respectively). Backward-pointing problems and filled slot types are classified with a comparable score (87.7 and 83.4 F-score, respectively). RI clearly performs poorer than MBL, and its performance is less stable, which is reflected by the standard deviation figures of the 10-fold CV. This suggests that the rule induction strategy tends to produce rule sets that cover unseen data less effectively than the strategy of extrapolating the class from examples that are nearest neighbours.

4.4.3 Parameter and feature use in MBL

Parameter optimisation led to a variety of settings for MBL: the optimised k is at least five or higher. Jeffrey divergence is the similarity metric picked most often (in seven partitions). The optimal feature weighting metric turns out to be shared variance (on the same seven partitions as Jeffrey divergence).

We took a general look at the weights MBL associated with the features in order to see which features were regarded informative by the learner. Grouping the higher-weighted features according to their source, we find that features that receive high weights include the following (lexical items are translated to English):

- DM prompt history: current prompt type, previous prompt type
- DM lexical, current prompt: ‘from’, ‘to’; various words present in the opening prompt
- DM lexical, previous prompt: ‘from’, ‘to’, ‘you’, ‘want’
- ASR confidence: topconf, topconfpernode, topconfdiff
- ASR lexical, BOW (recognised user words in current turn): ‘from’, ‘to’, ‘no’, ‘yes’, ‘o’clock’
- ASR lexical, prevBOW: ‘from’, ‘to’, #pause#
- ASR lexical: topconfstring.

Shared variance weighting (used together with Jeffrey divergence similarity metric) regards the lexical features most informative. IG-weighting (with MVDM similarity metric, employed once) associates the highest weights to the ASR confidence measures and to the DM prompt history features. Chi-square-based weighting (with overlap similarity metric, employed twice) regards the confidence measures most informative.

Algorithm	Component	Metric			
		acc	pre	rec	F
prompt baseline	ALL	41.9			
		1.6			
	TRA	74.8	81.7	76.0	78.7
		2.5	3.2	2.7	2.9
	SLOT	73.8	87.8	69.9	77.8
		2.6	2.0	3.2	2.2
	FWD PR	64.8	61.3	50.6	55.3
		2.4	3.8	3.3	2.7
	BWD PR	86.2	96.2	70.7	81.3
		2.3	1.9	5.7	3.9
MBL	ALL	49.3			
		2.8			
	TRA	84.1	92.4	85.9	89.0
		2.4	2.2	2.1	2.0
	SLOT	79.1	88.4	79.0	83.4
		2.8	2.5	4.0	3.1
	FWD PR	66.7	65.4	48.3	55.4
		2.5	3.9	4.5	3.8
	BWD PR	90.2	94.0	82.3	87.8
		2.5	3.5	4.4	3.8
RI	ALL	45.5			
		2.4			
	TRA	74.7	82.2	79.8	80.9
		4.9	4.7	6.3	5.4
	SLOT	73.3	82.7	70.7	75.7
		3.0	7.9	6.4	3.6
	FWD PR	66.2	64.1	50.6	55.6
		2.5	4.0	11.8	7.0
	BWD PR	83.6	89.6	70.8	78.6
		2.6	4.7	8.1	3.9

Table 4.12: Scores with standard deviation produced by MBL and RI on complex shallow interpretation, averaged over 10-fold CV experiments: overall accuracy, as well as accuracy (acc), precision (pre), recall (rec), and F-score (F) on task-related acts, filled slots, forward-pointing and backward-pointing communication problems. The prompt baseline performance is provided for comparison.

In order to point out problematic cases for the learner, we examined the classified material, and found that most classification errors were made on the forward-pointing component. In particular, the label `S_VA_PROB_OK` is frequently misclassified as `S_VA_OK_OK` (an incorrect prediction of the forward-pointing problem), whereas label `S_VA_OK_OK` is frequently misclassified as `S_VA_PROB_OK` (an incorrect prediction of the forward-pointing non-problem). Similar misclassifications on the forward-pointing component include the labels `N_VOID-<...>`, `S_D-<...>`. Likewise, `A;Y_VOID`-type of labels are often classified as `Y_VOID` labels, as well as `A;S_D-<...>`-type labels are often classified as `S_D-<...>` labels, suggesting that acceptance in the user input is difficult to point out.

4.4.4 Parameter and feature use in RI

For RI, parameter estimation yielded very varied algorithm settings, from which few clear tendencies are observable. In general, covering a minimal number of 1-5 instances per rule is found to be beneficial for RI. This might indicate that it is optimal for RI to make specific rules that cover only a few examples. The rule sets induced during training are large, consisting of 54-340 rules (223 rules on average). The low performance of the algorithm indicates however that part of the induced rules are locally optimal but too specific to generalise to unseen (i.e., test) data.

For illustrative purposes we trained RI on the total data, where parameter search resulted in the following algorithm settings:

- amount of learning instances to be minimally covered by each rule: 2 (default)
- hypothesis simplification: 1, i.e., leave hypothesis as it is (default: simplify less)
- negative tests on the nominal feature attributes allowed (default: disallowed)
- number of optimisation rounds on the induced rule set: 2 (default)
- class ordering: by decreasing frequency (default: order by increasing frequency)
- loss ratio of false-positives/false negatives: irrelevant, since loss ratio is only supported for two-class problems.

RI induced 207 rules from the total data. In general, we see that the algorithm made use of all kinds of features provided to it. Most use is made of the ASR and DM lexical features (i.e., the BOWs of both the user input and the system prompt), as well as of the most recent prompt type, the topconf, and the topconfpernode features. Many of the rules have multiple conditions, on average there are 2-3 conditions in a rule. Rules are induced for 73 classes, the number of rules induced per class is 1-13; on average three rules are induced per class. Rules are first induced for the most frequent class (`N_VOID_OK_OK`).

Below we present seven rules from this rule set in order to show the type of rules on basis of which RI obtains the reported results. Note that the rules are meant to illustrate the way RI operates, not the interpretation of user input. We select rules that cover a relatively large number of examples. The structure of a rule is

If *<feature test>* **and** *<feature test>* (etc.) **then** *class*. (*n/m*)

where $\langle \text{feature test} \rangle$ is a test on the presence of a nominal feature value, the presence of an element of a set feature, or a range of a numeric feature. n indicates the number of instances a rule covers, m the number of false predictions. Lexical items are translated from Dutch into English. Feature names are the following: *prompt t*: current prompt type; *prompt t-1*: previous prompt type; *prompt t-2*: previous-previous prompt type (and so on); other feature names are as introduced in Table 4.4.

- 1 **If** ‘connection’ \in sysBOW \wedge ‘sorry’ \notin sysBOW \wedge ‘no’ \in BOW **then** (380/108)
N_VOID_OK_OK.
- 2 **If** ‘so’ \in sysBOW \wedge ‘at’ \in sysBOW \wedge ‘no’ \notin BOW \wedge topconf \leq 701.06 **then** (125/32)
Y_VOID_OK_OK.
- 3 **If** ‘whether’ \in sysBOW \wedge topconf \leq 690.24 \wedge F0minpos \geq 1 **then** (34/5)
Y_VOID_OK_PROB.
- 4 **If** ‘which’ \in sysBOW \wedge tempo \geq 1.95312 **then** S_VA_OK_OK. (65/10)
- 5 **If** prompt $t = \text{Q_DTH;LVA}$ \wedge tcpernode \geq 137.13 \wedge prevBF \leq 1 \wedge rmsmaxpos \leq 0.42 **then** S_D_OK_OK. (22/2)
- 6 **If** ‘day’ \in sysBOW \wedge a7 = {empty} \wedge BF \leq 2 **then** A;S_D_OK_PROB. (8/6)
- 7 **If** ‘time’ \in sysBOW \wedge ‘six’ \in sysBOW \wedge F0max \leq 210 \wedge BF \leq 21 **then** (4/0)
S_THLPROB_OK.

The first rule assigns N_VOID_OK_OK to turns where the user BOW contains ‘no’, and the system said ‘connection’ but did not say ‘sorry’. The second rule assigns Y_VOID_OK_OK to turns in which the system says ‘so’ and ‘at’ (which is probably an explicit verification of the time slot), and the user answer is recognised with a smaller top confidence score than 701.06, whereas ‘no’ is not recognised in the input. The third rule classifies input as an affirmative answer that reflects awareness of problems (Y_VOID_OK_PROB) in case the system asks a question including ‘whether’ (probably as part of the phrase ‘could you repeat whether ...’) and the user input is recognised with a lower top confidence score than 690.24 whereas the pitch minimum is reached after at least 1 second.

Rule 4 fires in case the system prompt included the word ‘which’ and the tempo of the user input is faster than 1.95312 syllables per second; such input is classified as unproblematic slot-filling of departure and arrival station. This rule represents earlier findings claiming that problematic turns have a slower speech rate than unproblematic ones. Rule 5 shows that if the most recently asked system prompt is Q_DTH;LVA (asking for travel time, and implicitly verifying departure and arrival station) and the normalised top confidence score in the ASR output is higher than 137.13, the loudest part of the input occurs earlier than 42 seconds of the input, and the branching factor in the word graph of the previous user input is smaller than 1 (i.e., there is no branching in the graph), then the user is filling the day slot, no communication problems have occurred, and the user input is going to be well processed. Rule 6 indicates that in case the input took place not later than at the sixth exchange of the interaction (since the seventh item in the prompt history is empty), and to the system prompt which contains the word ‘day’ the user’s reply is recognised with a smaller branching factor than 2, then in that turn the user accepts a system error, as well as provides the value for the day slot.

The seventh rule classifies slot-filling of time of day and hour of travel in an unproblematic dialogue situation, and this input is going to be erroneously processed: the system prompt contains the word ‘time’ and ‘six’ (the latter obviously a verified slot value), whereas the user’s answer exhibits a pitch maximum not higher than 210 Hz and branching factor smaller than or equal to 21. This rule illustrates that there are many factors determining the class of user input: acoustic and prosodic features, probably contributing most to the problem components of SI (for example, the branching factor and pitch may be good cues to erroneous processing and dialogue feedback), whereas lexical properties may determine the task-related act(s) and the slot(s) in the input.

4.4.5 Detailed analysis of task-related acts and information units

Tables 4.13 and 4.14 show the classification results decomposed by TRA labels and slot labels, respectively. We see that the TRA label that is overall classified best is slot-filling (s). The classification results of other TRA labels are roughly in line with the relative frequency of the given label: *y*, the third most common TRA label in the corpus, is classified with a good score by both algorithms (90 F-score by MBL and 85 by RI). Both algorithms outperform the baseline by a broad margin on this label. The label *n* is classified with a similar score by MBL (86.1 F-score), but with a much lower result by RI (73.4 F-score), that is not better than the baseline strategy. As was to be expected, acceptance is difficult to classify, not only due to its low frequency but also because this act closely resembles truly slot-filling or truly affirmative TRAs.

Looking at the results decomposed by slot labels (Table 4.14), we can observe a similar tendency: labels that occur with high frequency in the data (e.g. VOID, *v*, *a*, cf. Table 4.2) are better learnt than low frequency classes. Learner performance on the non-slot label VOID is calculated in the sub-component evaluations shown in the table; however, performance on classifying VOID is ignored in the calculation of the global precision, recall, and F-score of the slot component of SI (displayed in Table 4.12), since VOID is not some slot to be filled, but rather signals the absence of the slot-filling activity.

Note that there are extremely large standard deviation figures for a number of TRA or slot types, such as NSTD, *t*, or acceptance. This is due to the way in which scores are summarised over the ten data folds: whenever no numerical score is obtained in a fold (for example because some slot type is not present in some fold, resulting in illegal arithmetic operations), we assume the score to be zero, so that means can be still calculated. Observe that the NSTD TRA type can likewise not be calculated by the baseline strategy. In fact, this suggests that giving a non-standard input is not a TRA that depends on the type of the most recent system prompt.

4.4.6 Discussion

According to Table 4.12, the component for which MBL achieves the highest F-score is the task-related act label (89.0 F-score), and the result obtained on backward-pointing problems (87.7 F-score) is quite similar to this. The classification results of filled slot types are of somewhat lower scale (83.4 F-score). The component that is learnt most successfully by RI is, likewise, the task-related act label (80.9 F-score), and similarly to

Algorithm	TRA label	Metric		
		pre	rec	F
prompt baseline	S	95.7	87.3	91.3
		0.7	2.5	1.4
	Y	68.7	64.6	66.3
		5.6	8.1	5.5
	N	64.0	74.6	68.8
		6.6	4.7	4.8
	A	40.3	18.1	24.5
		19.7	7.7	10.5
	NSTD	-	-	-
		-	-	-
MBL	S	96.7	93.9	95.3
		0.9	1.8	0.9
	Y	86.6	93.1	89.7
		6.2	5.4	5.0
	N	92.4	80.8	86.1
		3.8	4.9	3.3
	A	64.6	15.5	22.4
		32.9	7.1	9.3
	NSTD	55.1	46.1	46.7
		22.1	22.6	16.8
RI	S	96.7	82.1	88.2
		1.4	12.4	6.9
	Y	83.2	89.4	85.1
		17.2	3.6	10.8
	N	67.1	85.1	73.4
		16.1	6.0	8.1
	A	37.1	17.8	23.6
		21.3	11.8	14.5
	NSTD	77.5	43.6	52.3
		23.8	22.9	22.4

Table 4.13: MBL and RI performance decomposed by **TRA labels** on complex shallow interpretation, averaged over 10-fold CV experiments in terms of precision, recall, and F-score. The top section shows scores with standard deviation according to the prompt baseline, the middle section shows results of MBL, the bottom section shows results of RI.

Algorithm	Slot label	Metric		
		pre	rec	F
prompt baseline	V	94.9	87.8	91.1
		2.2	5.1	2.7
	A	92.9	77.0	84.1
		3.8	5.6	3.8
	D	74.1	77.7	75.7
		4.8	6.4	4.1
	T	15.0	0.7	1.3
		32.0	1.4	2.6
	H	79.8	49.5	60.7
		5.4	8.4	6.7
	@	20.0	1.7	3.2
		40.0	3.5	6.4
MBL	V	80.8	93.0	86.4
		3.0	1.6	1.9
	A	94.2	92.5	93.3
		2.7	4.2	2.8
	D	93.3	88.2	90.6
		3.6	4.8	3.8
	T	77.7	78.0	77.5
		4.6	8.7	4.9
	H	60.4	13.3	20.0
		28.7	8.8	11.6
	@	83.1	67.3	74.0
		5.4	7.3	4.3
RI	V	84.8	40.5	49.2
		16.0	12.4	19.4
	A	89.8	94.3	92.0
		2.7	1.9	1.6
	D	94.3	84.7	89.1
		2.2	4.8	2.6
	T	91.2	81.5	86.0
		2.2	5.3	3.4
	H	76.2	57.7	63.6
		11.2	14.9	7.8
	@	60.3	33.3	39.7
		21.6	14.7	12.1

Table 4.14: MBL and RI performance decomposed by **slot labels** on complex shallow interpretation.

the results of MBL, the backward-pointing problem (78.6 F-score) is classified second-best. Filled slot types are classified with somewhat less success (75.7 F-score). Both algorithms attain the lowest score on the forward-pointing problem component, which is equal to the prompt baseline strategy (55.3 F-score).

Prediction of TRAs, slots, and backward-pointing problems is done better by MBL than by the baseline strategy and by the rule-induction learner. This suggests that MBL is able to learn the complex SI class more optimally from the data than RI. This finding seems to be in line with the conclusions of [Rotaru and Litman 2003] who claim that, depending on a number of factors, MBL and RI can outperform each other (cf. Section 3.1). However, note that in our experimental matrix MBL is systematically better than RI. Due to their classification method, rule induction algorithms generally perform worse when the instance space is complex, and when there are no homogeneous class or feature subsets on which the data can be efficiently partitioned. In our material there are 148 classes in the data, and no class has real majority. In addition, these classes are represented by a large number of features, and possible patterns of the feature values may not be optimally captured by rule conditions.

The results of parameter estimation support this observation: while MBL is able to cover the data by a more homogenous set of parameters, RI applies much data-specific settings to each data subset that probably do not generalise to unseen data; the fact that the classification accuracy of the rules induced with these settings is quite low may indeed indicate that these are over-fitted on the training material. As opposed to it, the fact that a larger nearest neighbourhood size turns out to be more optimal for MBL reflects that the memory-based learner is able to effectively flatten the instance space by comparing a large number of nearest neighbours that vote for the class according to their distance.

The prompt baseline strategy yields relatively high scores because there appear to be strong correlations between system prompts and typical user answers that follow it. This is not surprising; the hard part of the task is to predict those cases where the user gives a different response than what is most likely. MBL is able to find similarity between memory examples and new instances with a 49% overall accuracy, meaning that this algorithm can classify almost half of the user turns perfectly in terms of task-related acts, slots, problem awareness, and problem origin. This at the same time provides further evidence that despite the observed statistical co-occurrences between system prompts and user answers, it is not obvious to predict the type of answer most probably triggered by a prompt, for example that yes/no type system prompts trigger Y and N task-related acts.

It is an important finding that both classifiers use lexical information, presented as an unsophisticated bag-of-words, both from the system and the user turns to a large extent. This may indicate that not only continuous (i.e., numerical), but also symbolic (i.e., lexical) items are able to separate the data well in terms of class labels. This is obviously related to the nature of the SI task as defined by us, since some aspects of some SI labels, in particular the task-related act labels Y and N, may be strongly lexical-oriented. It is however a non-trivial knowledge, gained from our experiments, that the shallow bag-of-words representation encodes information that can be well utilised by the learners, as opposed to information encoded by features that are more structured, such as the prompt history features. Namely, we found that the system prompt representations are utilised to a smaller extent in classifying the complex class than their corresponding

BOW representations, despite the fact that they quite systematically encode traditional, hierarchical notions of dialogue acts. This suggests that a fine-grained representation of pragmatic-semantic information is not necessarily more informative for interpreting user input in task-oriented dialogues with a SDS than other, less structured pragmatic-semantic primitives.

4.5 Summary

In this chapter we have described our research material, the OVIS corpus, and presented results on machine learning of the complex SI class of user turns in this corpus. We discussed the task design of these experiments by explaining the class labelling scheme applied to the corpus, and by giving an account on the features employed in the learning experiments. We explained in detail that the class labelling incorporates all four components of the information we want to extract from the user turns: the conducted task-related act (TRA), the slots that are being filled (SLOT), whether the turn is going to cause communication problems between the user and the dialogue system (FWD PR), and whether the user shows awareness of communication problems (BWD PR).

We emphasised that the employed features serve the purpose of providing context for the focus turn in the learning experiments at a low level and in a straightforward way. We presented some statistical findings with respect to the system's prompts and prosodic properties of the users' replies to these prompts. We also investigated whether there are co-occurrence tendencies between a particular prompt and a particular user reply (both described in terms of our annotation scheme).

Finally, we presented the outcomes of a series of learning experiments conducted with the memory-based and the rule induction classifiers that were optimised with respect to their parameters. We found that MBL attained higher performance in general, when compared to either RI or an informed baseline that assigns the majority class given the most recent prompt. MBL produced 49.3% overall learning accuracy on the complex SI task, whereas RI gained a significantly lower overall classification accuracy, 45.5%. Learning performance differed substantially for the four included components. The task-related act of a user turn was learnt best: MBL reached an 89-point F-score on this component (RI: 80.9). Backward-pointing problems were identified with a similar score by both classifiers. Slots were predicted with a somewhat lower precision and recall (MBL: 83.4 F-score, RI: 75.7). Neither MBL nor RI could significantly outperform the baseline for forward-pointing communication problems (55.3).

An important finding is that the statistical tendencies of prosodic, as well as prompt co-occurrence phenomena are not reflected in the actual classification results. The analysis of the learning process furthermore indicates that structured information (i.e., prompt history represented in terms of dialogue acts and slots) is not necessary more informative for the SI task than unstructured information (i.e., prompt history represented in terms of a bag-of-words).

Analysis of the classification results led us to the conclusion that the complex SI task is difficult, since some (aspects of) components make it hard for the learners to correctly assign the class of a user input. For example, the acceptance TRA cannot be reliably

detected by the algorithms, possibly because it closely resembles the slot-filling TRA. Most importantly, it is very hard to predict whether some input is going to be correctly or erroneously processed by the given SDS. The fact that the prompt baseline best identifies forward-pointing problems might indicate that this fairly unpredictable component is at least partly dependent on the system's prompting.

Given these difficult aspects of shallow interpretation, our next goal is to find more effective ways for inferring the SI of user turns from our data. In the next chapter we investigate whether it is possible to improve these scores if the data are presented in a different way to the classifiers.

Chapter 5

Partitioning Information

The goal of this chapter is to investigate two issues raised by the findings of the previous chapter: the influence of class label design and that of feature design on learner performance in the SI module. The approach we take is to partition information in the data provided to the learners. The outline of the chapter is the following. For each component we conduct two consecutive series of experiments with both MBL and RI. In the first series we perform class partitioning, in the second feature partitioning. We present and analyse the obtained results per SI component.

The class and the feature partitioning experiments provide a possibility to compare memory-based learning and rule induction to a considerable extent: all experiments are conducted by both MBL and RI under identical conditions, while the tasks, as well as the features, are systematically varied. By these experiments we attempt to further investigate whether, as observed in [Rotaru and Litman 2003], it is dependent on the task, the number of features, and the type of features whether memory-based learning or rule induction performs better.

In the discussion we recapitulate the results of the information partitioning experiments conducted in this chapter by pointing out the major findings with respect to task and feature design in the SI task.

5.1 Method

The results of Chapter 4 indicate that some SI components are more difficult to classify than others, which might have caused low overall scores. In fact, in the complex task design described in the previous chapter we took a naive approach to SI using all features to classify all components simultaneously; the question arises whether our data can be used in a more optimal way. To this end we first design experiments in which the classifiers are trained to learn each SI component in all possible combinations of components. The class label is partitioned so that components are learnt either in isolation or in combination with one or two other SI components. We refer to the phenomenon of learning components together as co-learning. Our aim is to see whether a different class label design

yields improvement for learning a particular component over the results presented in the previous chapter. Note that the experiment conducted in that chapter investigated the most complex co-learning task, since all four components were simultaneously co-learned; we refer to that experiment as the complex experiment.

5.1.1 Class partitioning

In the first experimental series we test all component combinations for each SI component, running seven learning experiments per component: on the component in isolation, as well as in combination with one or two other components. The experiments are carried out with the same set-up as the complex experiment: we train and test the classifiers in 10-fold cross-validation combined with algorithm parameter optimisation. The obtained results are characterised in terms of accuracy, precision, recall, and F-score, computed in relation to the given component. For comparative purposes both the prompt baseline scores and the results of the complex task are provided in the tables that display the results of the class partitioning experiments. All statistical significance tests are reported on the basis of paired *t*-tests.

Note that each component combination has a different class label distribution: in the experiments where a component is learnt in isolation, the number of classes is the lowest in that component's series, whereas in all other experiments in that series the number of classes is higher. The number of class labels in an experiment defines the entropy in the classification task.

If the class labels are uniformly distributed, the entropy (see Equation 3.7) of a task is maximal. This situation almost occurs in some of our tasks (namely, in the forward-pointing and in the backward-pointing isolated tasks, as well as in the combination of these two components), however, in most tasks the distribution of the labels in a task is skewed to some degree (cf. the simple statistics given in Section 4.2).

At the same time, it might be informative to calculate the ratio of the actual entropy, given the number of actual classes, to the maximum entropy. This ratio, which we call actual entropy ratio, AER, defined in Equation 5.1, characterises how much uncertainty a class distribution contains compared to the maximally uncertain, uniform-distribution entropy:

$$AER = \frac{H(actC)}{H(uniC)}, \quad (5.1)$$

where $H(actC)$ is the actually measured entropy, based on the actual frequencies of classes *actC*, and $H(uniC)$ is based on the same number of classes but with uniform frequencies. The AER figure characterises the reduction in uncertainty with respect to the maximum uncertainty in the classification task, given the actual number of class labels in the task. Table 5.1 displays the figures of the actual class label number and the AER per component combination.

We may also assume independence between the labels, hypothesising that each class label of each component co-occurs with each class label of each other component, and

Component(s)	Actual nr of classes	AER	IER
TRA	8	.66	.66
SLOT	30	.57	.57
FWD PR	2	.99	.99
BWD PR	2	.99	.99
TRA_SLOT	63	.63	.47
TRA_FWD PR	16	.72	.72
TRA_BWD PR	15	.72	.69
SLOT_FWD PR	48	.66	.62
SLOT_BWD PR	47	.66	.62
FWD PR_BWD PR	4	.98	.98
TRA_SLOT_FWD PR	104	.69	.52
TRA_SLOT_BWD PR	90	.67	.51
TRA_FWD PR_BWD PR	29	.76	.68
SLOT_FWD PR_BWD PR	81	.48	.44
TRA_SLOT_FWD PR_BWD PR	148	.74	.54

Table 5.1: Class label number and entropy ratios per class label combination, for details see equations 5.1 and 5.2.

also that these combinations are uniformly distributed. Such uniform distribution and such independence of components' class labels is in our study obviously not the case, since e.g. a slot-filling TRA never co-occurs with a VOID slot label. However, to analyse the reduction in entropy with respect to this 'independent entropy', we define the independent entropy ratio measure (IER). IER characterises the ratio of the actual entropy and such a component-independent maximum entropy:

$$IER = \frac{H(actC)}{H(indepC)} \quad (5.2)$$

where $H(actC)$ is the measured entropy in the task given the actual frequency of classes $actC$, and $H(indepC)$ is the ultimate maximum entropy in the task, given uniformly distributed classes of all possible class label combinations, $indepC$. The rightmost column of Table 5.1 displays the IER per component combination. Generally, the IER values characterise not only the ratio of reduction of uncertainty in the task, but the predictability of dependency between the components in a given combination as well.

From the table we can establish that most class partitioning tasks are much less entropic than would be expected under independence and uniform distribution assumptions, since most of the AER and IER figures are smaller than 1. It is an intriguing issue whether these figures are predictive about how the classifiers perform on some task. Notably, one conjecture might be that low AER or IER would predict a high degree of success in learning.

5.1.2 Feature partitioning

The findings of the previous chapter indicated that the classifiers made use of all kinds of features provided to them. At the same time, as noted earlier, there is redundant representation of the same information by some of the features we employ. In order to see the extent to which the different information sources contribute to the SI task, we design separate experiments in which the classifiers draw on information from only one feature group at a time: on features that are obtained from the dialogue manager (henceforth: DM feature group), on features that are obtained from the word graph output of the speech recogniser (henceforth: ASR feature group), or on the prosodic features measured in the recorded user input (henceforth: PROS feature group).

After finding out the optimal class label design for a SI component in the class partitioning experiments, in a second series of experiments we analyse the contribution of each feature group to the classification of that component. The task here is to measure performance on the three isolated feature groups in classifying the optimal component combination obtained from class partitioning. Here we are primarily interested in the relative importance of the isolated feature groups, hence we do not test all combinations of feature groups.

It is interesting to note that the three feature groups differ in the way they encode information. The DM group contains exclusively symbolic features: prompt types and prompt words, the PROS group contains exclusively numeric features, whereas the ASR group contains a mix of both (cf. Table 4.4). The number of features contained by each group is different, too. There are 944 features in the DM group, 13 features in the PROS group, and 1,525 features in the ASR group.

The set-up of the feature partitioning experiments is to reuse in 10-fold CV the optimal parameter settings obtained in the class partitioning experiments. This can be regarded as the semi-optimisation of algorithm parameter settings.

5.2 Task-related acts

5.2.1 Class partitioning

The first component we investigate concerns the task-related acts in the user input. The experimental results of co-learning the different class combinations for the TRA, as well as of the isolated experiment, are shown in Table 5.2.

BEST SCORES We see that the scores of MBL are overall somewhat higher than those of RI; however, most of these differences are statistically insignificant. MBL attains the highest score when the TRA component is learnt in isolation (91.7 F-score), whereas RI obtains its highest F-score of 90.5 points in co-learning the TRA and the backward-pointing problem; the difference between the two learners is statistically insignificant. It is remarkable that RI produces a near-10-point improvement over its complex experiment's F-score ($t = 4.9$, $p < 0.01$; the italicised bottom line of the table), indicating that class partitioning is beneficial for this classifier for this component. The best F-score of MBL also improves significantly over the F-score of the complex experiment ($t = 5.0$, $p < 0.01$). In terms of

Algorithm	Class label	Metric			
		acc	pre	rec	F
baseline		74.8	81.7	76.0	78.7
		2.5	3.2	2.7	2.9
MBL	TRA	86.6	94.3	89.3	91.7
		0.7	1.5	1.0	0.7
	TRA_SLOT	86.4	93.5	89.3	91.3
		1.2	1.2	0.8	0.6
	TRA_FWD PR	86.3	94.4	88.3	91.2
		0.8	1.3	1.0	0.9
	TRA_BWD PR	86.8	94.1	89.3	91.6
		1.0	1.3	1.2	0.9
	TRA_SLOT_FWD PR	82.3	93.7	86.9	90.1
		0.5	1.3	1.0	0.7
	TRA_SLOT_BWD PR	85.6	92.8	88.7	90.7
		1.4	1.7	1.3	1.2
	TRA_FWD PR_BWD PR	85.9	94.2	87.8	90.9
		1.2	1.6	1.5	1.3
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>84.1</i>	<i>92.4</i>	<i>85.9</i>	<i>89.0</i>
	<i>2.4</i>	<i>2.2</i>	<i>2.1</i>	<i>2.0</i>	
RI	TRA	84.8	91.9	87.8	89.8
		1.4	2.5	1.2	1.5
	TRA_SLOT	84.7	91.5	88.8	90.1
		1.9	2.1	1.7	1.6
	TRA_FWD PR	83.9	91.3	86.9	89.0
		2.2	3.4	2.0	2.2
	TRA_BWD PR	86.0	92.0	89.1	90.5
		1.7	2.0	1.5	1.5
	TRA_SLOT_FWD PR	78.5	86.5	83.1	84.7
		3.5	3.5	3.6	3.1
	TRA_SLOT_BWD PR	82.1	89.1	86.7	87.9
		2.9	3.2	1.3	1.9
	TRA_FWD PR_BWD PR	84.8	91.2	87.9	89.8
		1.9	2.4	1.4	1.7
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>74.7</i>	<i>82.2</i>	<i>79.8</i>	<i>80.9</i>
	<i>4.9</i>	<i>4.7</i>	<i>6.3</i>	<i>5.4</i>	

Table 5.2: Scores with standard deviation produced by MBL and RI on shallow interpretation of the **TASK-RELATED ACT** component, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of task-related act type. The highest score is set in boldface. The italicised bottom lines show the results of the complex experiment. Scores of the prompt baseline are provided in the top row.

F-score, RI achieves a large, 50% error reduction on the TRA task compared with the complex experiment, whereas MBL produces a 24% error reduction. (In terms of accuracy, RI produces a 40% error reduction, and MBL a 10% error reduction.)

EFFECT OF COMPONENT TYPES AND NUMBER OF CLASS LABELS The outcomes of statistical significance tests show that the F-scores among the class partitioning experiments do not differ significantly from each other when the task is learnt in isolation or when not more than two components are combined (corresponding to the scores in the first four lines of each section in the table), even if one of the components is the difficult forward-pointing problem. The entropy ratios (ERs) of these tasks (cf. Table 5.1) show no correspondence with the attained scores, likewise, the actual number of class labels in an experiment does not seem to have an impact on the learning performance either. Consider for example that the class combination TRA_FWD PR_BWD PR has fewer classes (29) than the class combination TRA_SLOT (63), but the former is still learnt with a significantly lower score than the latter.

However, we observe that when more than two classes are co-learnt, no matter whether the forward-pointing problem is included, scores get significantly worse for both classifiers (again in no apparent correlation with the number of classes or ERs). It is furthermore noteworthy that both classifiers improve significantly with respect to their complex experimental results in all but one class combination: in the co-learning of TRAs, slots, and forward-pointing problems, where both MBL and RI attain a statistically insignificantly higher score than in the complex experiment. Again no apparent deviations can be read out from the entropy figures that would indicate such a performance.

CONCLUSION On the basis of the outcomes of the class partitioning experiments concerning the TRA component we can establish that class partitioning has a substantial, positive influence on the scores produced by our classifiers, and it results in practically equal performances of MBL and RI. We assume that it is the optimisation of class label combination together with the optimisation of algorithm parameters that leads to this result, providing further evidence for the conclusions made by [Daelemans and Hoste 2002] (cf. Section 3.1).

Another remarkable outcome of the experimental matrix is that co-learning the task-related act component with the backward-pointing problem component has a positive effect on both learners: RI obtains its best score on this combination, and the difference between the score attained on this combination and the best score by MBL is insignificant. This suggests that combinations between these two components show patterning in our data that the learners are able to utilise to the same extent. As we pointed out in Section 4.2, signalling awareness of communication problems can be regarded as a backchannelling act; it might be the consistent occurrence of certain labels of the awareness component with certain labels of the TRA component that lead to this result.

5.2.2 Feature partitioning

Below we report on the experiments in which the classifiers are trained on partitioned feature groups. In these experiments MBL is trained to classify the task-related act label in

Algorithm	Optimal class label	Feature group	Metric			
			acc	pre	rec	F
MBL	<i>TRA</i>	<i>ALL</i>	<i>86.6</i>	<i>94.3</i>	<i>89.3</i>	<i>91.7</i>
			<i>0.7</i>	<i>1.5</i>	<i>1.0</i>	<i>0.7</i>
		DM	77.5	84.5	79.1	81.7
			2.6	3.5	2.4	2.8
		ASR	82.8	90.8	85.4	88.0
			1.4	1.8	1.5	1.3
		PROS	64.5	73.8	68.0	70.8
			3.0	3.7	2.0	2.7
		<i>ALL</i>	<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
			<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>
RI	<i>TRA_BWD PR</i>	<i>ALL</i>	<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
			<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>
		DM	76.8	84.4	77.6	80.9
			2.4	3.0	2.5	2.7
		ASR	82.3	89.6	86.0	87.7
			1.9	2.4	2.1	1.7
		PROS	60.5	69.5	64.9	67.1
			3.2	4.1	3.3	3.2
		<i>ALL</i>	<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
			<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>

Table 5.3: Performance of the three feature groups in experiments by MBL and RI on the **TASK-RELATED ACT** component with optimised class labelling. The highest score is set in boldface. The italicised top rows in the sections show the scores of learning on all features.

isolation, whereas RI is trained to classify the task-related act combined with the backward-pointing problem component. The employed algorithm settings are the ones optimised on all features. Table 5.3 presents the outcomes of feature partitioning. For better displaying the scores obtained using all features are also reproduced in this table, corresponding to the top row of each section.

We see that the trends are similar for both learners across these scores. An important outcome is that none of the isolated information sources is able to produce the same or better classification results than when all features are used; this implies a number of findings. It suggests that our approach requires no explicit feature selection (which is usually computationally expensive), working equally well when all available information is presented to the learners. We also find evidence that none of the information sources is eligible in itself to produce the best results on classifying TRAs. The ASR group attains the highest scores, suggesting that information encoded by the speech recogniser’s output is most important for learning the task-related act of the user input. The ASR group’s F-score is 3.7 points lower for MBL and 2.8 points lower for RI than that on all features. These differences are significant (MBL: $t=9.4$, $p<0.01$, RI: $t=3.3$, $p<0.01$).

In Section 2.3.1 we noted that lexical and (micro-)syntactic cues are widely used for the automatic detection of dialogue acts in speech. We hypothesised that the ASR’s recognition lattice is capable of encoding and providing part of this information in a

shallow way that does not require possibly expensive computation of those cues. Since the ASR group is used with almost the same success for interpreting simple task-related acts as all the features together, we might assume that our results support this hypothesis (recall that most of our task-related acts correspond to aspects of traditional dialogue acts, cf. Sections 2.2.1 and 4.2.1).

Prosodic information contributes least to detecting the task-related act of the input, whereas the impact of the information coming from the dialogue manager is between those of the ASR and the PROS groups. Note that the features in the DM group do not encode the user's utterance at all; it is surprising that these features are more predictive of the task-related acts in the user input than prosodic attributes of the user utterance. Even if prosody contributes least to the identification of TRAs, the scores attained by this feature group are in line with those of previous studies of [Jurafsky et al. 1996, Reithinger et al. 1996, Samuel et al. 1998b, Stolcke et al. 1998a, Shriberg et al. 2001] are able to utilise prosodic information for detecting (more fine-grained) dialogue acts to a roughly similar extent.

5.2.3 Detailed analysis

In order to gain more insight into the extent to which the different TRA types are classified, in Table 5.4 we display the scores calculated for each TRA type. The figures in the table are obtained in the highest-scoring class partitioning experiment for both MBL (TRA) and RI (TRA_BWD PR). Compared to those of the complex experiment (Table 4.13), we observe that in general the scores improve. Note that for identifying a slot-filling act both learners, but especially RI, benefit substantially from optimising the class label: MBL improves from a 95.3 F-score (given in the first line of the middle section in Table 4.13) to 97.3 ($t=7.5$, $p<0.01$), and RI from a 88.2 F-score (given in the first line of the bottom section in Table 4.13) to 96.4 ($t=3.5$, $p<0.01$). This means that our SI module — optimised with respect to algorithm settings and class labels — can detect rather extensively whether the user is supplying slot-filling information in the input.

We observe that both learners classify affirmative answers (Y) to a significantly better extent than negative answers (N) (MBL: $t=5.1$, $p<0.01$, RI: $t=2.4$, $p<0.05$), which was also the case in the complex experiment, signalling that in our data it is easier to detect affirmative input than negative input. Note that the lower F-score of N is the result of a much lower recall on this class than on Y, indicating that the algorithms often fail to guess N. The high precision scores show at the same time that retrieving N is done rather accurately. This tendency requires further investigation since it is likely to have serious consequences for the detection and correction of errors that emerge during interacting with the given SDS.

The only TRA type for by MBL for which class partitioning yields a lower result than in the complex experiment is acceptance. Acceptance turns out to be the hardest TRA phenomenon to learn anyway, in line with the findings of the previous chapter. This is likely to be due to the arbitrariness of whether a user is inclined to ignore system error; furthermore, the sparseness of A in the data causes highly divergent scores per fold (cf. the large standard deviations).

For classifying non-standard user input (NSTD) the opposite holds: MBL benefits to a

Algorithm	TRA label	Metric		
		pre	rec	F
MBL	S	97.3	97.4	97.3
		0.7	1.4	0.6
	Y	95.1	92.9	93.9
		3.6	4.3	3.5
	N	97.7	85.3	88.3
		4.2	2.5	2.4
	A	40.7	14.7	20.4
		23.0	9.3	11.7
	NSTD	76.9	70.6	70.7
		17.3	25.3	19.6
RI	S	95.6	97.2	96.4
		1.6	0.9	0.9
	Y	89.4	91.5	90.4
		5.0	2.4	3.0
	N	90.1	86.5	88.2
		3.4	4.1	2.8
	A	43.2	21.3	27.3
		19.5	11.3	12.8
	NSTD	77.7	44.9	54.8
		30.8	23.1	23.2

Table 5.4: MBL and RI performance on interpreting **TASK-RELATED ACT TYPES**, averaged over 10-fold CV experiment in terms of precision, recall, and F-score. The scores are obtained with the most optimal class label composition: MBL: TRA, RI: TRA_BWD PR.

large extent from class partitioning, since the F-score of learning this TRA type improves from 46.7 to 70.7. It is an intriguing issue what exactly effectuates the decrease of A and the increase of NSTD in class partitioning. Below we list and discuss the most interesting rules induced by RI on these two classes. Note that the class label includes two components since the optimal class co-learned by RI is TRA_BWD PR. Naturally, the rules induced on our data probably do not generalise, and, as emphasised earlier, the rule set is presented to supply information about the internal mechanisms, especially the use of feature types, in RI, thus not for the purpose of describing general human-machine interaction types.

- 1 **If** 'so' \in sysBOW \wedge 'and' \in sysBOW \wedge topconf ≤ 559.76 **then** A;Y_PROB. (20/15)
- 2 **If** 'yes' \in BOW \wedge RMSstdev ≥ 1040 \wedge 'and' \in sysBOW **then** A;Y_PROB. (7/2)
- 3 **If** 'so' \in sysBOW \wedge 'between' \in sysBOW \wedge F0max ≤ 223 \wedge 'correct' \in BOW **then** A;Y_PROB. (6/1)
- 4 **If** 'to_PP' \in prev sysBOW \wedge 'to_PP' \in sysBOW \wedge 'o'clock' \in sysBOW \wedge topconfnnode ≥ 119.247 **then** A;S_PROB. (24/10)
- 5 **If** 'from' \in prev sysBOW \wedge 'to_PP' \in prev sysBOW \wedge 'o'clock' \in sysBOW \wedge RMSmean ≥ 266 \wedge F0min ≥ 75 **then** A;S_PROB. (12/1)

The first rule set displays rules induced for classes containing acceptance. We see that the rules cover a relatively small number of examples with quite a few counter-examples. Conditions are made on particular values of lexical and prosodic context, making the rules highly situation-specific. In cases when acceptance co-occurs with affirmative input (rules 1-3), lexical conditions are made on items present in the current user BOW such as ‘yes’ and ‘correct’, which are (not necessarily) present in the recognition hypothesis, on items present in the current system BOW (‘so’; *dus*, always cuing an explicit verification prompt), and, interestingly, on prosodic features (loudness, pitch). The latter may indicate that it is possible to pinpoint particular prosodic values in an utterance containing acceptance.

In cases when acceptance co-occurs with slot-filling (rules 4-5), conditions are made on items present in both the current and the previous system BOW (e.g., indicated by ‘to_PP’, i.e., the preposition ‘to’; *naar*), indicating repeated system prompts, as well as on prosodic and confidence-related features of the input (loudness, normalised highest confidence score).

The second rule set displays selected rules induced on the NSTD task-related act type. We see that users give non-standard input both when they react to a previous problem (class label: NSTD_PROB, rules 3-4), and when communication is unproblematic (class label: NSTD_OK, rules 1-2). Examples corresponding to the latter case can be characterised by no pauses in the input (cf. rule 1 – the input is probably empty, since all non-empty word graphs contain pauses), as well as by interaction-specific conditions such as the particular station slot value ‘ groningen ’ in rule 2 (probably because supplying a station name at that point of the dialogue was not appropriate).

- 1 **If** ‘#pause#’ \notin BOW **then** NSTD_OK. (8/1)
- 2 **If** $\text{RMSmaxpos} \leq 0.13 \wedge \text{‘ groningen ’} \notin \text{prev BOW}$ **then** NSTD_OK. (7/5)
- 3 **If** $\text{dur} \leq 1.28$ **then** NSTD_PROB. (27/16)
- 4 **If** $\text{dur} \leq 1.28 \wedge \text{F0min} \leq 74 \wedge \text{prompt } t \neq \text{Q_DTH;L_VA} \wedge \text{prompt } t \neq \text{E_H}$ **then** NSTD_PROB. (17/5)

RI learns from the data that non-standard answers in reaction to system error may consist of a very short (or, possibly, empty) input, lasting for 1.28 seconds or less (rule 3), or of a short answer in combination with a particular pitch height and dialogue situation, e.g., where the most recent prompt is neither asking for travel time while implicitly verifying departure and arrival stations, nor an explicit verification of hour (rule 4). The rules again cover a relatively low number of examples in the data, and have relatively many counter-examples.

5.3 Information units

5.3.1 Class partitioning

BEST SCORES The experimental results of class partitioning for the slot component are shown in Table 5.5. The most important outcome of this matrix of experiments is that the best scores attained by the two classifiers do not significantly differ from each other.

Algorithm	Class label	Metric			
		acc	pre	rec	F
baseline		73.8	87.8	69.9	77.8
		2.6	2.0	3.2	2.2
MBL	SLOT	82.3	89.7	83.9	86.7
		2.5	2.2	3.1	2.0
	TRA_SLOT	83.5	90.7	84.9	87.7
		2.2	1.8	2.6	2.0
	SLOT_FWD PR	81.5	90.5	81.2	85.5
		2.0	0.9	3.8	2.3
	SLOT_BWD PR	82.1	90.1	83.2	86.5
		1.8	1.2	2.8	1.8
	TRA_SLOT_FWD PR	80.9	90.0	80.6	85.0
		1.6	1.1	3.0	1.8
	TRA_SLOT_BWD PR	82.1	89.9	83.5	86.6
		2.4	1.8	3.6	2.3
	SLOT_FWD PR_BWD PR	79.2	88.5	79.0	83.4
		2.1	1.8	4.0	2.8
	TRA_SLOT_FWD PR_BWD PR	79.1	88.4	79.0	83.4
		2.8	2.5	4.0	3.1
RI	SLOT	82.6	88.4	82.9	85.5
		2.4	4.4	3.8	2.8
	TRA_SLOT	80.7	80.9	83.2	82.0
		2.6	4.0	2.7	3.2
	SLOT_FWD PR	77.9	85.8	77.1	80.9
		1.7	7.0	3.4	2.6
	SLOT_BWD PR	80.9	88.1	80.7	84.2
		1.5	3.1	2.4	1.8
	TRA_SLOT_FWD PR	76.8	81.5	77.8	79.3
		2.3	7.7	4.0	3.0
	TRA_SLOT_BWD PR	78.7	82.5	80.0	81.1
		2.7	5.1	3.2	3.3
	SLOT_FWD PR_BWD PR	74.9	80.3	76.6	77.7
		2.3	7.6	7.3	2.3
	TRA_SLOT_FWD PR_BWD PR	73.3	82.7	70.7	75.7
		3.0	7.9	6.4	3.6

Table 5.5: Scores with standard deviation produced by MBL and RI on shallow interpretation of the **SLOT** component, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of slot type. The highest score is set in boldface. The italicised bottom lines show the results of the complex experiment. Scores of the prompt baseline are provided in the top row.

MBL learns the slot component best when combined with the TRA component, yielding a 87.7 F-score. The improvement of MBL over the F-score of the complex experiment is statistically significant ($t=5.2$, $p<0.01$). It seems intuitive that co-learning the task-related act type helps in classifying the filled slots (but not necessarily the other way round, see the outcomes in the previous section), since these two components license each other. However, RI classifies the slot component best in isolation (85.5 F-score), again improving considerably over its complex experimental score. RI's best score is also significantly better than its second-best score (co-learning of slots and backward-pointing problems). In terms of F-score, MBL produces a 26% error reduction, whereas RI achieves a 40% error reduction on the SLOT task compared to the complex experiment.

EFFECT OF COMPONENT TYPES AND NUMBER OF CLASS LABELS It can be observed that MBL produces no statistically different scores between co-learning three or four components in case one of the components is the forward-pointing problem (cf. the experiments on TRA_SLOT_FWD PR, SLOT_FWD PR_BWD PR, and the complex learning), while ER figures are at a variance in these experiments. Only the one not including the FWD PR is capable of outperforming the complex experiment. Note that this cannot be inferred from the AER or IER of this task, since the ERs are not particularly lower than those of the three-component experiments. In experiments with RI the scores produced by co-learning three (or four) components in which one of the components is the forward-pointing problem produce some significant differences between each other, but none of these outperform the prompt baseline.

The fact that MBL learns the slot component best in combination is interesting since in the combined experiment there are more than twice as many class labels (63) as in the experiment where slots are classified in isolation (30), and also because the ERs of the better experiment are higher. The difference in F-score between the scores of these two experiments is significant ($t=2.9$, $p<0.05$). Another remarkable result is that RI's best score (on the isolated component) does not differ significantly from the result on co-learning the slot and the backward-pointing problem component (84.2 F-score), which again shows higher entropy ratios. This may indicate that the presence or absence of the act of signalling awareness of communication problems might contribute to detecting the kind of slots the user is filling; an explanation for this can be that when users become aware of problems, they often do not fill the demanded slots (i.e., the classifier has to predict VOID), and constellations of VOID and BWD PR may appear in distinctive patterns in the data.

For the slot component it is not the case that learning more than two components aggravates the scores, considering e.g. that the result of SLOT_BWD PR of MBL is practically the same as that of TRA_SLOT_BWD PR, or that the result of TRA_SLOT of RI is practically the same as that of TRA_SLOT_BWD PR, and so on. In general, the experiments do not show a clear trend about how class partitioning determines which class combination yields the best scores. The number of co-learned components, the number of class labels or entropy in the task, and the presence of the forward-pointing problem component seem to affect performance in an intertwined way.

Algorithm	Optimal class label	Feature group	Metric			
			acc	pre	rec	F
MBL	<i>TRA_SLOT</i>	<i>ALL</i>	83.5	90.7	84.9	87.7
			2.2	1.8	2.6	2.0
		DM	71.0	78.6	74.0	76.1
			3.6	4.5	3.1	2.7
		ASR	78.4	86.2	79.6	82.7
			2.1	1.9	2.2	1.6
		PROS	51.1	48.1	49.7	48.9
			3.3	3.4	3.9	3.5
RI	<i>SLOT</i>	<i>ALL</i>	82.6	88.4	82.9	85.5
			2.4	4.4	3.8	2.8
		DM	71.0	87.8	65.4	74.6
			4.8	2.9	8.4	5.6
		ASR	76.8	85.9	77.2	81.2
			2.4	4.4	3.9	3.2
		PROS	51.4	59.9	38.9	45.0
			5.0	9.1	11.9	8.2

Table 5.6: Performance of the three feature groups in experiments by MBL and RI on the **SLOT** component with optimised class labelling. The highest score is set in boldface. The italicised top rows in the sections show the scores of learning on all features.

CONCLUSION The outcomes of the class partitioning experiments on learning the slot component support our preliminary findings that class partitioning has a substantial, positive influence on the scores attained by our classifiers, resulting in eliminating performance differences between MBL and RI, since the best scores obtained by the two classifiers show practically identical performance. Again, the scores attained by RI in this experimental matrix are overall somewhat lower than those of MBL.

Another remarkable outcome of the experiments, also in line with the findings on classifying TRAs, is that a task’s entropy does not seem to determine performance; the lowest ERs in the matrix are associated with experiments that are not the ones attaining the best score by any of the classifiers. At the same time, the ERs assigned to two-component experiments are roughly the same, but there are significant differences in the results of these experiments.

5.3.2 Feature partitioning

Table 5.6 presents the outcomes of the feature partitioning experiments conducted for classifying slots. Remarkably, the same trends can be observed here as in the feature partitioning experiments for task-related acts. In particular, both learners attain lower scores when some of the feature groups are removed from the experiment. Again the ASR group produces the highest scores, coming closest to the result of the experiment utilising

all features (marked in *italics*), but the difference between the F-score of these experiments remains statistically significant (MBL: $t = 5.5$, $p < 0.01$, RI: $t = 4.2$, $p < 0.01$). This means that information encoded by the speech recogniser's output is most useful for learning the slot type treated in the user input. The ASR group is the one containing most features, which might imply that it contains the most or best information as well. However, we believe that its informativity is much more related to its content than to its quantity: the large number of features in this group is due to a lot of redundancy, as well as to the binarisation of the features (cf. Section 4.3.2), whereas e.g. [Rotaru and Litman 2003] find that performance improves in case the number of relevant features is increased.

Information encoded by the DM group contributes less to detecting what slot(s) the user is filling in a turn, which is somewhat surprising given the correlations described in Section 4.3.1. As in the case of task-related acts, in these experiments prosody contributes least to the detection of slots in the user input. We believe that the interaction between features in different feature groups is very important for learning the slot component, for which we also found evidence in the rule sets presented above, where conditions are made on features from all three sources.

5.3.3 Detailed analysis

To provide details about the extent to which different slot types are classified by MBL and RI, we display these scores in Table 5.7.

Compared to the complex experiment (Table 4.14), a general observation is that all scores improve through class partitioning. Both classifiers produce the largest improvement on the day, time of day, hour, and arrival value slot types (D, T, H, @, respectively). For example, classification of the D slot improves from 77.5 to 81.3 F-score for MBL, and from 63.6 to 77.7 F-score for RI; that of H from 74.0 (MBL), respectively 58.1 (RI) to 85.2, respectively 80.0.

Below we present rules with the largest coverages and relatively small number of counter-examples induced by RI on these labels.

- 1 **If** 'when' \in sysBOW \wedge *teststringlength* $\leq 4 \wedge$ *tempo* $\geq 0.977199 \wedge$ 'o'clock' \notin BOW **then** D. (171/23)
- 2 **If** 'when' \in sysBOW \wedge *topconfpernode* $\geq 136.334 \wedge$ 'to_PP' \notin BOW **then** D. (137/5)
- 3 **If** 'when' \in sysBOW \wedge *topconf* ≤ 886.4 **then** D. (186/29)
- 4 **If** 'time' \in sysBOW \wedge 'o'clock' \in BOW **then** H. (139/16)
- 5 **If** 'arrive' \in BOW \wedge 'to_PP' \notin BOW \wedge *dur* ≥ 2.81 **then** H@. (38/6)
- 6 **If** prompt $t = \text{Q_DTH;LVA} \wedge$ 'afternoon' \in BOW **then** DT. (6/0)
- 7 **If** 'when' \in sysBOW \wedge 'o'clock' \in BOW \wedge 'tomorrow morning' \in BOW **then** DTH. (12/3)

User turns are classified as filling the D slot if the system prompt contains 'when' and the most confident lattice path of the user reply consists of four or less words, none of them being 'o'clock' (which is an indicator of H rather than D), uttered with a speech rate of 0.97 syllables per second or more (rule 1); the latter indicates that slot-filling is often performed by a short (probably elliptical) sentence. Likewise, the system prompt

Algorithm	Slot label	Metric		
		pre	rec	F
MBL	V	95.3	94.2	94.7
		1.6	2.9	1.7
	A	93.0	92.5	92.7
		2.6	3.2	2.6
	D	86.2	77.3	81.3
		4.6	5.7	4.0
	T	68.1	31.6	42.1
		12.0	10.1	10.8
	H	86.0	84.5	85.2
		3.9	4.6	4.0
	@	92.4	66.1	75.9
		8.4	13.7	9.7
	VOID	92.8	94.9	93.8
		1.1	1.5	0.7
RI	V	94.1	89.6	91.7
		3.1	5.2	2.5
	A	93.4	88.8	91.0
		2.8	3.5	2.6
	D	83.5	8.1	77.7
		8.1	5.0	4.5
	T	72.7	60.6	64.2
		11.7	18.1	14.2
	H	80.6	80.0	80.0
		5.6	7.8	5.0
	@	77.3	74.6	73.3
		16.5	18.7	11.4
	VOID	87.7	94.8	91.0
		5.5	2.9	2.7

Table 5.7: MBL and RI performance on interpreting **SLOT TYPES**, averaged over 10-fold CV experiment in terms of precision, recall, and F-score. The scores are obtained with the most optimal class label composition: MBL: TRA_SLOT, RI: SLOT.

containing ‘when’, whereas the user reply not containing the preposition ‘to’, and the normalised top confidence being greater or equal to 136.33 refers to the D slot (rule 2), indicating that after a ‘when’ prompt users often (repeatedly) provide the destination station name, since this is the stage where they can first infer from the implicit verification (which is always present in a ‘when’ prompt) that the system misrecognised their previous input. Rule 3 illustrates classification of the D slot in case the system prompt contains ‘when’ and the highest confidence of the recognised user reply is maximally 886.4.

Rule 4 aims at classifying user turns as filling the H slot with the simplest approach: in case the system prompt contains the word ‘time’ and the user answer contains the word ‘o’clock’. Rule 5 is an example of classifying the arrival hour conditioning on the prosodic duration feature as well as on the presence of the word ‘arrive’ and the absence of the preposition ‘to’ in the answer’s recognition lattice, the latter excluding potential examples that would refer to arrival station. In six cases with no counter-examples rule 6 covers user input as filling day and time of day if the system asks for travel time while implicitly verifying departure and arrival stations, and the word graph of the user turn contains ‘afternoon’, i.e., a particular slot value, while no conditioning is made on potential references to the day slot. The last rule illustrates classifying the combined label DTH in case the system asked a ‘when’ prompt and the user answer contained both the words ‘tomorrow morning’ (i.e., *morgenchend*) and ‘o’clock’.

It is also interesting to see some of the rules that cover non-slot input, i.e., the slots marked with the VOID label. Below we show some of the general rules concerning this slot type.

- 1 **If** $\text{dur} \leq 1.79 \wedge \text{tempo} \leq 0.653595$ **then** VOID. (705/14)
- 2 **If** ‘connection’ $\in \text{sysBOW} \wedge \text{topconf} \leq 779.72$ **then** VOID. (251/3)
- 3 **If** ‘to_PP’ $\notin \text{sysBOW} \wedge$ ‘which’ $\notin \text{sysBOW} \wedge$ ‘that’ $\in \text{BOW} \wedge \text{topconf} \leq 692.91$ **then** VOID. (66/0)

Again we see that conditions are made on features from all three feature groups: rule 1 conditions on prosodic features, whereas rule 2 and 3 on features from the DM and the ASR group. It is remarkable that the absence of slot-filling activity is characterised sufficiently by the absence of words that would refer to slot-prompting (rule 3), indicating a yes/no question (e.g., about the connection, as in rule 2). The user answer of VOID input type is best characterised by its tempo (probably due to answers to yes/no questions being short), recognition confidences, and the pronoun ‘that’ (i.e., *dat*), frequently employed in user’s phrases such as ‘No, that is not necessary.’, respectively ‘Yes, that’s right.’ in answers to yes/no questions.

5.4 Forward-pointing problems

5.4.1 Class partitioning

The figures in Table 5.8 show the results obtained in classifying forward-pointing problems with class partitioning.

BEST SCORES A remarkable outcome of these experiments is that both MBL and RI produce the highest F-score by the same task design: co-learning of forward-pointing problems and task-related acts. RI is able to outperform MBL in this experiment, although not significantly. Remarkably, the best score attained by MBL is not enough to perform significantly better than the prompt baseline, whereas that of RI shows a significant improvement over it ($t = 4.0$, $p < 0.01$). The scores are overall low. Despite the high standard deviations, there is a significant improvement with respect to the complex experiment both by MBL ($t = 4.3$, $p < 0.01$) and RI ($t = 3.0$, $p < 0.05$), an error reduction of 9% by MBL and 16% by RI.

Even though the baseline F-score and the complex experiment's F-score are practically identical, there is improvement by MBL only on one of them. This is due to the nature of the paired t -test that penalises for inconsistent differences between partitions of experiments. The same applies for the case when RI has a statistically smaller improvement of over its complex experiment than MBL, despite RI producing a 7-point improvement as opposed to the 4-point improvement of MBL.

EFFECT OF COMPONENT TYPES AND NUMBER OF CLASS LABELS It is surprising that the scores the classifiers produce on the isolated task are lower than those in the co-learning experiments, even though the ERs are the highest for the isolated task. In particular, rule induction is generally supposed to perform better when it has to classify fewer classes but our empirical results do not comply with this expectation. Due to the high standard deviations (which at the same time indicate that classification is very much dependent on the data sets for the FWD PR component), the differences between the isolated and the best experiment are statistically insignificant.

Again no direct correlation can be observed between the entropy figures and learner performance. In particular, the experiments with the smallest ERs do not produce better results than those with higher ERs. We see no trend that would suggest that the number of components to learn influences good or bad performance. In general, other than for the best scores, there are no significant differences between the scores in this experimental matrix.

CONCLUSION The outcome of class partitioning on the forward-pointing problem component is in line with those of the previous two components, producing nearly identical top scores for MBL and RI. Both classifiers significantly improve with respect to the complex experiment, but only RI with respect to the baseline.

A remarkable finding of this matrix of experiments is that it is beneficial to learn forward-pointing problems together with task-related acts. Since this has also been found earlier (in Section 5.2.1), we assume that TRAs show a consistent patterning with the presence of forward-pointing problems. In Section 5.5 we investigate this issue in more detail.

5.4.2 Feature partitioning

Table 5.9 displays the results obtained on learning the forward-pointing problem based on isolated feature groups. The feature partitioning experiments on the forward-pointing

Algorithm	Class label	Metric			
		acc	pre	rec	F
baseline		64.8	61.3	50.6	55.3
		2.4	3.8	3.3	2.7
MBL	FWD PR	68.1	67.7	49.6	57.0
		2.6	4.9	7.0	5.5
	TRA_FWD PR	68.6	67.1	53.5	59.4
		2.6	4.4	5.4	4.1
	SLOT_FWD PR	67.3	65.4	51.7	57.5
		3.4	4.5	7.1	5.5
	FWD PR_BWD PR	67.6	65.6	52.4	58.0
		2.0	4.9	5.3	4.3
	TRA_SLOT_FWD PR	67.5	66.2	50.4	57.1
		2.8	4.2	4.9	4.3
	TRA_FWD PR_BWD PR	68.5	67.0	53.0	59.1
		2.6	4.1	4.7	4.0
	SLOT_FWD PR_BWD PR	65.7	63.4	48.1	54.6
		2.3	2.8	5.1	3.9
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>66.7</i>	<i>65.4</i>	<i>48.3</i>	<i>55.4</i>
		<i>2.5</i>	<i>3.9</i>	<i>4.5</i>	<i>3.8</i>
RI	FWD PR	64.8	63.9	54.9	54.8
		3.2	8.0	23.4	11.5
	TRA_FWD PR	65.6	59.8	68.5	62.6
		3.0	5.5	13.4	5.2
	SLOT_FWD PR	66.4	65.0	50.2	56.2
		2.9	5.3	7.0	3.7
	FWD PR_BWD PR	67.8	67.3	50.5	57.2
		2.1	5.3	7.6	4.8
	TRA_SLOT_FWD PR	66.4	64.3	50.3	55.7
		3.0	4.5	11.5	0.8
	TRA_FWD PR_BWD PR	66.3	63.2	55.2	57.8
		1.9	4.8	12.4	6.1
	SLOT_FWD PR_BWD PR	66.1	64.1	50.8	55.5
		1.4	4.4	9.9	6.1
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>66.2</i>	<i>64.1</i>	<i>50.6</i>	<i>55.6</i>
		<i>2.5</i>	<i>4.0</i>	<i>11.8</i>	<i>7.0</i>

Table 5.8: Scores with standard deviation produced by MBL and RI on shallow interpretation of the **FORWARD-POINTING PROBLEM** component, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of slot type. The highest score is set in boldface. The italicised bottom lines show the results of the complex experiment. Scores of the prompt baseline are provided in the top row.

Algorithm	Optimal class label	Feature group	Metric			
			acc	pre	rec	F
MBL	<i>TRA_FWD PR</i>	<i>ALL</i>	<i>68.6</i>	<i>67.1</i>	<i>53.5</i>	<i>59.4</i>
			<i>2.6</i>	<i>4.4</i>	<i>5.4</i>	<i>4.1</i>
		DM	66.8	62.5	57.4	59.8
			3.0	4.3	4.7	4.1
		ASR	64.5	61.1	48.2	53.6
			1.8	2.8	7.3	5.2
		PROS	58.2	51.5	53.8	52.5
			2.3	1.5	4.3	2.2
RI	<i>TRA_FWD PR</i>	<i>ALL</i>	<i>65.6</i>	<i>59.8</i>	<i>68.5</i>	<i>62.6</i>
			<i>3.0</i>	<i>5.5</i>	<i>13.4</i>	<i>5.2</i>
		DM	62.8	56.6	68.2	59.8
			3.5	6.4	18.8	9.7
		ASR	65.0	57.9	72.1	63.5
			2.0	3.9	12.2	4.7
		PROS	50.9	46.2	81.9	57.7
			4.7	3.7	22.0	8.4

Table 5.9: Performance of the three feature groups in experiments by MBL and RI on the **FORWARD-POINTING PROBLEM** component with optimised class labelling. The highest score is set in boldface. The italicised top rows in the sections show the scores of learning on all features.

problem component diverge at some points from the trend observed so far (i.e., that the ASR group contributes most to classification).

It turns out that MBL outperforms the experiment utilising all features when it uses the isolated DM group, although statistically this difference is not significant. Remarkably, although RI also attains the exact same score on the isolated DM group, this is not RI's best score on the forward-pointing problem component: RI utilises the ASR group to the highest extent in the feature partitioning experiments, since this group outperforms the experiment on all features (although not significantly).

This means that the two classifiers make use of the ASR group to a different extent in predicting FWD PR: MBL benefits less from the ASR group than RI does, attaining a lower score in that experiment than on all the features. Comparing the highest scores obtained by the two classifiers (i.e., the F-score on the isolated DM feature group for MBL and the F-score on the isolated DM feature group for RI), we again find that these scores are statistically indistinguishable.

It can be established that the improvements are the result of an increased recall with respect to the experiments with the total feature vector. Namely, the recall of MBL arises from 53.5% to 57.4%, and that of RI from 68.5% to 72.1%. At the same time, the precision scores drop. Looking at the classification logs we indeed find that in the experiments that use the isolated feature groups there is less misclassification of the PROB cases of the

FWD PR component, and more misclassification of the OK cases, leading to the modified precision and recall figures.

5.4.3 Detailed analysis

When looking at the induced rule sets using the DM group, we see that on all but one data set the class left as default in the rule set is S_PROB, indicating that it is best for RI's accuracy to leave this frequent class as majority class. On the remaining class labels that contain the PROB label (e.g., A;S_PROB, N;S_PROB, N_PROB) both the coverage and the precision of the rules are low. In order to examine the conditions more thoroughly, we run an experiment on the full data set with RI. Below we reproduce the induced rule set.

- | | | |
|---|--|-----------|
| 1 | If 'from' ∈ sysBOW ∧ prompt $t - 3 = \text{EMPTY}$ then S_PROB. | (493/300) |
| 2 | If 'to_PP' ∈ sysBOW ∧ prompt $t = \text{Q_VA}$ then S_PROB. | (53/45) |
| 3 | If 'to_PP' ∈ prev sysBOW ∧ 'again' ∈ sysBOW then S_OK. | (95/44) |
| 4 | If 'to_PP' ∈ prev sysBOW ∧ 'when' ∈ sysBOW ∧ 'the' ∈ sysBOW then S_OK. | (52/42) |
| 5 | If 'connection' ∈ sysBOW then N_OK. | (489/308) |
| 6 | If 'so' ∈ sysBOW ∧ 'and' ∉ sysBOW then N_OK. | (193/146) |
| 7 | If 'whether' ∈ sysBOW then Y_OK. | (44/40) |
| 8 | Else A;Y_PROB. | (3/1391) |

There are 8 rules in the obtained rule set. The default class is S_PROB. We see that only rule 1 and 2 contain PROB cases. It is clear that both rules 1 and 2 refer to the same dialogue context. Rule 1 predicts communication problems in case the system asks about the departure station, which typically happens in the very first system turn or in case the prompt is repeated. Note that the second condition, made on the dialogue history, also indicates this: the third prompt is empty, meaning that only two prompts have been given up to that point of the interaction. Rule 2 predicts communication problems in case the system asks about the destination station (first condition) and the departure station simultaneously (second condition). Although the precision of these two rules is not high, they reveal that the opening prompt is a major problem source in our dialogues. The finding that MBL utilises the isolated DM group optimally in classifying forward-pointing problems relates to the frequency-based observations in Section 4.3.1 that already suggested that some system prompts correlate strongly with forward-pointing problems. In accordance with the findings in [Lendvai and Maruster 2003], these results suggest that the prompt design of the OVIS system might be the cause of certain communication errors.

In order to see what information in the ASR features contributes to the relatively high classification scores, we examine the rule sets induced by RI on the isolated ASR group. The rules show most conditioning on the length of the most confident string: e.g., if this is relatively long (i.e., consists of nine or more words), or, if the top confidence score is above a certain value, then the input is predicted to cause a problem. Interestingly, a lot of conditions are made on the presence of the words 'no' and 'not', indicating that turns in which users signal a problem are often misrecognised again in the next turn.

Since so far we have looked at rules in which FWD PR is co-learned with TRA, it is also interesting to see what rules are induced when the forward-pointing problem component

is learnt in isolation. In the isolated experiment the induced rule sets are very small, consisting of 4-7 rules. The optimal parameter setting automatically found by parameter search in most data sets (8 partitions) is to induce rules for the majority class (i.e., OK, cf. Section 4.2.3), leaving the minority class PROB as the default class on which no rules are induced. In the remaining two partitions the following rules are induced for the PROB class:

Partition 2:

- 1 **If** 'to_PP' \in sysBOW \wedge BF $\geq 3 \wedge$ 'I' \in BOW \wedge 'again' \notin prev sysBOW **then** (173/35)
PROB.
- 2 **If** 'connection' \notin sysBOW \wedge BF $\geq 2 \wedge$ 'one' \notin prev sysBOW \wedge 'to_PP' \in (114/25)
BOW \wedge RMSmean ≤ 285 **then** PROB.
- 3 **If** tcstringlength $\geq 4 \wedge$ 'from' \in prev sysBOW \wedge 'again' \notin sysBOW \wedge tempo (64/16)
 ≤ 2.05761 **then** PROB.
- 4 **If** 'connection' \notin sysBOW \wedge 'from' \in sysBOW \wedge 'yet' \notin prev sysBOW \wedge (55/8)
'#pause#' \in prev BOW \wedge topconf ≤ 760.83 **then** PROB.
- 5 **If** topconf $\geq 775.47 \wedge$ BF $\geq 1 \wedge$ 'again' \notin sysBOW \wedge 'want' \in BOW \wedge 'from' (51/11)
 \in prev sysBOW \wedge RMSstdev ≤ 695 **then** PROB.
- 6 **Else** OK. (1805/989)

Partition 5:

- 1 **If** topconf $\geq 781.91 \wedge$ BF $\geq 2 \wedge$ prompt $t - 3 = \text{'#empty#'}$ \wedge RMSmax \leq (65/6)
5248 \wedge F0max $\leq 177 \wedge$ 'good morning' \notin prev sysBOW **then** PROB.
- 2 **If** 'to_PP' \in sysBOW \wedge BF $\geq 3 \wedge$ 'but' \notin sysBOW \wedge F0max $\geq 236 \wedge$ (88/10)
RMSmean ≤ 285 **then** PROB.
- 3 **If** tcstringlength $\geq 4 \wedge$ 'yet' \notin prev sysBOW \wedge topconfpernode $\leq 120.439 \wedge$ (44/5)
'from' \in prev sysBOW \wedge RMSstdev $\geq 564 \wedge$ F0max ≥ 245 **then** PROB.
- 4 **If** topconf $\geq 785.33 \wedge$ BF $\geq 2 \wedge$ 'me' \notin prev sysBOW \wedge 'to_PP' \in BOW **then** (235/110)
PROB.
- 5 **If** 'to_PP' \in sysBOW \wedge topconfpernode $\leq 137.984 \wedge$ 'but' \notin sysBOW \wedge 'where' (108/43)
 \in sysBOW **then** PROB.
- 6 **If** 'connection' \notin sysBOW \wedge tcpernodediff $\geq 0.008469 \wedge$ 'sorry' \notin prev sys- (61/15)
BOW \wedge tcpernodediff $\geq 0.132225 \wedge$ topconfpernode ≥ 115.57 **then** PROB.
- 7 **If** 'connection' \notin sysBOW \wedge 'yes' \in BOW \wedge 'but' \notin sysBOW \wedge 'o'clock' \in (155/145)
BOW **then** PROB.
- 8 **Else** OK. (1485/584)

We see that the rules refer to highly specific and complex dialogue situations, informing about a number of subtleties. For example, the first rule induced on Partition 2 describes the situation when the system asks the user to repeat the input concerning the destination station. The system asks this prompt for the first time (i.e., 'could you say again' is not in the previous system BOW). The branching factor of the user input is high, probably also contributing to incorrect processing of the user's reply.

It can be established from the rule sets that RI makes extensive use of features that are numeric (i.e., confidence-based features as well as prosodic features), probably because it can efficiently separate classes on the basis of numeric splits. Lexical elements are also widely used in the conditions, both from the system and the user BOW.

5.5 Backward-pointing problems

In the first chapter of our study we noted that in case the dialogue system assumes it is unable to correctly process the user input, it typically repeats its prompt. A repeated prompt thus always means that there is a communication problem between system and user and the system is aware of this. Naturally, our goal is to discover all other communication problems (as well), not only the ones that are signalled by the system anyway. We computed what the performance of a ‘system-knows’ strategy is in problem awareness. This baseline strategy is to always assume a backward-pointing problem (i.e., the user becomes aware of a communication problem) when the system repeats its previous prompt (972 times in the corpus). However, the scores of this baseline are lower than those of the prompt baseline: although the precision of this strategy is 100%, its recall is only 60.2%, yielding a 75.1 F-score and 82.8% accuracy. Although the system-knows baseline is interesting from the point of view of error reduction with respect to the given SDS, from the point of view of the improvement of classification performance of our learners it is more informative to compare those to the (higher) prompt baseline.

5.5.1 Class partitioning

The class partitioning experimental matrix on backward-pointing problems is displayed in Table 5.10.

BEST SCORES In the performance of MBL there is no significant difference between the best score produced in co-learning TRA_SLOT_BWD PR and the score of TRA_BWD PR. In the performance of RI there are two identical best scores produced in co-learning TRA_BWD PR and BWD PR in isolation. This is a unique result since in no other class partitioning experiment were two identically highest scores produced. Both learners improve over the complex experiment in their best scores significantly (MBL: $t = 2.7$, $p < 0.01$, RI: $t = 7.3$, $p < 0.01$), as well as over the baseline. With the best scores both learners obtain a substantial error reduction on the F-score of the complex experiment: the reduction is 24% for MBL, and 46% for RI.

The most noteworthy outcome of the class partitioning experiments on the backward-pointing problem component is that both classifiers produce scores that are highest, or insignificantly different from the highest, by co-learning the BWD PR with the TRA component. This is remarkable, since the situation was the same in the class partitioning experiments of the task-related act component (cf. Table 5.2).

EFFECT OF COMPONENT TYPES AND NUMBER OF CLASS LABELS The results of combinations of three components are somewhat lower for RI than the results of combinations of

Algorithm	Class label	Metric			
		acc	pre	rec	F
baseline		86.2	96.2	70.7	81.3
		2.3	1.9	5.7	3.9
MBL	BWD PR	89.9	95.0	80.7	87.2
		1.4	1.3	3.3	2.3
	TRA_BWD PR	91.5	94.3	85.4	89.7
		0.8	1.7	2.2	1.4
	SLOT_BWD PR	91.5	92.5	87.4	89.9
		1.2	2.3	2.2	1.5
	FWD PR_BWD PR	90.0	94.0	82.1	87.6
		1.1	2.5	3.9	1.8
	TRA_SLOT_BWD PR	92.3	93.7	88.1	90.8
		0.9	2.5	2.1	1.2
	TRA_FWD PR_BWD PR	90.4	95.2	81.8	87.9
		1.1	2.4	3.9	2.1
	SLOT_FWD PR_BWD PR	89.6	93.5	81.5	87.0
		1.6	2.9	4.5	2.7
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>90.2</i>	<i>94.0</i>	<i>82.3</i>	<i>87.8</i>
		<i>2.5</i>	<i>3.5</i>	<i>4.4</i>	<i>3.8</i>
RI	BWD PR	90.5	92.4	85.1	88.5
		1.4	3.5	3.9	1.5
	TRA_BWD PR	90.5	92.1	85.1	88.5
		1.1	1.5	2.6	1.6
	SLOT_BWD PR	88.9	91.0	82.6	86.4
		2.5	3.9	5.0	3.2
	FWD PR_BWD PR	89.2	94.8	79.4	86.3
		0.9	2.6	3.4	1.6
	TRA_SLOT_BWD PR	87.0	89.9	78.8	83.9
		2.0	3.9	3.4	2.5
	TRA_FWD PR_BWD PR	88.4	93.5	78.7	85.4
		1.2	1.5	2.7	1.7
	SLOT_FWD PR_BWD PR	85.6	88.2	77.8	82.2
		1.9	5.1	7.5	2.9
	<i>TRA_SLOT_FWD PR_BWD PR</i>	<i>83.6</i>	<i>89.6</i>	<i>70.8</i>	<i>78.6</i>
		<i>2.6</i>	<i>4.7</i>	<i>8.1</i>	<i>3.9</i>

Table 5.10: Scores with standard deviation produced by MBL and RI on shallow interpretation of the **BACKWARD-POINTING PROBLEM** component, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of slot type. The highest score is set in boldface. The italicised bottom lines show the results of the complex experiment. Scores of the prompt baseline are provided in the top row.

less components; however, this is not the case for MBL. The presence of the FWD PR component does not seem to influence the performance of RI (note that for example the F-score of the FWD PR_BWD PR experiment is only 0.1 point lower than that of the SLOT_BWD PR experiment, while the latter does not include the difficult FWD PR component), but seems to influence that of MBL (compare e.g. the same combinations).

The isolated experiment, in which ERs are almost maximal, leads to a relatively lower result for MBL (the F-score of this experiment is even lower than that of the complex one), however, RI produces (one of) its highest score on it; this outcome again signals the difference of working principle between our two classifiers, and illustrates well that an algorithm's bias can lead to skewed results on an unoptimised task.

CONCLUSION As in the case of the three other SI components, based on the class partitioning experiments on the backward-pointing problem component we can establish that class partitioning has a substantial, positive influence on our classifiers, resulting in practically identical performances of MBL and RI.

Again, co-learning the task-related act component with the backward-pointing problem component has a positive effect on both learners: RI obtains (one of) its best score on this combination, and the difference between the score attained on this combination and the best score by MBL is insignificant. These results point to the same trend that was observed in Section 5.2.1, suggesting that awareness of problems is useful to be regarded as a backchannelling act, since we have found some evidence that BWD PR is closer in its patterning to the task-related acts than to the isolated conceptual category 'problem awareness'.

5.5.2 Feature partitioning

The outcomes of the feature partitioning experiments are displayed in Table 5.11. Note that for RI we run experiments on both of the winning class combinations.

The results of the feature partitioning experiments indicate that, similarly to the performance of MBL on the FWD PR component, most information comes from the DM features when classifying BWD PR. It is noteworthy that scores of the learners on the DM group improve above the system-knows baseline by a large margin, indicating that the classifiers reduce a large part of the errors the system is not aware of by drawing on information from the dialogue manager itself. Both MBL and RI (in the isolated experiment) produce a 34% error reduction with respect to the system-knows baseline (note that their accuracy is the same, 88.7%, on the DM-group experiment).

Obviously, the information encoded by the DM features provides most clues to the identification of a communication problem for the BWD PR component; however, our empirical investigation shows that other features contribute to this identification as well, since in the experiment drawing on all features a significantly higher score is attained by both classifiers (MBL: $t = 11.9$, $p < 0.01$, RI: $t = 5.4$, $p < 0.01$).

We observe that the contribution of prosodic features is less than we expected on basis of their correlation with the BWD PR class (cf. Section 4.3.3.1). This indicates that in our study prosody provides less information to detecting awareness of communication problems than in other works (cf. Section 2.3.4).

Algorithm	Optimal class label	Feature group	Metric			
			acc	pre	rec	F
MBL	<i>TRA_SLOT_BWD PR</i>	<i>ALL</i>	<i>92.3</i>	<i>93.7</i>	<i>88.1</i>	<i>90.8</i>
			<i>0.9</i>	<i>2.5</i>	<i>2.1</i>	<i>1.2</i>
		DM	88.7	90.5	82.6	86.3
			1.1	3.1	2.8	1.7
		ASR	76.8	77.3	66.0	71.0
			1.7	4.6	2.7	1.8
		PROS	54.4	47.0	42.8	44.6
RI	<i>BWD PR</i>	<i>ALL</i>	<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
			<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>
		DM	88.7	94.1	78.8	85.7
			1.3	3.4	4.0	1.8
		ASR	74.8	76.7	63.3	67.7
			2.9	8.0	13.0	6.3
		PROS	56.4	50.5	37.6	38.8
	<i>TRA_BWD PR</i>	<i>ALL</i>	<i>90.5</i>	<i>92.1</i>	<i>85.1</i>	<i>88.5</i>
			<i>1.1</i>	<i>1.5</i>	<i>2.6</i>	<i>1.6</i>
		DM	88.4	95.7	76.6	85.0
			1.5	1.4	3.9	2.5
		ASR	75.2	78.5	59.6	66.9
			2.8	4.4	10.4	6.1
		PROS	57.4	54.2	6.5	10.8
			2.9	23.7	5.8	8.8

Table 5.11: Performance of the three feature groups in experiments by MBL and RI on the **BACKWARD-POINTING PROBLEM** component with optimised class labelling. The highest score is set in boldface. The italicised top rows in the sections show the scores of learning on all features.

Since the outcomes of the feature partitioning experiments show the same trend across learners and within learners, we conclude that the outcomes of feature partitioning are dependent on the component to be classified, but possibly not on the class partitioning type, i.e., the composition of the class label (see the comparison of BWD PR and TRA_BWD PR). To substantiate this hypothesis would however require more comparisons, which is beyond the scope of the current study.

5.5.3 Detailed analysis

When looking at the induced rule sets from the DM group features, we see that only on two data sets is the default class S_PROB; this shows that in general it is easier for RI to induce rules for the isolated PROB class (note that this was not the case for the FWD PR component). In order to examine classification by RI more thoroughly, we run an experiment on the full data set. The settings we use are the majority values of the settings optimised by WPS in the experiment utilising all features: cover at least 20 examples, find rules for least frequent classes first, i.e., order rules by increasing frequency, expect noisy data, allow negation in conditions, don't simplify hypothesis (i.e., multiply coding cost by 1), set loss ratio to 1, optimise 1 time. Below we reproduce the induced rule set.

- | | | |
|---|---|------------|
| 1 | If 'not' \in sysBOW \wedge 'I' \in sysBOW then PROB. | (849/3) |
| 2 | If 'where' \in sysBOW then PROB. | (312/19) |
| 3 | If 'and' \in sysBOW then PROB. | (109/54) |
| 4 | If 'to_PP' \in prev sysBOW \wedge 'which' \in sysBOW then PROB. | (44/26) |
| 5 | Else OK. | (2023/299) |

There are five rules induced from the data. It illustrates the bias of RI well that the first two rules capture our system-knows baseline. The first rule covers situations when the system apologises for not understanding the user input, the second rule refers to a repeated prompt about the departure and/or destination place (since the first time the system poses a prompt with the word 'where' is the opening prompt, in an answer to which the user can never signal awareness of problems). The third rule conditions on the word 'and' that is in system prompt that verifies travel time (e.g., 'Do you want to travel between four and twelve in the morning?'). The fourth rule seems to indicate a repeated system question about the destination place.

The rule set reveals that most often users become aware of system problems from apologising and repeated prompts. Interestingly, no rules are made that would characterise cases where the system's erroneous implicit verification reveals incorrect input processing; this suggests that these cases are hard to capture on the basis of prompt words only. Note that none of the conditions includes prompt types (e.g. Q_VA). We hypothesise that prompt types are better captured by other, user-input-related features (e.g., the lexical items in the user's word graph), which explains why the isolated DM feature group is unable to outperform the experiment that uses all features.

To ascertain this, we run an experiment on the full data set with all the features available. The settings we use are identical to those used for the DM full experiment above. The induced rule set is displayed below.

- 1 **If** ‘not’ \in sysBOW \wedge ‘I’ \in sysBOW **then** PROB. (849/3)
- 2 **If** ‘where’ \in sysBOW **then** PROB. (312/19)
- 3 **If** ‘o’clock’ \in prev sysBOW \wedge topconf ≥ 777.2 **then** PROB. (97/24)
- 4 **If** ‘to_PP’ \in prev sysBOW \wedge prev BF $\geq 2 \wedge$ ‘to_PP’ \in sysBOW \wedge ‘uh’ \notin prev BOW **then** PROB. (33/7)
- 5 **If** ‘o’clock’ \in prev sysBOW \wedge ‘at’ \notin sysBOW **then** PROB. (71/33)
- 6 **If** ‘to_PP’ \in prev sysBOW \wedge ‘when’ \notin sysBOW \wedge ‘time’ \notin sysBOW \wedge ‘yes’ \notin BOW \wedge ‘which’ \in prev sysBOW **then** PROB. (26/4)
- 7 **Else** OK. (2035/225)

The first two rules, as above, learn the system-knows baseline. The fourth rule seems to be an extended variant of rule 4 from the DM rule set: a repeated system question about the destination place. It includes conditions on the branching factor and the absence of a filled pause (‘uh’) in the previous input’s recognition; these values seem to indicate either that in the previous turn the user was already aware of a problem (the filled pause may cue hesitation), or that the previous user turn was problematically processed (note the large branching factor) and hence needs to be re-entered now. We also see that this rule, although it covers somewhat less examples than its ‘simpler’ version, is more precise (33/7 vs 44/26).

5.6 Discussion

On the basis of our extensive investigation of learner performance on the four components of the SI task, we came to the following conclusions.

5.6.1 Class label design

We investigated task design via systematic class partitioning. The observed trends suggest that class label design has a substantial impact on learner performance. We can establish that it is beneficial to perform class partitioning for the SI task, since all components improved significantly over the score attained in the complex experiment, and also over the prompt baseline (the latter with the exception of MBL on the FWD PR component). A general trend seems to be that it is optimal to combine at most two SI components (optimal in 7 cases out of 8), and, in case two components are combined, one of them should be the TRA component.

It turns out to be optimal for both the TRA and the BWD PR component to be co-learned, which implies that BWD PR could be best labelled as a task-related act and merged in the TRA component of the SI module. This might seem to reflect common knowledge, however, note that previous work on detecting BWD PR (cf. Section 2.2.4) has not drawn such a conclusion. Given this finding, it might in general be advisable for the automatic detection of awareness sites to learn those in combination with other dialogue-act-like class labels (i.e., to detect ‘problem’ and/or ‘slot-filling’ and/or ‘negation’, etc.), instead of learning it in a two-class fashion (i.e., detect ‘problem’ or ‘no problem’).

Entropy in a task does not seem to determine performance. Note that the entropy figures in Table 5.1 are based on sheer counts of class labels, not taking into account that

in the empirical experiments the information in the features and the classifier's bias largely determine learning performance.

5.6.2 Feature design

We have found that in the majority of cases (5 times out of 8) the ASR feature group provides most information to classifying a component. Note that this group contains the largest number of features (1,525). In one case the ASR feature group attains a higher score than the full feature experiment, although not on a significant scale. Three times the DM feature group, containing 944 features, provides most information for classifying a component, once for classifying forward-pointing problems, and twice for classifying backward-pointing problems. In one case this group outperforms the full feature experiment (although not significantly).

Prosodic features (the smallest number of features, 13) in general contribute least to the tasks, suggesting that although it is possible to analyse (components of) the SI task in prosodic terms (in accordance with the findings of e.g. [Hirschberg et al. 2004]), it might be more optimal to combine these features with other, automatically available pieces of information. At the same time, since [Hirschberg et al. 2004] find that prosodic anomalies within a certain speaker's input are more important for predicting recognition problems than anomalies across speakers, our prosodic features might produce better performance when normalised with respect to speaker identity. We conducted a pilot study on problem detection using normalised prosodic features, however, it has not led to more success of the problem components of SI.

We conclude that processing the information coming from all sources turns out to be best for classification, implying on the one hand that it is the largest set of features that seems to be most useful for our task, and on the other, that the SI task may not require explicit selection of features, which is typically a computationally expensive enterprise. On more large-scale investigation of individual features in detecting communication problems see [Lendvai et al. 2002b].

5.6.3 MBL and RI compared

Our empirical investigation shows that learning performance becomes practically identical for both the memory-based learner and the rule induction learner when the tasks are optimised by class partitioning. Most importantly, when the best score of MBL and RI is compared per component, we observe that on all tasks MBL and RI produce statistically indistinguishable performance. This outcome further supports the hypothesis and the evidence supplied for the findings of [Daelemans et al. 1999, Daelemans and Hoste 2002], whereas it contradicts those in [Rotaru and Litman 2003] (see Section 3.1).

In general, the scores produced by RI are somewhat lower than those by MBL, and also exhibit more variation. RI produces improvement in the class partitioning experiments on a larger scale than MBL; obviously, it has a much larger margin to improve on as compared to the complex experiment, in which MBL produced relatively high scores already. We see that RI often provides insight into the features that it can use particularly well in a task. It would be possible to analyse feature usage in MBL as well, for example by observing

assigned feature weights and selected nearest neighbours; however, such an analysis lies out of the scope of the current study.

5.6.4 Evaluation

The extent to which we are able to classify the individual SI components is not easy to compare with those in other studies, since many important factors are quite different in most of the surveyed studies (cf. Section 2.2); not only the learners, the computational costs of the method, and the fine-grainedness of classes, but the employed evaluative measures as well. For an easy overview, in Table 5.12 we reproduce the best scores attained on the individual SI components in the information partitioning experiments described in this chapter.

Algorithm	Class label	Metric			
		acc	pre	rec	F
MBL	TRA	86.6	94.3	89.3	91.7
		0.7	1.5	1.0	0.7
	SLOT	83.5	90.7	84.9	87.7
		2.2	1.8	2.6	2.0
	FWD PR	68.6	67.1	53.5	59.4
		2.6	4.4	5.4	4.1
	BWD PR	92.3	93.7	88.1	90.8
		0.9	2.5	2.1	1.2
	TRA	86.0	92.0	89.1	90.5
		1.7	2.0	1.5	1.5
RI	SLOT	82.6	88.4	82.9	85.5
		2.4	4.4	3.8	2.8
	FWD PR	65.0	57.9	72.1	62.6
		2.0	3.9	12.2	5.2
	BWD PR	90.5	92.4	85.1	88.5
		1.4	3.5	3.9	1.5

Table 5.12: Best scores produced by MBL and RI on the shallow interpretation components in optimal class- and feature design, averaged over 10-fold CV experiments.

The best scores reported in the literature on classifying dialogue acts (note that these are more large-scale than our TRAs) are in the range of 70-80% accuracy, recall, or F-score (cf. the surveys in [Choi et al. 1999, Reithinger and Engel 2000]). The work that describes the study most similar to ours about classification of slots is [Cettolo et al. 1996], who report 67.2% accuracy on classifying recognised speech in terms of slots in the travel domain (by combining a rule based module with binary classification trees). At the same time, [Rayner and Hockey 2003] attain 77.8% accuracy on extracting semantic atoms from spoken utterances, combining rule-based and n -gram-based methods. On classifying forward-pointing problems [Litman et al. 2000] and [Walker et al. 2000b] report

a top 93.5%, respectively 86.2% accuracy. On classifying backward-pointing problems, [Litman et al. 2001] report 80.7% precision and 81.1% recall.

5.7 Summary

We hypothesised that learning the complex class label incorporating the four SI components (cf. Chapter 4) might not yield the optimal performance that can be attained with our shallow approach to extracting pragmatic-semantic information from spoken user input. Our goal in the current chapter was to learn an optimal class and feature combination for each SI component. The experimental results support our hypothesis that automatically searching for optimal class label combinations is possible, and that the results of this can substantially improve ML performance on a task. Feature partitioning however proved not to be beneficial for the learners on the SI task, suggesting that our learners best utilise information from all knowledge sources in the SI task.

On the basis of the conducted analyses we can state that MBL and RI produce practically identical results on the SI task when both their parameters and the task composition are optimised. We find no evidence that memory-based learning and rule induction attain different performances depending on the task — reflected by the four different SI components — or the employed features — reflected by the three different feature groups.

The fact that some members of some feature groups seem to supply more information to the learners than others might seem to be a data-specific finding (see the analyses of the induced rules). We would like to emphasise however that the goal of our study is not to generate observations pertaining to the specific feature values reported here, but to estimate the extent to which our method, pursued by a shallow approach for the detection of a limited set of basic user input types — that nonetheless aim to be general — of human-machine interaction, is capable of producing a shallow interpretation of user input to a SDS. From an explanatory perspective however, inspecting the rules induced from feature values can ex

Chapter 6

Filtering Information

The findings of the previous chapters revealed that our classifiers successfully use the ASR word graph features in the SI module. However, noise that is supposedly present in these features, especially in the bag-of-words features, might interfere with their optimal utilisation. In 1.1 we emphasised that both the user input to a SDS and the ASR output are often noisy. Disfluencies such as filled pauses, repetitions, stutters, and ungrammatical constructions are a main example of noisy user input; in fact, the occurrence of disfluencies in user utterances is regarded as a stumbling block for speech recognition, aggravating imperfect NLU (cf. [Stolcke et al. 1998b, Duchateau et al. 2003]). The ASR output often contains incorrectly recognised words, for instance because input containing words that are not covered by the ASR grammar may become completely garbled [Rayner and Hockey 2003].

Another cause of noise in the input might be the following. [Cettolo et al. 1996] claims that not all semantic contents of an utterance may be relevant to the communication. For example, based on the literature, syntactic head words seem to be more important in NLP tasks than non-head words. Since it could be that some of the ASR features represent noise or superfluous information that may have negative effect on interpreting the spoken input, we hypothesise, based on the literature, that removing such items may lead to improved performance of the SI module.

Recently there is an increased interest in applying natural language processing techniques directly to the word graph or the n -best list output of an ASR, which is claimed to increase robustness of the interpretation system [He and Young 2004]. Some of these approaches (e.g., [Van Noord et al. 1999]) are deployed in a task and language dependent way, implying that they need to be redeveloped for each new system. Others use statistical techniques to model errors in the ASR output, and utilise the outcome of such processing for information extraction purposes [Palmer and Ostendorf 2001], or to disambiguate the ASR output [Koeling 2002]. At the same time, hand-crafted rules combined with various data-driven classification techniques are also utilised for a variety of NLU tasks that incorporate the treatment of noise, e.g. disfluency detection [Stolcke et al. 1998b], recognition error correction [Ringger and Allen 1997, Stolcke et al. 1998b, Zechner and Waibel 1998], interpretation of command and control tasks [Nakano et al. 1999, Rayner and Hockey 2003],

speech-to-speech translation [Kiefer et al. 2000].

Needless to say, directly processing the raw ASR hypotheses is a complicated task, since instead of a linear sequence of words a graph or a large number of word strings (i.e., the paths generated from the graph) need to be treated. Furthermore, [Boves et al. 1996] emphasise that such input may be defective since the recogniser may miss essential words completely, whereas [Zechner and Waibel 1998] claim that the paths themselves may be ungrammatical in a different way than disfluencies.

In the current chapter we investigate whether filtering the word graph output, transformed into n -best paths, improves performance of the SI module. We describe three approaches that attempt to filter out those items from the n -best lists that may correspond to noise or to superfluous information in the spoken input: (i) disfluencies, (ii) words that are not the head of their syntactic chunk, and (iii) words that do not belong to the set of 15 most frequent words in the bag-of-words. Filtering is applied to the user's BOW contained in the feature vector, and the filtered feature vector is subsequently used in learning the SI task as optimised in the previous chapter.

The first filtering method attempts to remove potential disfluencies from the word graph automatically, since previous work suggests that this might be beneficial for automatic processing of natural language (cf. e.g. [Heeman and Allen 1994, Spilker et al. 2001, Spilker et al. 2000]). We describe ML-based techniques that aim at filtering disfluencies from the BOW vector: classification of disfluencies is carried out both by MBL and RI that are trained on the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN). The disfluency filtering technique is first evaluated on the CGN, and is subsequently applied to the word graph material of the OVIS corpus.

Our second approach to filtering is to edit the user BOW based on syntactic knowledge. Syntactic information is often used in interpretation tasks (cf. Section 2.3). We again take a shallow approach: syntactic analysis of the recognition lattice paths is performed with a memory-based shallow parser tool, based on which all words but syntactic chunk heads are removed from the set of recognised words. The words that are syntactic chunk heads in the word graph paths are then used in the shallow interpretation task. Syntactic chunk head words are claimed to represent semantic aspects of the entire chunk well enough to be utilised in higher-level NLP tasks (cf. e.g. [Buchholz 2002, Hacıoglu et al. 2004]). Our technique of chunk non-head filtering allows to test the performance of syntactic chunk heads in the SI module.

The third filtering method is based on word frequency. The approach taken in the work of [Rotaru and Litman 2003] is to select the 9, as well as the 15 highest-ranked features out of 141 according to their information gain calculated in various language learning tasks (among others in classifying speech recognition errors and user awareness in spoken dialogues). We take a slightly different approach by selecting the same amount (15) from the most frequent words in the user's BOW, and filter out all other words from the BOW. The impact of frequency-based filtering is again measured directly on the SI task.

The structure of the chapter is the following. In Section 6.1 we describe disfluency filtering: we introduce the notion of a disfluency, and report on earlier work on automated processing of disfluencies. In the first part of the section our approach to ML-based disfluency detection is explained, introducing the training material (the CGN corpus), and reporting on the performance of our technique on the CGN corpus. In the second

part we give an account of how disfluency detection is performed on the OVIS word graph paths, and finally present the experimental findings about incorporating the disfluency filtered BOWs into the SI module.

In Section 6.2 we utilise a memory-based shallow parser to identify syntactic chunk heads in the OVIS material, after which the chunk heads are used (together with the other features) to classify the SI components. Finally, in Section 6.3 we give an account of frequency-based filtering. The chapter is concluded by discussing the findings about the filtering approaches, and by summarising the treated issues.

6.1 Filtering disfluencies

The group of speaker phenomena commonly referred to as disfluencies includes hesitations, filled pauses, laughter, repetitions, false starts, abandoned grammatical constituents, incompletely uttered words (also called as fragments), self-corrections, and the like. Although human listeners are good at handling disfluent items in spoken language utterances (cf. [Levelt 1989, Shriberg 1994]), they are likely to cause confusion when present in the input to automatic NLP systems, resulting in poor human-computer interaction [Nakatani and Hirschberg 1994, Eklund and Shriberg 1998].

It has not been established in the literature in what ways precisely disfluencies introduce ungrammaticality into the structure of an utterance. Figure 6.1 shows part of an utterance that has disfluent elements in it. The example, taken from the CGN corpus, features a fragmented word (*‘interne-’*, i.e., an incomplete version of the word *‘internet’*) within the larger chunk of the abandoned constituent *‘van interne’*, which is corrected by the speaker after the editing term *‘sorry’* by replacing it with *‘van electronic commerce’*. As the example shows, often there is structure in and around disfluent chunks: before the interruption point (i.e., where the speaker interrupts himself) there may be word(s) meant to be erased (called the *reparandum*), whereas the word(s) that follow it (called the *repair*) may be intended to replace the erased part.

Traditional implementations of speech recognisers, taggers, and parsers do not treat disfluent passages in the input as constituents of the sentence but rather as items that need to be discarded [Bear et al. 1992, McKelvie 1998, Nakatani and Hirschberg 1994, Oviatt 1995, Spilker et al. 2001]. As opposed to this, some research has also focused on processing ill-formed input: for example, [Charniak and Johnson 2001] claim that once a parser is trained on ungrammatical data, ill-formedness in new material does not have a negative effect on the parser.

In many situations disfluent items can play a discourse role (cf. [Levelt 1989] on context-sensitive monitoring), which may be useful to take into consideration for a number of NLP tasks. Our approach in the current study is simply to locate and remove disfluent passages in the word graph lattice of recognised dialogues, and investigate whether doing so improves further processing of these dialogues.

'... het veilig gebruik [van interne--¹ ² sorry³]⁴ van electronic commerce⁵ ...'
 ... the safe usage of interne- sorry of electronic commerce ...

Figure 6.1: Example disfluency from the CGN corpus, sample nr. fn000056. Notation: 1: reparandum, 2: interruption point, 3: editing term, 4: disfluent chunk, 5: repair. Note that usually the first occurrence of a repeated word is regarded as disfluent (e.g. in our case the first occurrence of 'van' is marked as inside the disfluent chunk).

6.1.1 Filtering disfluencies from transcribed words

Researchers who have worked on automatic disfluency detection in the past decades include [Hindle 1983, Bear et al. 1992, Heeman and Allen 1994, Nakatani and Hirschberg 1994], [Oviatt 1995, Shriberg et al. 2001]. Most of their work involves relatively small datasets, since annotating discourse for disfluencies is a difficult and time-consuming process. In addition, many of these studies tend to focus on a subset of disfluent phenomena, such as repairs or fragmented words, and are usually concerned with (American) English. Exceptions include [Eklund and Shriberg 1998, Eklund 2004] on Swedish, and [Spilker et al. 2001] on German. Many studies invest in trying to group disfluencies according to various aspects, which is often a hard task since these phenomena are manifold in nature, an issue which is extensively treated in [Eklund 2004].

In the literature it is often assumed that once the interruption point is determined (most often by identifying an incompletely pronounced word), it is possible to carry out complete reconstruction of the correct sentence structure automatically [Bear et al. 1992, Heeman 1999, Shriberg et al. 2001]. The presence of a fragmented word is often regarded as an integral property of a speech repair and is employed as a readily available feature in automatic processing of disfluencies (cf. [Nakatani and Hirschberg 1994]), although [Spilker et al. 2001] and [Heeman 1999] observe that finding word fragments automatically is an unsolved problem, since automatic identification of a fragmented word is not straightforward. In [Lendvai 2003] we concentrated on the automatic detection of fragmented vs non-fragmented words. Our pilot study concluded that classification of fragmented words could be carried out by memory-based learning with a 74.9 F-score. Moreover, unlike some disfluency types, for example filled pauses ('uhm'), fragmented words are typically not recognised by ASRs.

In the current study we aim at detecting any types of disfluencies via classification by MBL and RI. Since disfluencies are not systematically indicated in the transcriptions of user input in the OVIS corpus, we lack the possibility to train on OVIS. After listening to the recorded speech material of the OVIS corpus however, we concluded that OVIS contains artificially clean transcriptions, since some user turns contain disfluent or ungrammatical items that are not transcribed in the corpus. In addition, [Eklund 2004] reports that disfluencies frequently occur in human-machine travel booking dialogues.

Therefore, in the current study our two classifiers are trained on the CGN which contains annotations of disfluencies. Classification performance is first evaluated on the CGN, after which the same technique is applied to the paths in the speech recognition lattice of the OVIS dialogues. The impact of filtering is directly measured on the SI task.

6.1.1.1 Introduction to the CGN corpus

The CGN contains human–human dialogues, monologues, and multilogues that are sampled from different regions of the Netherlands and the Flemish area of Belgium. This corpus is designed to provide a research source for language and speech engineers and linguists. It consists of 1,000 hours of orthographically transcribed speech from adult speakers of contemporary standard Dutch. For approximately 100 hours of speech, which is about one million words from the total material, detailed annotations are made on the phonetic and syntactic levels. The discourses are of various levels of spontaneity, ranging from television broadcasts to telephone conversations. The number of speakers in CGN spans from 1 (in newsreading files) to 7 (in parliamentary sessions). Each speaker is assigned a unique identification code. For details on the overall design of the corpus see [Oostdijk 2002].

For the current study we made use of CGN Release 6. As disfluencies are reported to occur both in dialogue and monologue [Shriberg et al. 2001, Eklund 2004], we make use of 1-speaker data as well. Our material comprises a representative sample of 1,322 full discourses, consisting of 1,009,968 lexical tokens in 129,932 utterances. This means that one dialogue consists of 100 sentences on average. Utterance segmentation is performed automatically in the corpus, based on the detection of longer pauses in the speech material. The punctuation marks at the end of segments can be period ‘.’, question mark ‘?’, and a row of three dots ‘...’, the latter standing for unfinished sentences. The average utterance length is 7.8 words in the corpus. All sentences are orthographically transcribed, part-of-speech-tagged, lemmatised, and morpho-syntactically tagged [Van der Wouden et al. 2002]. In addition, a full syntactic dependency tree is manually built for each utterance.

We illustrate the CGN material in Figure 6.2, featuring the first turns of a spontaneous dialogue, and in Figure 6.3, showing a more restricted 2-speaker discourse from the corpus. The dialogues are translated into English. The Dutch transcriptions are provided in Figures 4 and 5 of the Appendix.

In the CGN tokens are marked for several speech phenomena, among others for disfluencies. These include the following, their amount in our material given in brackets:

- filled pauses (31,682): ‘uh’, ‘uhm’, ‘hu’, ‘hm’, ‘mmm’, ‘mm-hu’, etc.
- editing terms, empty coordinating conjunctions, discourse markers (56,832): ‘oh’, ‘tjonge’ (*gee*), ‘hoor’ (an emphasiser), ‘hè’ (*huh*), etc.
- mispronounced but complete words, (self-created) onomatopoeic words (1,298): ‘hij belde [belde] niet’ (*he did not clal [call]*)
- fragmented words (9,073): ‘hij be- belde niet’ (*he did not c- call*)
- garbled material (6,921)
- laughter, coughing, crying (8,045).

[Eklund and Shriberg 1998, Eklund 2004] describe a rare disfluency phenomenon: a filled pause occurring inside a word, observed in Germanic languages that heavily use compounding as a word-formation method. The filled pause in such cases stands mostly

Turn	Utterance
S1	I have to go to uhm Hilde whose uh whose upper body is of course still in plaster.
S2	mm-hu.
S1	and then I have to go to tae-bo.
S2	well they won't know how to spell that, want to bet?
S1	and I also have to... haha no.
S2	haha. no seriously.
S1	yes? tae-bo that is T A E hyphen B O but this name is must not be used anymore.
S2	oh?
S1	because... haven't you heard it?
S2	no.
S1	because uh Billy Blanks the guy who uh invented tae-bo is uhm...
S2	you don't need t- e- you don't need to explain. it's not about the contents.
S1	no but I'd like to tell this to you. GARBLED.
S2	GARBLED. yeah OK yeah but I know who you mean yeah.
S1	yeah well tae-bo... so Billy Blanks has this uh has a lawsuit.
S2	yeah?
S1	and so this concerns that that so this name tae-bo was so to speak invented by him and it took him fifteen years to develop all that and so on.
S2	yeah.
S1	and so he thinks that only he should be allowed to use it. so that only he should be allowed to sell the videotapes and so that sports schools shouldn't use the name tae-bo.
S2	without that they GARBLED yeah.
S1	without that they pay a whole lot of money to h-... so he just wants to earn a whole lot of money with that. and of course he just earns oh tons of money with that tae-bo.
S2	yeah yeah.
...	...

Figure 6.2: The first turns of a spontaneous dialogue sampled from the CGN corpus (sample nr. fn000451), translated into English.

Turn	Utterance
S1	the Flemish government has a new health care insurance planned starting in the middle of next year. the heavily disabled are to receive financial compensation for their non-medical costs. this amount may be between three thousand five hundred and six thousand five hundred francs per month. the system is financed by the Flemish government as well as by the citizens. these pay thirty francs per month via their governmental or private health insurance. Johny Vansevenant.
S2	the decision that a health care insurance should be introduced was made already by the previous Flemish government. but its actual realisation has not yet been carried out. the system will now be organised via the governmental or private health insurance companies. these will collect a contribution of thirty francs per month from their members.

Figure 6.3: The first turns of a broadcast dialogue sampled from the CGN corpus (sample nr. fv600473), translated into English.

between two stems in a compounded noun (e.g., ‘beach-uh-volley’), but – in highly rare cases – also within morphological constituents of a word (e.g., ‘daar-uh-door’; *there-uh-fore*). In our material such a filled pause is present 202 times.

At the same time, disfluency phenomena happening at the sentence level, such as repairs of ill-formed phonetic, syntactic or semantic constituents of the sentence, abandoned grammatical constructions, repetitions, and the like, are not explicitly marked in the corpus. However, part of these can be inferred from ‘...’ punctuation marks (marking abandoned sentences that may or may not be continued), and from the syntactic dependency tree annotation of a sentence, in which disfluent items are not connected to the full tree. Figure 6.4 contains an example sentence from the CGN corpus with the complete morpho-syntactic analysis tree. Note that certain leaves are not connected to the tree: in the given sentence such left-out leaves consist of a false start (‘ik uh’; *I uh*), a filled pause ‘uh’ after the word ‘sceptis’ (*scepticism*), and a repetition (zo’n; *such a*).

By definition, we consider all items that are not incorporated under the syntactic tree as disfluencies. These include some, but not all, of the word-level encodings listed above. Some disfluencies are still connected to the syntactic tree when they have a full pragmatic, semantic, or syntactic role in the sentence. This is of course hard to annotate consistently, which probably introduces some noise into the mark-up, despite the clear-cut annotation guidelines and transcription protocols of CGN (cf. [Van der Wouden et al. 2002]). It also entails that some out of the 11,648 abandoned sentences (i.e., the utterances ending with ‘...’) are still incorporated in (bigger) syntactic trees.

[Oviatt 1995] finds that in human-human dialogues there are more disfluencies than in human-computer dialogues, due to the lack of constraints in the presentation form of the former. Spontaneous spoken dialogue is especially abundant in disfluent events. Such dialogues are less focussed on a topic or a task, resulting in a more relaxed way of speech construction and speech planning. According to our criteria, in our full material there are

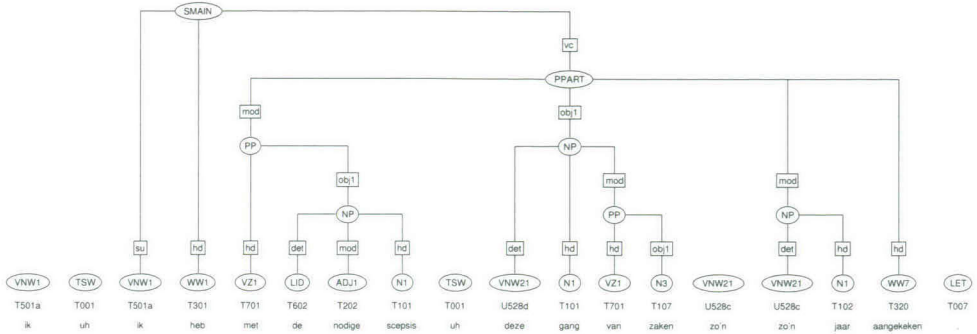


Figure 6.4: Example sentence with full morpho-syntactic tree from the CGN: ‘ik uh ik heb met de nodige scepsis uh deze gang van zaken zo’n zo’n jaar aangekeken’ *I uh I have followed this process with uh a certain amount of scepticism for for about a year.*

49,577 disfluent items, constituting 34,423 bigger disfluent chunks. This means that 9.1% of all lexical tokens in the data are part of a disfluent chunk, and a disfluent chunk consists of 1,4 items on average. In Figure 6.4 we have for example three disfluent chunks.

We assign the class label IN-DISFL to all disfluent words, and the class label OUT-DISFL to all other words in the corpus. The learning task will be to classify each token in the corpus in terms of these two classes.

6.1.1.2 Features

In our study the identification of cues for detecting disfluencies is based on close inspection of the data and on the literature [Plauche and Shriberg 1999, Heeman 1999, Shriberg et al. 2001, Nakatani and Hirschberg 1994, Oviatt 1995]. We focus on using word-based information only, in order to investigate the feasibility of disfluency detection with readily available features. [Heeman and Allen 1994] assume that local context is sufficient in detecting most speech repairs, without taking syntactic well-formedness or speech prosody into consideration. At the same time, we consider that exploiting the manually annotated part-of-speech (POS) labels would give too much advantage to our model, as opposed to a disfluency detection task in a real implementation where no 100%-correct POS information would be available. Furthermore, in the POS annotation of the CGN disfluent items have a unique POS label instead of a true syntactic role label, which would be a give-away feature during the classification task.

Table 6.1.1.2 lists the 31 contextual properties that we extracted automatically from the corpus material, subdivided into groups according to the aspect they describe. Nine lexical string features represent the focus item itself and its neighbouring four left and four right unigram lexical contexts (if any). Our lexical context window is therefore of length nine. The second feature group consists of 20 binary features that mark whether an overlap in wording occurs between the focus item and its context window. Two features in the third group represent overlap in initial letters between the focus item and its immediate

Aspect	Feature
Lexical identity	(1) Left4 context item (2) Left3 (3) Left2 (4) Left1 (5) Focus item (6) Right1 (7) Right2 (8) Right3 (9) Right4
Lexical overlap	(1) Left4/Left3 (2) Left3/Left2 (3) Left2/Left1 (4) Left1/Focus (5) Focus/Right1 (6) Right1/Right2 (7) Right2/Right3 (8) Right3/Right4 (9) Focus/Left4 (10) Focus/Left3 (11) Focus/Left2 (12) Focus/Right2 (13) Focus/Right3 (14) Focus/Right4 (15) Left1/Right1 (16) Left1/Right2 (17) Left4/Left2 (18) Left3/Left1 (19) Right1/Right3 (20) Right2/Right4
First Letter overlap	(1) Left1/Focus (2) Right1/Focus

Table 6.1: Overview of the employed features for the disfluency detection task, grouped according to their aspect.

left and right context. Matching words or word-initial letters are often to be found both at the reparandum onset and the repair onset, as in the correction ‘**van** interne– **van** electronic commerce’ (*of interne– of electronic commerce*) in Figure 6.1.

By employing these features we allow the learners to make use of possible correlations between certain feature values and the potential presence of a disfluent item. Note that some features of the overlap groups deliberately re-introduce properties that are implicitly present in the lexical features already. We found it important to express the word and letter overlaps explicitly in order to ensure that the learners, that otherwise may be unable to capture sub-wordform similarities, can utilise potentially relevant information.

6.1.1.3 Experimental set-up

Our first goal is to classify disfluent chunks in CGN based on the above contextual properties of each utterance. The experimental set-up here is the same as throughout our study: we train MBL and RI and conduct parameter optimisation by WPS in a 10-fold CV, where partitioning is based on whole discourses. The performance of the learners is evaluated in terms of accuracy, precision, recall, and F-score, where accuracy measures the overall percentage of correctly predicted IN-DISFL and OUT-DISFL class labels. Thus, in the example sentence in Figure 6.4 both words in ‘ik uh’ need to be classified as IN-DISFL to count as a correct classification of the chunk. Likewise, precision, recall, and F-score apply to entire chunks in our evaluation.

An additional technique we use in these experiments is attenuation. Infrequent or unknown words are often problematic for machine learning techniques since the occurrence statistics of such items are unreliable. At the same time, the word form of infrequent items may contain useful information; for instance, a capitalised word is likely to be a named entity, a word that contains a number is usually either a digit or the name of an object (e.g., TU-154), a hyphen tends to indicate compounding. In addition, the final letters of a word may give away morphological clues, e.g., -ly (adverb) in English, or -dt (verb) in

Dutch. Attenuation is a masking technique for words occurring below a certain frequency threshold, retaining some word form information for these while masking the actual word. Besides addressing the sparse data problem, another advantage of attenuation is that it reduces the search space since the number of different feature values that need to be checked becomes much smaller. The attenuation method we use is a simplified version of [Van den Bosch and Buchholz 2002], which is in turn based on a proposal by [Eisner 1996]:

- **If** a word occurs less than 100 times in the training data **then** convert it to MORPH **and**
 - **if** it contains a number **then** add -NUM
 - **if** it contains a hyphen **then** add -HYP
 - **if** its first letter is a capital **then** add -CAP
 - **if** none of these three tests apply **then** add the last two letters of the word
- **Else** retain the original word.

For the example sentence in Figure 6.4 this strategy produces the sequence ‘ik uh ik heb met de MORPH-ge MORPH-is uh deze MORPH-ng van zaken zo’n zo’n jaar MORPH-en’ (approximately: *I uh I have MORPH-ed this MORPH-ss with uh a certain MORPH-nt of MORPH-sm for for about a year*). The attenuation method is applied to each training and test data set, creating attenuated versions of both; the frequency thresholds are established based on the training data sets. We hypothesise that for the current learning task attenuation will not have a negative effect (and might even have a positive effect) since the binary overlap features, which are not based on the attenuated words, are likely to compensate for some of the potential information loss.

6.1.1.4 Baseline

To quantify the performance of our disfluency detection method, we need to define a baseline. The most straightforward baseline is to always predict the majority class: most words in the corpus are not disfluencies, thus this baseline amounts to always predicting OUT-DISFL, resulting in correct prediction in 89.9% of the cases. However, for the class of interest (IN-DISFL) this strategy leads to a recall of 0 (all disfluencies are missed), an undefined precision and hence an undefined F-score.

A somewhat more intelligent baseline is the following. The most frequent kind of relatively easily detectable disfluencies are four basic filled pauses (FPs), transcribed as ‘uh’, ‘uhm’, ‘hu’, and ‘hm’ in the CGN corpus. We define a FP-baseline that predicts that all filled pauses are disfluencies and everything else is not. This baseline has an accuracy of 92.3%, a relatively high precision (not 100%, since one in four filled pauses is part of a larger disfluent chunk), a similar recall (it misses most disfluent chunks) and an F-score of 74.5 (displayed in Table 6.2).

6.1.1.5 Results of testing on CGN data

Table 6.2 shows the average performance of MBL in three series of 10-fold CV experiments, as well as the FP baseline. The result of the experiment is that both classifiers outperform

Algorithm	Metric			
	acc	pre	rec	F
FP baseline	92.3	76.0	73.1	74.5
	1.8	5.7	5.1	5.2
MBL	97.9	87.6	84.9	86.2
	0.6	1.4	2.2	1.8
RI	97.3	87.1	78.6	82.6
	0.8	2.0	3.4	2.7

Table 6.2: Performance on detecting disfluent chunks in CGN by MBL and RI, averaged over 10-fold CV experiments, in comparison with the filled pause baseline.

the FP-baseline significantly (MBL: $t=7.6$, $p<0.01$, RI: $t=4.7$, $p<0.01$). The accuracy of both classifiers is rather high (MBL: 97.9%, RI: 97.3%). The F-score of classifying disfluent chunks is 86.2 for MBL and 82.6 for RI, which is a 11.7, respectively 8.1 points increase compared to the baseline strategy, due to improved precision and recall on the IN-DISFL class. MBL produces a significantly better F-score than RI ($t=7.1$, $p<0.01$).

Note that although the algorithm parameters are optimised in this experiment, the class design is not, since the task here is to perform binary classification. Optimising the class label in such cases is not as ‘straightforward’ as we have performed it in Chapter 5. It is an empirical issue in what ways it would be possible to decompose the global IN-DISFL and OUT-DISFL classes in a robust way.

The settings resulting from the optimisation process by WPS (wrapped progressive sampling) show a clear trend of algorithm parameter use. For MBL, in nine folds the MVDM (modified value difference) distance metric is found optimal, in all but one cases combined with GR (gain ratio) feature weighting. In general, a k larger than or equal to 3, but smaller than or equal to 19 is used. The most reliable features for the learner are the focus word itself, as well as whether the focus word overlaps with the immediate right or second right word in the context window.

For RI, in 8 out of 10 folds it is optimal to cover a single example per rule, and to simplify the induced hypothesis. In all folds it is optimal to order the rules by increasing frequency, and to allow negation. We run an experiment on the full data set with RI with these settings, and display the obtained rule set below. Feature names are the following: FOC: focus word, L1: immediate left context word, R1: immediate right context word, LX FOC/L1: lexical overlap between focus word and immediate left context word, LT FOC/R1: first letter overlap between focus word and immediate right context word, and so on.

- 1 **If** FOC = ‘uh’ **then** IN_DISFL. (25381/184)
- 2 **If** L1 = EMPTY \wedge R1 = EMPTY **then** IN_DISFL. (31919/21)
- 3 **If** LX FOC/R1 \wedge FOC \neq ‘yes’ \wedge FOC \neq ‘that’ \wedge FOC \neq ‘no’ \wedge R4 \neq EMPTY **then** IN_DISFL. (5072/807)
- 4 **If** LX FOC/R2 \wedge R1 = ‘uh’ **then** IN_DISFL. (1595/129)
- 5 **If** LX FOC/R1 \wedge FOC \neq ‘yes’ \wedge FOC \neq ‘you’ \wedge FOC \neq ‘that’ \wedge FOC \neq ‘no’ \wedge R3 \neq EMPTY \wedge !LX FOC/L2 **then** IN_DISFL. (415/99)

6	If FOC = GARBLED \wedge L3 = EMPTY then IN_DISFL.	(2194/154)
7	If LX FOC/R1 \wedge FOC \neq 'yes' \wedge FOC \neq 'you' \wedge FOC \neq 'that' \wedge FOC \neq 'no' \wedge R2 \neq EMPTY \wedge !LX R1/R2 then IN_DISFL.	(349/102)
8	If LX FOC/R2 \wedge LX L3/L1 \wedge !LX FOC/R1 then IN_DISFL.	(960/298)
9	If FOC = 'uhm' then IN_DISFL.	(2698/28)
10	If FOC = 'haha' then IN_DISFL.	(2300/43)
11	If LX FOC/R2 \wedge FOC \neq 'yes' \wedge LX L1/R1 \wedge !LX L1/FOC \wedge R3 \neq EMPTY then IN_DISFL.	(873/170)
12	If FOC = GARBLED \wedge L1 \neq 'the' then IN_DISFL.	(2323/400)
13	If LT L1/FOC \wedge LX FOC/R1 \wedge FOC \neq 'yes' \wedge FOC \neq 'that' \wedge L2 \neq EMPTY then IN_DISFL.	(168/48)
14	If LT L1/FOC \wedge LX FOC/R1 \wedge FOC \neq 'yes' \wedge FOC \neq 'that' \wedge FOC \neq 'no' then IN_DISFL.	(82/14)
15	If LT L1/FOC \wedge LX FOC/R1 \wedge FOC \neq 'yes' \wedge FOC \neq 'that' \wedge FOC \neq 'no' \wedge L3 \neq 'you' \wedge R4 \neq EMPTY then IN_DISFL.	(52/2)
16	Else OUT_DISFL.	(905519/25698)

We see that RI uses the overlap (most often: lexical overlap) features extensively, both between the focus word and its context, or between context words. Many conditions are made on the identity of the focus word: for example, if it is a filled pause, a garbled item, or laughter, then it is classified disfluent; and when it is carrying some specific, but apparently, generally important content, such as the words 'you', 'yes', and 'no', then it is not disfluent. Note that the lexical items are translated into English, but in some cases this might be misleading since conditioning on the Dutch item 'de' can correspond both to a definite article (i.e., 'the') and to a word fragment (i.e., de-); the same may hold for a number of other short 'true' words (e.g., je-, ne-, etc.).

In fact, 59 times 'de' is marked as a fragment in the CGN (but not marked as such in our experimental material, since that would be a give-away cue), and 'je' is marked 3 times as a fragmented word. Even if in some cases 'yes' corresponds to a fragmented word, we believe that in the majority of cases conditioning on 'yes' and 'no' means conditioning on the affirmative, respectively negative true word (and not a fragmented word). Our classifiers thus learn that 'yes' and 'no' are most of the time not annotated as disfluencies (e.g., as filler words) in CGN.

6.1.2 Filtering disfluencies from recognised words

6.1.2.1 Preprocessing the OVIS data

Having trained on the CGN, the next step is to apply the disfluency filter to the OVIS material. First we preprocess the OVIS word graph lattices by unfolding all paths in each lattice. As an illustration of this, consider the word graph in Figure 6.5. This graph represents the ASR output produced after processing the input 'ik moet volgende week dinsdag van schiphol naar nijmegen' (*i need to go next week tuesday from schiphol to nijmegen*), which is the first user turn in the dialogue in Figure 4.2. Unfolding the lattice results in eight paths: 'ik moet volgende week dinsdag van schiphol naar nijmegen' (*i need to go next week tuesday from schiphol but nijmegen*), 'ik moet volgende week dinsdag

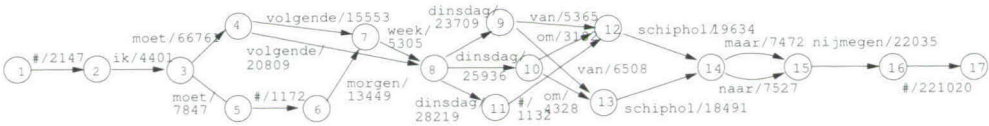


Figure 6.5: Word graph of the user input in turn U1 of Figure 4.2 ‘ik moet volgende week dinsdag van schiphol naar nijmegen’ (*i need to go from schiphol to nijmegen on tuesday next week*). Hash marks stand for pauses, the confidence score of each word hypothesis is given after the slash.

van schiphol naar nijmegen’ (*i need to go next week tuesday from schiphol to nijmegen*), ‘ik moet volgende week dinsdag schiphol naar nijmegen’ (*i need to go next week tuesday schiphol to nijmegen*), ‘ik moet volgende dinsdag schiphol naar nijmegen’ (*i need to go next tuesday schiphol to nijmegen*), and so on. Note that the word graph in principle unfolds in many more than eight paths, given the various transitions that contain pauses, and/or tokens with different transition probabilities; however, when generating the word strings, identical strings are collapsed.

As a result of preprocessing all the word graphs in the OVIS corpus, we have a material of 17,242 paths with 90,527 words to classify.

6.1.2.2 Results of testing on OVIS data

Since disfluencies are not annotated in the unfolded paths, we cannot directly measure the performance of the learners on detecting disfluencies in the OVIS material; however, it is possible to make some observations with respect to the words that are classified as disfluent. Looking at the classified material, we see that repetitions and filled pauses are correctly detected in the experiment. Below are four classification examples, the items classified disfluent marked by brackets, an approximate translation given in *italics*:

- (i) nee [ik wil] ik wil verbinding
no [I want] I want connection
- (ii) ik wil graag van Den Haag Mariahoeve naar [van] van Centraal
I want from The Hague Mariahoeve to [from] from Central
- (iii) [uh] van of naar Zwolle
[uh] from or to Zwolle
- (iv) wil van Zaandam [naar Arnhem] naar Arnhem
want from Zaandam [to Arnhem] to Arnhem.

In fact, we see that disfluency filtering is capable of eliminating proper disfluencies, however, contrary to what is hypothesised in the literature, this does not seem to result in more grammatical, or less ambiguous word strings (i.e., the string ‘I want from The Hague Mariahoeve to from Central’ is syntactically or semantically not substantially better formed than ‘I want from The Hague Mariahoeve to from from Central’, etc.).

At the same time, we can establish that stylistic differences between the training and test data have some unfavourable effects on classification. In particular, in the CGN short sentences that consist of one or two items are most often regarded as fully disfluent (i.e., abandoned) chunks. Therefore, short user input, which is typical in interactions with a SDS, is often classified disfluent in the OVIS material (by both classifiers). This is obviously wrong in many cases: for example when the user provides only a station name or a day in reply to a system prompt. It also often happens that the user answers simply ‘yes’ or ‘no’ (to a yes/no system prompt), and these words are classified as disfluent.

We have seen above in the experiment on CGN material that many times ‘yes’ and ‘no’ are distinctively not classified as disfluencies (see the RI rules). We assume that the contextual differences around these tokens in the two corpora are large enough to result in a different treatment (by both learners) of the same tokens (cf. rules 3, 5, 7, 11, 13, 14, 15 in the above rule set). We believe that a simple rule-based pre- or post-processing procedure would solve some of the anomalies originating in stylistic and annotational differences between CGN and OVIS, possibly resulting in better classifier performance; however, testing this empirically lies out of the scope of our study.

In total 3,818 word hypotheses are classified as disfluent by MBL in the full OVIS data set, and 3,449 by RI. Words that are classified as belonging to a disfluent chunk include all types of words, ranging from specific slot values to filled pauses and unintelligible material. Table 6.3 displays the 20 tokens that are most frequently classified as disfluent by the two learners. The most remarkable finding, illustrated by the two lists in the table, is that many content words are classified as disfluent by both learners. These include words such as ‘yes’, ‘no’, and various slot values (‘today’, ‘tomorrow’, ‘Friday’, ‘nine’, etc.) that were previously found to contribute much to the classification of the SI components (cf. Section 4.4.2, as well as the findings of Chapter 5).

It is noteworthy that regardless of the classification bias, the words that are most frequently classified disfluent by MBL and RI are very much the same. This may indicate that the information in the training data as presented to the learners is more determining for classification results of disfluent phenomena than the kind of algorithmic approach employed.

In this experiment RI induced a rule set that contains 50 rules and 247 conditions about the IN-DISFL class. In general, these rules contain up to 6 conditions, have a small coverage, and many counter-examples.

6.1.2.3 Incorporating the disfluency filtered BOW in SI

The next step in our investigation is to use the disfluency-filtered BOW in the SI task. Note that the set-up of this experiment is identical to the general experimental set-up in the SI module (i.e., 10-fold CV combined with WPS), the only difference being that instead of the full BOW we use the disfluency-filtered BOW. For each component the class label is as optimised in the class partitioning experiments of the previous chapter.

We preprocess the material as follows. The words classified as part of a disfluent chunk are removed from the BOW representation by switching the ‘1’ indicating presence in the BOW to ‘0’. Since 172 words are always classified as disfluent by MBL, and 170 by RI (which is again a very similar result), these are always removed from the BOW representation

MBL		RI	
<i>frequency</i>	<i>item</i>	<i>frequency</i>	<i>item</i>
1057	uh	1057	uh
688	GARBLED	688	GARBLED
420	no	374	no
391	yes	346	yes
129	uhm	166	to_PP
120	to_PP	129	uhm
76	want	52	from
53	from	46	tomorrow
51	tomorrow	40	I
42	I	27	want
29	today	23	today
26	o'clock	20	nope
26	at	16	day_after_tomorrow
26	nope	16	at
20	day_after_tomorrow	16	zero
20	zero	13	Friday
19	Friday	13	not
18	Saturday	12	two
18	nine	12	nine
15	two	10	yes_EMPH

Table 6.3: The 20 items most frequently classified as disfluent by MBL and RI. ‘EMPH’ indicates the emphatic word form (i.e., *jawel*).

(i.e., are always ‘0’), thus the net effect of disfluency filtering is that the BOW vector is reduced from 759 to 587 bits in the MBL classification experiment, and to 589 bits in the RI learning experiment. Words that are always classified as part of a disfluent chunk include filled pauses such as ‘uh’ and ‘uhm’, as well as various low frequency words such as small station names (e.g., ‘jirrenveen’ and ‘zwaagwesteinde’) or discourse markers (e.g., ‘jazeker’; *sure*, or ‘welnee’; *of course not*).

6.1.2.4 Effects of disfluency filtering on SI

The results of the learning experiments are shown in Table 6.4, where scores are given for each SI component. Compared to the scores gained in Chapter 5, here marked in *italics*, we see that the impact of this filtering technique produces only a small improvement or no improvement over the scores of Chapter 5. Scores that indicate improvement are printed in **bold**.

In general, RI gains more from disfluency filtering than MBL: RI yields (seemingly) improved scores on all four components, whereas MBL only on the SLOT component. This suggests that filtering might indeed remove words that negatively influence the SI task, so that better patterning can be discovered in the data (primarily by RI), but discarding these items does not practically influence the target task of SI, since none of the improvements

are statistically significant.

The obtained results show that filtering disfluencies does not have a significantly positive effect on the SI task, although it does not deteriorate the results either. We believe that the differences between the training and the test material contribute to suboptimal filtering. For example, the ASR n -best path output never contains fragmented words, whereas these form one of the main disfluency types in the training material. We assume that the OVIS lattice paths, on which we attempted disfluency detection, contain distortions on a much larger scale than proper disfluencies (that are difficult or impossible to recognise by ASR), and that these distortions are ungrammatical in a different way than disfluencies (cf. [Zechner and Waibel 1998, Palmer and Ostendorf 2001]).

For completeness' sake, we run two experiments for learning the FWD PR component by RI, since in feature partitioning the isolated ASR group showed (insignificantly) better performance than learning on all features (see Section 5.4.2). We tested both the full (but disfluency-filtered) feature vector, as well as the isolated disfluency-filtered ASR group in learning FWD PR. The results on the FWD PR component, displayed by the line printed in small size, however show that using all features outperforms the isolated ASR group by RI, signalling that non-significant score differences do not hold generalisation power as the effect of feature partitioning observed in the previous chapter is not exhibited in the current task. (Note that although for learning the FWD PR component by MBL the isolated DM group showed better performance than learning on all features (see Section 5.4.2), in the current experiment on FWD PR we employed all features in order to measure the effect of disfluency filtering.)

6.2 Filtering non-heads of syntactic chunks

6.2.1 Training on CGN data

Our second method for filtering the word graph is to discard everything from the lattice except the words that act as syntactic chunk heads. For this end we use the memory-based shallow parser of [Canisius 2004] developed for Dutch, and automatically assign a syntactic analysis to each token in the unfolded paths of the word graphs. Since the OVIS corpus does not provide syntactic annotations, it is not an option to train the parser on it, and again we use the CGN corpus for training. The memory-based shallow parser is reported to attain 83.9% precision, 85.9% recall, and 84.9 F-score on tagging and chunking Dutch spontaneous speech in the CGN corpus material.

Since the ASR output is material claimed to be ill-formed in different ways than spontaneous speech (see above), we expect this discrepancy to again lead to suboptimal chunk head filtering, the inferences of which are not known on the end task.

6.2.2 Testing on OVIS data

The preprocessing step here includes unfolding the paths in each ASR lattice and capitalising station names to reduce potential parsing errors. Capitalisation is carried out on the basis of a fixed-length list of occurring station names; it could also be done internally in the ASR.

Algorithm	Component	Metric			
		acc	pre	rec	F
MBL	<i>TRA</i>	<i>86.6</i>	<i>94.3</i>	<i>89.3</i>	<i>91.7</i>
		<i>0.7</i>	<i>1.5</i>	<i>1.0</i>	<i>0.7</i>
		87.0	94.1	89.3	91.6
		0.9	1.4	0.9	1.0
		<i>83.5</i>	<i>90.7</i>	<i>84.9</i>	<i>87.7</i>
	<i>SLOT</i>	<i>2.2</i>	<i>1.8</i>	<i>2.6</i>	<i>2.0</i>
		84.1	92.0	85.4	88.6
		2.6	2.2	2.6	2.1
		<i>68.6</i>	<i>67.1</i>	<i>53.5</i>	<i>59.4</i>
		<i>2.6</i>	<i>4.4</i>	<i>5.4</i>	<i>4.1</i>
	<i>FWD PR</i>	66.5	62.4	57.1	59.4
		2.5	4.4	6.4	3.8
		<i>92.3</i>	<i>93.7</i>	<i>88.1</i>	<i>90.8</i>
		<i>0.9</i>	<i>2.5</i>	<i>2.1</i>	<i>1.2</i>
		91.9	94.4	86.4	90.2
	<i>BWD PR</i>	<i>2.0</i>	<i>3.2</i>	<i>2.3</i>	<i>2.3</i>
		<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
		<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>
		86.1	92.4	88.9	90.6
		1.7	1.4	1.6	1.3
RI	<i>TRA</i>	<i>82.6</i>	<i>88.4</i>	<i>82.9</i>	<i>85.5</i>
		<i>2.4</i>	<i>4.4</i>	<i>3.8</i>	<i>2.8</i>
		83.4	90.2	84.6	87.3
		2.8	1.8	3.2	2.2
		<i>65.0</i>	<i>57.9</i>	<i>72.1</i>	<i>62.6</i>
	<i>SLOT</i>	<i>2.0</i>	<i>3.9</i>	<i>12.2</i>	<i>5.2</i>
		65.8	58.3	76.0	65.3
		3.1	3.8	12.4	5.3
		65.2	59.7	66.2	61.4
		<i>2.8</i>	<i>5.4</i>	<i>14.8</i>	<i>5.4</i>
	<i>FWD PR</i>	<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
		<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>
		91.0	93.1	85.6	89.0
		<i>1.5</i>	<i>3.3</i>	<i>5.7</i>	<i>2.6</i>
		<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
	<i>BWD PR</i>	<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>
		91.0	93.1	85.6	89.0
		<i>1.5</i>	<i>3.3</i>	<i>5.7</i>	<i>2.6</i>
		<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
		<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>

Table 6.4: Performances by MBL and RI when filtering the user bag-of-words from **disfluencies**, using the optimised class labelling as estimated in the class partitioning experiments (cf. Chapter 5). Performance is averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of each SI component. The italicised lines show the results of the experiment with unfiltered features (cf. Chapter 5).

The shallow parser analyses the paths syntactically: each token is assigned a complex label that encodes three types of information about the word: its POS tag, its syntactic chunk tag, and a tag marking whether it is the head in its syntactic chunk. Below is an illustration of the parsing output of the path that is correctly recognised from the first user turn of our example dialogue. Syntactic chunks are indicated by square brackets; head words are marked as HD, for the rest of the syntactic labels in CGN we refer to [Van der Wouden et al. 2002]. Note that syntactic chunks are generally quite small in the data; many of them consist of only one word that is then automatically labelled as the chunk head. In this example only the token ‘volgende’ (*next*) is classified as chunk non-head.

```
{ [NP-SU-1 ik/VNW1-HD] [SMAIN-1 moet/WW1-HD] [NP volgende/WW1 week/N1-HD]
  [NP dinsdag/N5-HD] { PNP [PP van/VZ1-HD] [NP Schiphol/N5-HD] } { PNP [PP
naar/VZ1-HD] [NP Nijmegen/N5-HD] } }
```

6.2.3 Incorporating the chunk non-head filtered BOW in SI

Drawing on the material obtained from shallow parsing, each word marked as the head of a syntactic chunk is retained in the BOW vector, whereas all non-heads are filtered out from it by switching their corresponding feature values from ‘1’ to ‘0’. In total 573 words are always classified as chunk heads in the OVIS material, reducing the chunk head filtered BOW to 573 bits. The new BOW representations for each user turn are used together with the other word graph, DM, and prosodic features to classify user input in terms of SI components by MBL and RI.

Note that the set-up of the following experiments is again the same as the general set-up in the SI module (10-fold CV combined with WPS), but instead of the full BOW we now use the chunk non-head filtered BOW. We learn each SI component as optimised in the information partitioning experiments in the previous chapter.

6.2.4 Effects of chunk non-head filtering on SI

The results of the experiments are shown in Table 6.5 per SI component. We can establish that in general no improvement follows chunk non-head filtering. We see that filtering non-head words leads to somewhat higher scores than no filtering only on the SLOT component for both learners, and additionally on FWD PR for RI. However, the results show that filtering out words that are not chunk heads does not lead to significantly better or worse scores compared to no filtering on the BOW. There might be a number of reasons for this.

One obvious limitation of this experiment is again the marked difference between training and test data that probably produces suboptimal results. Another factor might be that our shallow parser produces small syntactic chunks, so that relatively few words are actually filtered out from the BOW vector. At the same time, directly incorporating possibly imperfect parsing results may lead to the accumulation of error in the end task. We

Algorithm	Component	Metric			
		acc	pre	rec	F
MBL	TRA	<i>86.6</i>	<i>94.3</i>	<i>89.3</i>	<i>91.7</i>
		<i>0.7</i>	<i>1.5</i>	<i>1.0</i>	<i>0.7</i>
		86.2	94.0	88.4	91.1
		1.3	1.5	1.2	1.5
	SLOT	<i>83.5</i>	<i>90.7</i>	<i>84.9</i>	<i>87.7</i>
		<i>2.2</i>	<i>1.8</i>	<i>2.6</i>	<i>2.0</i>
		83.2	91.1	84.7	87.8
		2.1	1.2	2.8	1.7
	FWD PR	<i>68.6</i>	<i>67.1</i>	<i>53.5</i>	<i>59.4</i>
		<i>2.6</i>	<i>4.4</i>	<i>5.4</i>	<i>4.1</i>
		66.3	62.6	55.4	58.5
		2.4	4.1	6.7	3.4
	BWD PR	<i>92.3</i>	<i>93.7</i>	<i>88.1</i>	<i>90.8</i>
		<i>0.9</i>	<i>2.5</i>	<i>2.1</i>	<i>1.2</i>
		91.6	94.2	85.8	89.8
		1.8	2.5	2.8	2.3
RI	TRA	<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
		<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>
		83.4	90.7	86.4	88.5
		2.4	2.7	2.0	2.0
	SLOT	<i>82.6</i>	<i>88.4</i>	<i>82.9</i>	<i>85.5</i>
		<i>2.4</i>	<i>4.4</i>	<i>3.8</i>	<i>2.8</i>
		83.2	89.7	84.3	86.9
		1.8	2.6	3.0	1.8
	FWD PR	<i>65.0</i>	<i>57.9</i>	<i>72.1</i>	<i>62.6</i>
		<i>2.0</i>	<i>3.9</i>	<i>12.2</i>	<i>5.2</i>
		64.5	57.2	74.4	63.9
		2.8	3.5	13.0	5.5
	BWD PR	63.6	56.7	72.7	63.0
		4.3	4.4	11.2	4.0
		<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
		<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>
		90.5	93.4	84.1	88.4
		1.7	3.2	3.2	1.9

Table 6.5: Performances by MBL and RI when filtering out **chunk non-head words** from the user bag-of-words. Classification is carried out in terms of the optimised class labelling as estimated in the information partitioning experiments (cf. Chapter 5). Performance is averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of each SI component. The italicised lines show the results of the experiment with unfiltered features (cf. Chapter 5).

believe that these issues could be solved to some extent by postprocessing the data, or by combining simple heuristics (e.g., word-based rules) with classification.

However, the fact that the occurring improvements are statistically insignificant also suggests some positive outcome of this experiment; namely, that we have evidence about syntactic chunk heads being predictive about pragmatic-semantic information (as defined by the SI components), since removing all non-heads does not significantly harm performance on the SI task.

6.3 Filtering on the basis of word frequency

Our third filtering method draws on word frequency in the BOWs. In the study of [Rotaru and Litman 2003] feature subset selection is carried out on the basis of the information gain of a feature as measured in classification of various human-machine dialogue phenomena. In our study however, classification by the memory-based learner is not always IG-based (since IG is only one of the four possible feature weighting metrics in the four distance functions, cf. Section 3.1.1). We opted for filtering the BOW on the basis of word frequency in the speech recognition output, which is established by general counts in our material.

6.3.1 Incorporating the frequency filtered BOW in SI

Our method is to retain information corresponding to the presence or absence of the 15 most frequent words in the user's BOW, and filter out all other words from the feature vector. We again measure the impact of frequency filtering directly on the SI task. Although frequency filtering as implemented in this study draws on simple counts in the OVIS corpus, an easily implementable ML-based alternative would be to filter on the basis of automatically assigned feature weights.

The left column of Table 6.6 shows the 15 most frequent words in the ASR output of the OVIS corpus, whereas the right column of the same table shows the first 15 words ranked according to their IG as summed over the IG weight calculations of MBL in the complex experiment. Note that the two lists largely overlap, indicating that a high IG assigned in the complex experiment often corresponded to a high frequency word. Note that the assigned IG weights were not used by MBL in the complex experiment except for one partition, in combination with the MVDM metric, since the feature weighting metrics optimised by WPS turned out to be different (cf. Section 4.4.3). For completeness' sake we reproduce the 100 most frequent words in Table 1 of the Appendix.

6.3.2 Effects of frequency-based filtering on SI

The results of classifying SI components by incorporating the frequency-filtered BOW in the feature vector are shown in Table 6.7. The scores indicate that frequency-based filtering has in general a negative effect on classification performance, deteriorating it on all but one SI component: the FWD PR. All scores are significantly worse than in the non-filtered experiments on the $p < 0.01$ level, (t -scores in order of appearance in the table: 5.2, 5.1, 6.9 for MBL, and 4.0, 4.3, 3.3 for RI).

FREQUENCY-BASED TOP 15				IG-BASED TOP 15			
<i>frequency</i>	<i>item</i>	<i>ov. MBL</i>	<i>ov. RI</i>	<i>IG</i>	<i>item</i>	<i>ov. MBL</i>	<i>ov. RI</i>
4886	to_PP	+	*	.570	to_PP	+	*
4113	no	+	*	.511	no	+	*
3897	from	+	*	.465	from	+	*
3284	want	+	*	.418	yes	+	*
3267	I	+	*	.381	o'clock	+	
3254	o'clock	+		.241	at	+	*
2670	at	+	*	.197	I	+	*
2206	yes	+	*	.194	want	+	*
1627	uh	+	*	.145	nope	+	*
1605	that			.134	arrive		
1485	thank			.121	ten		
1444	on			.119	from.EMPH		
1322	not		*	.109	thank		
1305	travel			.104	the_PN		
1293	ten			.097	not		*

Table 6.6: The 15 most frequent, respectively highest-IG-ranked words of the recognised input in the OVIS corpus. IG is summarised over the IG weights assigned by MBL in the complex experiment (Chapter 4). ‘PN’ indicates a proper noun form (i.e., *Den*), ‘EMPH’ indicates the emphatic word form (i.e., *vanuit*), ‘+’ indicates overlap of the word if classified disfluent by MBL, ‘*’ by RI (cf. Table 6.3).

Concerning the FWD PR component, we see that RI seems to improve using the frequency-filtered BOW, but this is an insignificant increase. Note that it occurs using all features, whereas when only ASR features are used (printed in small font), the performance seems to drop, although not significantly. However, it is remarkable that without information about the prompting context and the prosody of the input (i.e., the DM and PROS features) forward-pointing problems seem to be less predictable. Since neither the decrease nor the increase are significant, we have no clear evidence about the effect of filtering in combination with feature partitioning; however, as the performance of the other learner, MBL, does not show a significant decrease or increase in the score on this component, we may conjecture that frequency-based word graph filtering is not harmful for the FWD PR component, but is harmful for all other components.

On the other hand, this seems to indicate that the presence or absence of the most frequently recognised words is predictive enough about future problems. Other SI components need the information supplied by all other recognised tokens as well in order to predict future problems; apparently, many cues are lost when the representation of the recognition hypothesis is reduced from 759 to 15 bits.

It is noteworthy that we can also observe a lot of overlap (marked by ‘+’ for MBL and ‘*’ for RI) between the lists of the most frequent, respectively highest-IG-ranked words in Table 6.6, and the list of words that are most often classified disfluent (see Table 6.3). This means that the majority in the small set of words that are kept in the frequency-filtered BOW are the ones that are often filtered out from the still quite large set of disfluency-

Algorithm	Component	Metric			
		acc	pre	rec	F
MBL	<i>TRA</i>	<i>86.6</i>	<i>94.3</i>	<i>89.3</i>	<i>91.7</i>
		<i>0.7</i>	<i>1.5</i>	<i>1.0</i>	<i>0.7</i>
		85.8	93.6	88.0	90.8
	<i>SLOT</i>	1.0	1.3	0.9	0.9
		<i>83.5</i>	<i>90.7</i>	<i>84.9</i>	<i>87.7</i>
		<i>2.2</i>	<i>1.8</i>	<i>2.6</i>	<i>2.0</i>
		79.0	87.5	78.4	82.7
		2.2	2.8	3.1	2.4
	<i>FWD PR</i>	<i>68.6</i>	<i>67.1</i>	<i>53.5</i>	<i>59.4</i>
		<i>2.6</i>	<i>4.4</i>	<i>5.4</i>	<i>4.1</i>
		67.1	64.1	54.4	58.7
	<i>BWD PR</i>	1.8	4.3	3.9	3.4
		<i>92.3</i>	<i>93.7</i>	<i>88.1</i>	<i>90.8</i>
		<i>0.9</i>	<i>2.5</i>	<i>2.1</i>	<i>1.2</i>
		89.8	93.2	82.3	87.4
		1.3	3.0	2.2	2.0
RI	<i>TRA</i>	<i>86.0</i>	<i>92.0</i>	<i>89.1</i>	<i>90.5</i>
		<i>1.7</i>	<i>2.0</i>	<i>1.5</i>	<i>1.5</i>
		83.1	90.5	86.5	88.4
	<i>SLOT</i>	1.0	1.3	1.3	0.7
		<i>82.6</i>	<i>88.4</i>	<i>82.9</i>	<i>85.5</i>
		<i>2.4</i>	<i>4.4</i>	<i>3.8</i>	<i>2.8</i>
		76.6	88.1	72.9	79.7
		2.8	3.6	4.2	2.1
	<i>FWD PR</i>	<i>65.0</i>	<i>57.9</i>	<i>72.1</i>	<i>62.6</i>
		<i>2.0</i>	<i>3.9</i>	<i>12.2</i>	<i>5.2</i>
		66.9	60.4	74.2	65.5
	<i>BWD PR</i>	3.8	6.0	12.7	5.3
		62.1	55.0	65.5	59.3
		4.4	5.9	12.2	7.4
		<i>90.5</i>	<i>92.4</i>	<i>85.1</i>	<i>88.5</i>
		<i>1.4</i>	<i>3.5</i>	<i>3.9</i>	<i>1.5</i>
		87.9	90.1	81.7	85.2
		1.6	6.5	6.8	2.7

Table 6.7: Performances by MBL and RI on the SI components (in terms of the optimised class labelling as estimated in the class partitioning experiments, cf. Chapter 5) when the BOW is filtered using the **15 most frequent words** in OVIS. Performance is averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score measured on the classification of each SI component. The italicised lines show the results of the experiment with unfiltered features (cf. Chapter 5).

filtered BOW. Observe that for example a filled pause is among the 15 most frequent words in the corpus, whereas it is one of the most frequent disfluencies in general.

In fact, this may mean that the experiments using the frequency-filtered BOW are to some extent complementary to the experiments using the disfluency-filtered BOW. Although we indeed see that frequency-based filtering aggravates performance on three SI components, whereas disfluency filtering does not, the obtained results do not clearly support this hypothesis that would need further investigation. We conjecture that information about the SI components is probably carried not by the individual words, but the co-occurrence of these.

6.4 Discussion

In effect, by the three filtering approaches we investigated the feasibility of incorporating higher-level information in the SI task. In particular, by disfluency filtering we aimed at blocking information in the SI module that could be incorrect or superfluous in terms of syntactic and/or lexical criteria. By chunk non-head filtering we aimed at promoting information judged syntactically more dominant. By frequency filtering we aimed at restricting information in the SI module to words that are supposed to carry information of the highest value in the given SDS's domain.

Two out of the three filtering approaches, disfluency filtering and chunk non-head filtering showed an encouraging, but statistically insignificant positive effect on the SI task, whereas the results for the third method, frequency-based filtering, showed primarily negative effects on the SI task. We assume that the investigated filtering approaches are difficult NLP classification tasks in themselves; in particular, we have found that they exhibit sensitivity to differences between training and test data.

We have to emphasise that the evaluation of the results obtained by testing the filtering methods directly on the SI task needs to be taken with certain precaution: since the BOW representation is utterly shallow, the effects of filtering may not reach an optimal effect on the SI task, for example because the frequency or the syntactic context of a word is not represented by the BOW. At the same time, incorporating incorrect classification results (since e.g. neither automatic disfluency detection nor shallow parsing is perfect) in new classification tasks may yield cumulative error, which deteriorates learning performance.

6.5 Evaluation

The question arises what performance could be expected from the learners on the SI task, if they had access to perfectly recognised material. Therefore, we will run additional experiments in which the BOW is created on the basis of the transcribed user input. This emulates the situation where classification of SI components is based on perfectly recognised and 'perfectly disfluency filtered' (i.e., left out from the annotation), respectively automatically chunk non-head filtered and frequency filtered user input. The obtained results will provide us with the topline scores that could be ultimately attained by the these filtering techniques.

<i>frequency</i>	<i>item</i>
1018	to_PP
887	from
789	I
770	no
744	want
521	yes
500	o'clock
390	at
211	travel
179	that
174	the_PN
140	not
136	groningen
135	you_POL
133	from_EMPH

Table 6.8: The 15 most frequent items in the transcribed utterances of the OVIS corpus. ‘PN’ indicates a proper noun form (i.e., *Den*), ‘EMPH’ indicates an emphatic word form (i.e., *vanuit*), ‘POL’ indicates a polite word form (i.e., *u*).

6.5.1 Analysis of transcribed utterances in OVIS

Table 6.8 shows the 15 most frequent words in the transcribed OVIS corpus. Note that the most frequent transcribed words overlap largely with the most frequently recognised words (Table 6.6). The tokens that are not contained in the list of most frequent transcribed words are ‘uh’, ‘thank’, ‘on’, and ‘ten’. The ranking of words is somewhat different between the two lists, suggesting that there are differences in the magnitude of the frequency of certain words in the word graphs, respectively in transcribed utterances. For example, ‘no’ is found to be hypothesised very frequently (4,113 times, cf. Table 6.6), and ‘yes’ much less frequently (2,206 times, cf. Table 6.6), whereas in reality these two words occur on a more similar scale (770 vs 521 times, cf. Table 6.8). Recall that we have found that the N TRA is more difficult for our classifiers to detect in the input (cf. Section 5.2.3) – it might be that the ASR of this system has problems with recognising the words corresponding to the N TRA (which are mainly ‘no’, ‘not’, ‘don’t’; *nee*, *geen*, *niet*). It may be inferred from the statistics that the ASR hypothesises the occurrence of ‘no’ to a rather substantial extent, which might indicate that it has difficulties with recognising ‘no’, and/or with recognising the material that might surround ‘no’ (e.g., a correction of a slot value).

6.5.2 Data preprocessing

Since disfluencies are not transcribed in the real user utterances, the transcribed utterances can be directly used to emulate ‘perfect’ disfluency filtering. The resulting BOW has 559 bits, which thus comprise the full lexicon based on transcribed utterances. For emulating automatic chunk non-head filtering on the ‘perfectly recognised’ (i.e., transcribed)

Algorithm	Component	Disfluency filt		Non-head filt		Frequency filt	
		acc	F	acc	F	acc	F
MBL	TRA	89.1	93.3	88.6	92.9	88.9	93.1
		1.3	0.8	1.0	0.8	1.1	0.6
	SLOT	86.8	90.8	85.6	90.2	83.6	88.7
		1.6	1.2	1.8	1.4	1.0	0.7
	FWD PR	68.5	61.1	68.4	60.9	67.8	60.4
		2.1	3.7	3.2	3.3	3.3	4.4
	BWD PR	93.1	91.8	92.8	91.4	93.0	91.6
		1.1	1.1	1.1	1.1	0.9	1.1
RI	TRA	88.6	92.6	88.0	91.9	88.0	92.4
		1.9	1.6	1.9	1.6	1.4	1.2
	SLOT	88.7	91.6	85.6	89.8	80.9	85.7
		1.3	1.2	1.3	1.1	1.0	1.1
	FWD PR	65.8	64.9	66.1	64.3	66.3	61.2
		3.2	3.4	3.0	3.5	3.4	6.7
	BWD PR	92.1	90.4	92.8	91.4	91.7	90.0
		1.6	1.5	1.0	1.3	2.2	2.7

Table 6.9: Topline scores in terms of accuracy and F-score produced by MBL and RI using the transcribed utterance in optimal class- and feature design, averaged over 10-fold CV experiments. The column **Disfluency filt** shows performance based on all the features where the BOW represents the transcribed user utterance from which disfluencies were removed by the transcribers. The column **Non-head filt** shows performance based on the chunk non-head filtered transcribed user utterance. The column **Frequency filt** shows performance based on the frequency filtered transcribed user utterance.

input, we again use the shallow parser and analyse the transcribed sentences syntactically, discarding tokens that are non-heads. The resulting BOW has 490 bits, since 69 tokens are always classified by the shallow parser as chunk non-heads. For emulating topline frequency filtering, we discard tokens from the transcribed strings that are not in the 15 most frequent transcribed words. The resulting BOW in this experiment has therefore 15 bits.

The set-up of the topline experiments is identical to the general set-up throughout this study (10-fold CV combined with WPS for parameter optimisation). In all three topline experiments we use the filtered BOW as well as our other features to classify user input in terms of the SI components optimised for algorithm parameters and class labels. We display the scores of the three experiments with both learners in Table 6.9.

6.5.3 Evaluating filtering on the basis of topline experiments

Since disfluencies are not transcribed in this material, it is not possible to measure the effect of the disfluency filtering method in the topline experiments, thus the figures in the

Disfluency filt column of Table 6.9 can be regarded as an illustration of the performance of the learners when both perfect disfluency filtering and perfectly recognised input is assumed.

The only comparison we can make is between the scores in column *Disfluency filt* to those in column *Non-head filt*. These scores are practically the same for all SI components, indicating that, as was the case on the recognised material, filtering out non-chunk heads from the transcribed user input seems not to have impact on classification performance.

Frequency filtering however seems not to deteriorate performance on the transcribed material to the same extent as on the word graph material: we see that when the learners have to draw on the perfectly recognised and frequency filtered user words, performance decreases only on the SLOT component (recall that on the recognised material it decreased also on TRA and BWD PR). We assume that the SLOT component requires knowledge about more than only the top 15 words to keep up learner performance (about findings on the role of specific words in classifying the SLOT component see Section 5.3.3). The TRA and the BWD PR components do not suffer from frequency filtering; it might be that in the transcribed input more consistent co-occurrences can be found between the presence or absence of the top 15 words and TRA and BWD PR classes, than in those of the recognised input.

Comparing the topline scores attained on filtering the transcribed user utterance to those obtained using the filtered word graphs (reported earlier in this chapter, e.g., the corresponding non-italicised lines in Table 6.4), we can establish the following. For the majority of tasks both learners produce an improvement of a few points of F-score when using the actual words uttered by the user instead of using the recognised words. Significant improvements are displayed in Table 6.10. Both for MBL and RI the filtering methods for the TRA and SLOT components perform significantly better on the transcribed word string than on the word graphs, but not for the FWD PR component. It is difficult to see a trend for the BWD PR component.

In sum, the topline experiments suggest that all three filtering methods could in principle produce higher scores on classifying task-related acts and slots in the user input, given an improved speech recognition output. However, the fact that none of the filtering methods seems to improve performance on the FWD PR component implies that it is as efficient to consider the noisy word graph material for identifying user turns that are sources of communication problems than to consider perfectly understood user words.

Given that there are no significant differences across the filtering methods in the topline experiments (i.e., across the rows of Table 6.9), and the significant, but relatively small improvements of the topline scores over the recognised material, it may be hypothesised that the room for improvement available for filtering recognised material is rather small. We conjecture that our SI approach is robust to noise in itself already, so that removing noise from the feature vector cannot substantially ameliorate ML performance on classifying the SI components.

6.5.4 Evaluating SI on the basis of the topline experiments

Comparing the topline SI scores (on manually disfluency filtered material, cf. the *Disfluency filt* column of Table 6.9) with our best scores obtained in Chapter 5 (see Table

Algorithm	Component	Statistical significance					
		Disfluency filt		Non-head filt		Frequency filt	
MBL	TRA	$t = 3.7$	$p < .01$	$t = 3.5$	$p < .01$	$t = 6.2$	$p < .01$
	SLOT	$t = 6.3$	$p < .01$	$t = 4.5$	$p < .01$	$t = 8.9$	$p < .01$
	FWD PR	-		-		-	
	BWD PR	-		-		$t = 6.8$	$p < .01$
RI	TRA	$t = 5.0$	$p < .01$	$t = 4.5$	$p < .01$	$t = 8.6$	$p < .01$
	SLOT	$t = 4.7$	$p < .01$	$t = 6.1$	$p < .01$	$t = 10.8$	$p < .01$
	FWD PR	-		-		-	
	BWD PR	-		$t = 5.0$	$p < .01$	$t = 3.1$	$p < .05$

Table 6.10: Statistical significances in a paired t -test of learner performance on filtering transcribed user input improving over filtering recognised input.

5.12), two trends seem to emerge. Most importantly, we again see that the performance of MBL and RI is practically identical for all SI components, corresponding to our findings on recognised material. The extent to which the SI components can be learnt likewise corresponds to our previous findings, namely, that we are able to learn task-related acts most successfully, followed by the backward-pointing problem and the slot components, and that forward-pointing problems are very hard to predict on the basis of the user utterance in shallow context.

Comparing these two tables, it can be observed that the classification performance of the SI module is a few points below the topline scores in terms of F-score. In the column showing statistical significance in Table 6.11 we present an evaluation of the improved performance on transcribed user input over recognised user input (without filtering). The figures show that when they have access to transcribed material, both learners improve significantly on the TRA, SLOT, and BWD PR components, but not on the FWD PR component.

The rightmost column of Table 6.11 shows how much reduction in errors is produced on the level of F-score by the learners, given that they have access to transcribed instead of recognised words in the user input. Both MBL and RI show the largest error reduction for the SLOT component (MBL: 25%, RI: 42%), followed by the TRA and the BWD PR components. No significant improvement occurs on the FWD PR by any of the learners. These figures indicate that with perfect speech recognition and the described shallow interpretation approach it would be possible to further improve the results of the SI module, especially for detecting which slots are being filled by the user; however, on predicting forward-pointing problems our shallow interpretation approach is capable of reaching the performance that would be attained based on perfectly recognised user input.

6.6 Summary

In this chapter we described experimental results of three filtering techniques. Two of these techniques, namely disfluency filtering and chunk head filtering, can be seen as ML-based

Algorithm	Component	Statistical significance		Error reduction
MBL	TRA	$t = 4.6$	$p < .01$	19%
	SLOT	$t = 5.0$	$p < .01$	25%
	FWD PR	-		4%
	BWD PR	$t = 2.6$	$p < .05$	10%
RI	TRA	$t = 2.4$	$p < .05$	22%
	SLOT	$t = 8.4$	$p < .01$	42%
	FWD PR	-		6%
	BWD PR	$t = 2.7$	$p < .05$	16%

Table 6.11: Statistical significances and reduction levels of learner performance on unfiltered transcribed user input improving over unfiltered recognised user input.

approaches that process n -best paths in ASR output, filtering out certain words from the paths automatically. The third approach, frequency filtering, in effect bears resemblance to weighting-based word filtering, although it draws on simple counts in our corpus. The filtering methods provide our two classifiers with a subset of the BOW features.

In three series of experiments we trained MBL and RI to learn the SI components (as optimised in Chapter 5) on the basis of all the features where the word graph features were systematically filtered with one of the three methods. We observed that stylistic, as well as annotation differences between the training and the test data interfered with the correct filtering of disfluencies, respectively chunk non-heads. Our empirical results show that the filtering techniques in this set-up have modest impact on the SI task. Moreover, when ‘noisiness’ is radically reduced such as in frequency-based filtering, many pieces of information are lost, which seems to deteriorate performance on the SI task.

We subsequently conducted topline experiments in which the filtering methods were applied to the transcribed user words instead of the recognised ones, on the basis of which the SI was learnt. The outcomes of these experiments signal that filtering that employs higher-level information only leads to a small improvement on shallow interpretation even when perfectly recognised user words serve as input. Based on the experiments conducted on transcribed user turns, our conclusion is that the proposed shallow approach is robust with respect to noise in the data, thus filtering this noise cannot have much further impact on classification performance on the four SI components.

We also concluded that if all noise could be eliminated (as simulated in the topline experiments), this would be beneficial for the TRA, BWD PR, and especially the SLOT component, but for detecting forward-pointing problems our SI module already produces the same performance.

A valuable outcome of the experimental series described in this chapter is that across the performances of the memory-based learner and the rule induction learner again similar trends can be found, reflecting our previous finding that if task design and algorithm settings are optimised, MBL and RI are likely to show similar scaling of classification performance on our NLU tasks.

Chapter 7

Conclusions

The research issues of our study were the following:

- (i) determine the extent to which supervised machine learning techniques can be used for shallow interpretation of user turns in spoken dialogue systems,
- (ii) explore whether the complex learning task of four-level shallow interpretation can be optimised by decomposing it to subtasks, and
- (iii) explore whether filtering noise from spoken user input on the basis of higher-level linguistic information leads to improved learning performance on the shallow interpretation task.

Corresponding to (i), we conducted a case study by training MBL and RI on a corpus of Dutch dialogues with a SDS in the travel domain. Pragmatic-semantic information was extracted from spoken user input in a shallow way, i.e., drawing on unsophisticated features, in terms of a four-level interpretation. Our investigation yielded the following results.

When the four interpretation levels are combined in a complex class label involving task-related acts, filled slot types, forward-pointing problems, and backward-pointing problems, machine learning performance is not optimal, although significantly better than the score of an informed baseline strategy that draws on the most recently posed system prompt. The component for which both MBL and RI achieve the highest F-score is the identification of task-related acts (89.0, respectively 80.9 F-score), the results obtained on backward-pointing problems are quite similar to these (87.7, respectively 78.6 F-score). Classification results of filled slot types are somewhat lower (83.4, respectively 75.7 F-score), whereas both algorithms attain the lowest score in this task on the forward-pointing problem component (55.4, respectively 55.6 F-score). Prediction of task-related acts, slots, and backward-pointing problems is done significantly better by MBL than by RI. It is difficult to classify the complex label, since some (aspects of) components are harder to predict than others (e.g., whether users accept system errors, or whether the user input is going to cause communication problems), which may aggravate learner performance in general.

Therefore, corresponding to (ii), in Chapter 5 we developed a method for improving the module's performance by means of partitioning the information presented to the learning algorithms. For each SI component we conducted two consecutive series of experiments with both MBL and RI. In the first series we performed class partitioning, in the second series feature partitioning. The large-scale experimental matrix provided a possibility to compare MBL and RI to a considerable extent: all experiments were conducted under identical conditions, but the class labels, as well as the feature groups, were systematically varied.

Based on the outcomes of this matrix we established that it is useful to optimise the task composition, i.e., class label, in the module. In particular, we have found that class partitioning has a substantial, positive influence on the scores produced by both classifiers for all SI components. The best scores produced by the SI module are displayed in Table 5.12 (page 103). The classification success of the various SI components exhibits the same trend for both learners as found in Chapter 4: the highest performance is produced on learning the task-related acts (MBL: 91.7 F-score, RI: 90.5), followed by the detection of backward-pointing problems (MBL: 90.8 F-score, RI: 88.5). The results for the remaining components — filled slot types (MBL: 87.7, RI: 85.5) and forward-pointing problems (MBL: 59.4, RI: 62.6) — are lower. The improvements gained by optimising the class label of the learning tasks account for substantial error reductions, especially for RI, reducing up to 50% of classification errors in terms of F-score. It is remarkable that in class partitioning MBL and RI produce statistically identical top performances concerning all four SI components.

We observed that the various groups of information source contributed to a different extent to the classification tasks, where primarily features of the speech recogniser output provided most information to classifying a component. Selecting a particular feature group did not improve our scores: in general, using information coming from all available sources turned out to be best for extracting pragmatic-semantic information from spoken user turns.

Corresponding to our third goal (iii), in Chapter 6 our aim was to block those pieces of information from the optimised learning algorithms that the literature supposes to negatively effect language processing: we designed three, primarily machine learning-based methods to automatically filter the speech recogniser's output from disfluent words, from syntactically less dominant words, and from words that do not frequently occur in the recognition hypotheses. We observed that disfluency filtering and chunk non-head filtering had a positive but statistically insignificant impact on the SI task, whereas frequency-based filtering deteriorated classification performance.

The experimental outcomes obtained on transcribed user input (emulating the situation in which speech recognition is perfect and clean of disfluencies) further show that the effect of noise filtering is overall minor on our module. We conjecture that filtering cannot substantially improve learning performance on the SI task, probably because in our optimised experimental set-up the classifiers can internally cope with noise and superfluous information by appropriate weighting or selection of features. At the same time, we hypothesise that the impact of noise filtering would become more prominent with better stylistic match of training and test data. The topline experiments show that, given perfect speech recognition, performance of the SI module could further improve.

In this study it was our general aim to create a SI module that is robust in several respects:

- it deploys adequate, generalisable machine learning techniques,
- it copes with noise in spoken input and in the shallow representation of such input, and
- it accounts for multi-layeredness in the input content.

We conceptualised shallow interpretation as a straightforward classification task. Our approach led to similar scale and performance tendencies in machine learning experiments with differently biased classifiers, suggesting that the method can be generally implemented by supervised learning techniques. The tested classification methods proved to be adequate for the given task, since even the most difficult goal, the complex prediction of all four levels of the SI task, both MBL and RI produced significant improvements above baseline learning techniques. Besides the ability to provide a shallow pragmatic-semantic interpretation of the user turn, we were supplied content-related knowledge about the human-machine interaction process represented by our corpus. The employed machine learning techniques produced satisfactory or good results of practical value. All utilised information was easily obtained from the SDS, making the established approach attractive for NLU applications.

We have dealt with noisiness on several levels. The learning algorithms drew on approximative, erroneous, and hypothetical measurements in the data, since the features extracted from the spoken user input represent a large number of possibly imperfect measurements and hypotheses of the SDS itself. Our experiments show that in the proposed set-up classifiers can tolerate noisiness, since these interpret user utterances well above the baseline even when faced with recognition errors. Moreover, when filtering techniques that incorporate higher-level linguistic information were applied to the noisy speech recogniser output, the cleaned data were not shown to yield significantly better results in learning to extract pragmatic-semantic information from user turns.

At the same time, we were also able to learn which factors of human-machine interaction can be identified as problem sources. Certain types of user input, mainly meta-replies such as accepting errors that the system has made, or providing non-standard answers independent of the dialogue context, as well as certain properties of the SDS were found to cause miscommunication, and thus to easily introduce problems into the interaction with a SDS. In the examined system such properties included the speech recogniser's difficulty in processing negative user answers, the design of the opening system question, as well as aspects of the dialogue manager's prompting strategy that does not facilitate recovery from misunderstandings between the human and the machine.

In order to account for multi-layeredness in the input content, we extracted information related to pragmatic and semantic levels of the user utterance: on the pragmatic level task-related acts, potential problem source, and problem awareness were detected, on the semantic level the supplied information unit types were identified. We came to the conclusion that such complex information is best to extract when the component combination is optimised, which is possible to determine via a matrix of machine learning experiments.

The outcomes of the experiments furthermore taught us that for shallow interpretation of user utterances it is optimal to co-learn at most two SI components and, in case two components are combined, one of them should be the task-related act component. The results also suggested that the backward-pointing problem component could be merged in the task-related act component of the SI module.

The goal of this study was to develop and test a general method to be implemented in a shallow interpretation module of a SDS. We reported on the performance of this module given a particular data set collected from interactions with a particular system. Our research is a case study, the details of which are not intended to serve as general findings about spoken human-machine interaction, but as findings about the described shallow approach. Its main finding is that drawing on unsophisticated — thus potentially noisy — features that characterise the dialogue situation (system prompts, full output of the speech recogniser, acoustic-prosodic measurements of the speech signal), and by performing automatic optimisation of the formulated machine learning task at least in terms of class labels and algorithm parameter settings, it is possible to extract sophisticated information of practical pragmatic-semantic value from the spoken user input with robust performance.

We assume that the method of automatically training classifiers on pragmatic-semantic tasks can be generally applied to data collected from dialogue systems: the proposed class design should be portable to other types of task-oriented dialogues that employ a closed set of domain concepts. We hypothesise that with the use of additional — shallowly formulated — engineering techniques (e.g., more features, their automatic combination and selection, more structured representation of speech recognition output, combination of information sources and learning methods) the module's current performance would improve. In the future we plan to apply this approach to different dialogue data to gain more results, and to prove that the method is suitable for other domains as well.

Bibliography

- [Aberdeen et al. 2001] Aberdeen, J., C. Doran, L. Damianos, S. Bayer, and L. Hirschman. 2001. Finding errors automatically in semantically tagged dialogues. In *Proc. of the First International Conference on Human Language Technology Research*.
- [Aha 1998] Aha, D. 1998. Feature weighting for lazy learning algorithms. In H. Liu and H. Motoda (Eds.), *Feature Extraction, Construction and Selection*. Kluwer.
- [Aha et al. 1991] Aha, D., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.
- [Allen 1995] Allen, J. 1995. *Natural Language Understanding*. Benjamin/Cummings.
- [Allen et al. 1996] Allen, J., B. Miller, E. Ringger, and T. Sikorski. 1996. Robust understanding in a dialogue system. In *Proc. of ACL*.
- [Allen and Core 1997] Allen, J., and M. Core. 1997. Draft of DAMSL: Dialog act markup in several layers. Unpublished manuscript.
- [Aust et al. 1995] Aust, H., M. Oerder, F. Seide, and V. Steinbiss. 1995. The Philips automatic train timetable information system. *Speech Communication* 17:249–262.
- [Balentine et al. 1999] Balentine, B., D. Morgan, and S. Meisel. 1999. *How to build a speech recognition application*. Enterprise Integration Group.
- [Banko and Brill 2001] Banko, M., and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proc. of ACL*.
- [Barkhuysen et al. 2005] Barkhuysen, P., E. Krahmer, and M. Swerts. 2005. Problem detection in human-machine interactions based on facial expressions of users. *Special Issue of Speech Communication on Error Handling*. To appear.
- [Batliner et al. 2003] Batliner, A., C. Hacker, S. Steidl, J. Haas, and E. Nöth. 2003. User states, user strategies, and system performance: how to match the one with the other. In *Proc. of ISCA workshop on Error handling in spoken dialogue systems*, 5–10.
- [Batliner et al. 1999] Batliner, A., E. Nöth, J. Buckow, R. Huber, V. Warnke, and H. Niemann. 1999. Prosodic feature evaluation: Brute force or well designed? In *Proceedings of the 14th Int. Congress of Phonetic Sciences*, Vol. 3, 2315–2318.

- [Bear et al. 1992] Bear, J., J. Dowding, and E. Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Meeting of the Association for Computational Linguistics*, 56–63.
- [Beun 1989] Beun, R. 1989. *The recognition of declarative questions in information dialogues*. PhD thesis, Katholieke Universiteit Brabant, Netherlands.
- [Bonnema et al. 1997] Bonnema, R., R. Bod, and R. Scha. 1997. A DOP model for semantic interpretation. In *Proc. of ACL/EACL*, 159–167.
- [Boros et al. 1996] Boros, M., W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann. 1996. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proc. of ICSLP*.
- [Bos et al. 1996] Bos, J., B. Gambäck, C. Lieske, Y. Mori, M. Pinkal, and K. Worm. 1996. Compositional semantics in Verbmobil. In *Proc. of COLING*, 131–136.
- [Boves et al. 1995] Boves, L., J. Landsbergen, R. Scha, and G. van Noord. 1995. Language and speech technology. Research plan 1995–1997. Technical report, Nijmegen University. Available from: <http://grid.let.rug.nl:4321/>.
- [Boves et al. 1996] Boves, L., J. Landsbergen, R. Scha, and G. van Noord. 1996. Language and speech technology. Progress report 1995–1996. Technical report, Nijmegen University. Available from: <http://grid.let.rug.nl:4321/progprep/>.
- [Buchholz 2002] Buchholz, S. 2002. *Memory-based grammatical relation finding*. PhD thesis, Tilburg University, Netherlands.
- [Bunt 1989] Bunt, H. 1989. Information dialogues as communicative action in relation to partner modeling and information processing. In M. Taylor, F. Néel, and D. Bouwhuis (Eds.), *The Structure of Multimodal Dialogue*. Amsterdam: North-Holland Elsevier.
- [Bunt 2000] Bunt, H. 2000. Dynamic interpretation and dialogue theory. In M. Taylor, F. Néel, and D. Bouwhuis (Eds.), *The Structure of Multimodal Dialogue, Volume 2*. Amsterdam: John Benjamins.
- [Bunt 2001] Bunt, H. 2001. Dialogue pragmatics and context specification. In H. Bunt and W. Black (Eds.), *Abduction, Belief and Context in dialogue. Studies in Computational Pragmatics*. John Benjamins.
- [Bunt and Black 2000] Bunt, H., and W. Black. 2000. The ABC of computational pragmatics. In H. Bunt and W. Black (Eds.), *Computational Pragmatics, Abduction, Belief and Context*, Studies in Computational Pragmatics, 1–46. Amsterdam: John Benjamins.
- [Busser 1998] Busser, B. 1998. TreeTalk-D: A machine learning approach to Dutch word pronunciation. In *Proc. of TSD Conference*, 3–8.
- [Canisius 2004] Canisius, S. 2004. Memory-based shallow parsing of spoken Dutch. Master's thesis, Maastricht University, Netherlands.

- [Cattoni et al. 2001] Cattoni, R., M. Federico, and A. Lavie. 2001. Robust analysis of spoken input combining statistical and knowledge-based information sources. In *Proc. of Automatic Speech Recognition and Understanding Conference*.
- [Cettolo et al. 1996] Cettolo, M., A. Corazza, and R. D. Mori. 1996. A mixed approach to speech understanding. In *Proceedings of ICSLP*.
- [Charniak and Johnson 2001] Charniak, E., and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL*, 118–126.
- [Choi et al. 1999] Choi, W., J. Cho, and J. Sea. 1999. Analysis system of speech acts and discourse structures using maximum entropy model. In *Proc. of ACL*.
- [Clark and Niblett 1989] Clark, P., and T. Niblett. 1989. The CN2 rule induction algorithm. *Machine Learning* 3:261–284.
- [Cohen 1995] Cohen, W. 1995. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*.
- [Cost and Salzberg 1993] Cost, S., and S. Salzberg. 1993. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning* 10:57–78.
- [Cover and Hart 1967] Cover, T., and P. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory* 13:21–27.
- [Curran and Osborne 2002] Curran, J., and M. Osborne. 2002. A very very large corpus doesn't always yield reliable estimates. In *Proc. of CONNL*.
- [Daelemans 1995] Daelemans, W. 1995. Memory-based lexical acquisition and processing. In P. Steffens (Ed.), *Machine Translation and the Lexicon*, Springer Lecture Notes in Artificial Intelligence, 85–98. Springer.
- [Daelemans and Hoste 2002] Daelemans, W., and V. Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *Third International Conference on Language Resources and Evaluation (LREC 2002)*, 755–760.
- [Daelemans et al. 1997] Daelemans, W., A. van den Bosch, and T. Weijters. 1997. Empirical learning of natural language processing tasks. In M. van Someren and G. Widmer (Eds.), *Machine Learning: ECML-97, Lecture Notes in Artificial Intelligence 1224*, 337–344. Springer.
- [Daelemans et al. 1999] Daelemans, W., A. van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning* 34:11–41.
- [Daelemans et al. 2003] Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 2003. TiMBL: Tilburg Memory Based Learner, version 5.0, Reference guide. ILK Technical Report 03-13, Tilburg University. Available from <http://ilk.uvt.nl>.

- [DiEugenio and Glass 2004] DiEugenio, B., and M. Glass. 2004. The Kappa statistic: A second look. *Computational Linguistics* 30(1):95–101.
- [Duchateau et al. 2003] Duchateau, J., T. Laureys, K. Demuynck, and P. Wambacq. 2003. Handling disfluencies in spontaneous language models. In *Computational Linguistics in the Netherlands 2002. Selected Papers from the Thirteenth CLIN Meeting*, 39–50. Rodopi.
- [Dybkjaer and Bernsen 2000] Dybkjaer, L., and O. Bernsen. 2000. The MATE markup framework. In L. Dybkjaer, K. Hasida, and D. Traum (Eds.), *Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue*, 19–28. San Francisco: Morgan Kaufmann Publishers 2000.
- [Eisele and Ziegler-Eisele 2002] Eisele, A., and D. Ziegler-Eisele. 2002. Towards a road map on human language technology: Natural language processing. In *Report on the Second ELSNET Roadmap Workshop. Version 2*.
- [Eisner 1996] Eisner, J. 1996. An empirical comparison of probability models for dependency grammar. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania.
- [Eklund 2004] Eklund, R. 2004. *Disfluency in Swedish human-human and human-machine travel booking dialogues*. PhD thesis, Department of Computer and Information Science, Linköping University, Sweden.
- [Eklund and Shriberg 1998] Eklund, R., and E. Shriberg. 1998. Crosslinguistic disfluency modeling: A comparative analysis of Swedish and American English human-human and human-machine dialogs. In *Proc. Int. Conf. on Spoken language processing*.
- [Feinman 1997] Feinman, A. 1997. Message types in goal-oriented discourse. Seminar in Artificial Intelligence. Available from: <http://www.cs.brandeis.edu/~afeinman/papers/message.html>.
- [Finke et al. 1998] Finke, M., M. Lapata, A. Lavie, L. Levin, L. M. Tomokiyo, T. Polzin, K. Ries, A. Waibel, and K. Zechner. 1998. CLARITY: Inferring discourse structure from speech. In *AAAI'98 Spring Symposium Series*.
- [Fix and Hodges 1951] Fix, E., and J. Hodges. 1951. Discriminatory analysis, non-parametric discrimination: consistency properties. Technical report, USAF School of Aviation and Medicine, Randolph Air Field, TX.
- [Flycht-Eriksson 1999] Flycht-Eriksson, A. 1999. A survey of knowledge sources in dialogue systems. *Electronic Transactions on Artificial Intelligence (ETAI). Special Issue on Intelligent Dialogue Systems*.
- [Fürnkranz 1997] Fürnkranz, J. 1997. Pruning algorithms for rule learning. *Machine Learning* 27(2):139–171.
- [Fürnkranz and Widmer 1994] Fürnkranz, J., and G. Widmer. 1994. Incremental reduced error pruning. In *Proceedings of ICML*, 244–251.

- [Gazdar et al. 1985] Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized phrase structure grammar*. Harvard University Press.
- [Gibbon et al. 1997] Gibbon, D., R. Moore, and R. Winski. 1997. *Handbook of Standards and Resources for Spoken Language Systems*. De Gruyter.
- [Goldberg et al. 2003] Goldberg, J., M. Ostendorf, and K. Kirchhoff. 2003. The impact of response wording in error correcting subdialogues. In *Proc. of ISCA workshop on Error handling in spoken dialogue systems*, 101–106.
- [Grünwald et al. 1998] Grünwald, P., P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. 1998. Minimum encoding approaches for predictive modeling. In *Proc. 14th Int. Conf. on Uncertainty in AI (UAI)*, 183–192.
- [Hacioglu et al. 2004] Hacioglu, K., S. Pradhan, W. Ward, J. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of CONLL*.
- [He and Young 2004] He, Y., and S. Young. 2004. Robustness issues in a data-driven spoken language understanding system. In *Proc. of NAACL'04*.
- [Heeman 1998] Heeman, P. 1998. POS tagging versus classes in language modeling. In *Proc. of Sixth Workshop on Very Large Corpora*, 179–187.
- [Heeman 1999] Heeman, P. 1999. Modeling speech repairs and intonational phrasing to improve speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.
- [Heeman and Allen 1994] Heeman, P., and J. Allen. 1994. Detecting and correcting speech repairs. In *Proc. of ACL*, 295–302.
- [Hermes 1988] Hermes, D. 1988. Measurement of pitch by subharmonic summation. *Journal of the Acoustical Society of America* 83:257–264.
- [Hindle 1983] Hindle, D. 1983. Deterministic parsing of syntactic nonfluencies. In *Proc. of ACL*, 123–128.
- [Hirose 1995] Hirose, K. 1995. Disambiguating recognition results by prosodic features. In Y. Sagisaka, N. Campbell, and N. Higuchi (Eds.), *Computing Prosody*, 327–342. Springer.
- [Hirschberg et al. 1999] Hirschberg, J., D. Litman, and M. Swerts. 1999. Prosodic cues to recognition errors. In *Proceedings of the 1999 International Workshop on Automatic Speech Recognition and Understanding*, 349–352, Keystone, CO.
- [Hirschberg et al. 2000] Hirschberg, J., D. Litman, and M. Swerts. 2000. Generalizing prosodic prediction of speech recognition errors. In *Proceedings of the 6th International Conference of Spoken Language Processing (ICSLP-2000)*, Beijing, China.
- [Hirschberg et al. 2001] Hirschberg, J., D. Litman, and M. Swerts. 2001. Identifying user corrections automatically in spoken dialogue systems. In *Proc. of NAACL*.

- [Hirschberg et al. 2004] Hirschberg, J., D. Litman, and M. Swerts. 2004. Prosodic and other cues to speech recognition failures. *Speech Communication* 43:155–175.
- [Hockey et al. 1997] Hockey, B., D. Rossen-Knill, B. Spejewski, M. Stone, and S. Isard. 1997. Can you predict responses to yes/no questions? yes, no, and stuff. In *Proc. of Eurospeech*.
- [Jackson and Moulinier 2002] Jackson, P., and I. Moulinier. 2002. *Natural Language Processing for Online Applications: Text Retrieval, Extraction & Categorization*. John Benjamins Publishing.
- [Jurafsky and Martin 2000] Jurafsky, D., and J. Martin. 2000. *Speech and Language Processing: An Introduction to natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall.
- [Jurafsky et al. 1997] Jurafsky, D., E. Shriberg, and D. Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado, Institute of Cognitive Science.
- [Jurafsky et al. 1996] Jurafsky, D., E. Shriberg, B. Fox, and T. Curl. 1996. Lexical, prosodic, and syntactic cues for dialog acts. In *Proc. of ACL/COLING-98 Workshop on Discourse Relations and Discourse Markers*.
- [Kamm et al. 1998] Kamm, C., D. Litman, and M. Walker. 1998. From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems. In *Proc. of ICSLP*.
- [Keizer 2003] Keizer, S. 2003. *Reasoning under uncertainty in natural language dialogue using Bayesian networks*. PhD thesis, University of Twente, Netherlands.
- [Kiefer et al. 2000] Kiefer, B., H. Krieger, and M. Nederhof. 2000. Efficient and robust parsing of word hypotheses graphs. In W. Wahlster (Ed.), *VerbMobil: Foundations of Speech-to-Speech Translation*, 428–437. Springer.
- [Koeling 2002] Koeling, R. 2002. *Dialogue-based disambiguation: Using Dialogue Status to Improve Speech Understanding*. PhD thesis, Groningen University, Netherlands.
- [Kohavi and John 1997] Kohavi, R., and G. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2):273–324.
- [Krahmer et al. 1999] Krahmer, E., M. Swerts, M. Theune, and M. Weegels. 1999. Error spotting in human-machine interactions. In *Proc. of Eurospeech*, 1423–1426.
- [Krahmer et al. 2001a] Krahmer, E., M. Swerts, M. Theune, and M. Weegels. 2001a. The dual of denial: Two uses of disconfirmations in dialogue and their prosodic correlates. *Speech Communication* 36(1):133–145.
- [Krahmer et al. 2001b] Krahmer, E., M. Swerts, M. Theune, and M. Weegels. 2001b. Error detection in spoken human-machine interaction. *International Journal of Speech Technology* 4:19–30.

- [Lendvai 2003] Lendvai, P. 2003. Learning to identify fragmented words in spoken discourse. In *Proc. of EACL Student Research Workshop.*, 25–32.
- [Lendvai and Maruster 2003] Lendvai, P., and L. Maruster. 2003. Process discovery for evaluating dialogue strategies. In *Proc. of ISCA Workshop on Error Handling in Spoken Dialogue Systems*, 119–122.
- [Lendvai et al. 2003] Lendvai, P., A. van den Bosch, and E. Krahmer. 2003. Memory-based disfluency chunking. In *Proc. of Disfluency in Spontaneous Speech Workshop (DISS'03)*, 63–66.
- [Lendvai et al. 2002a] Lendvai, P., A. Van den Bosch, E. Krahmer, and M. Swerts. 2002a. Improving machine-learned detection of miscommunications in human-machine dialogues through informed data splitting. In *Proc. ESSLLI Workshop on Machine Learning Approaches in Computational Linguistics*.
- [Lendvai et al. 2002b] Lendvai, P., A. Van den Bosch, E. Krahmer, and M. Swerts. 2002b. Multi-feature error detection in spoken dialogue systems. In *Proc. Computational Linguistics in the Netherlands (CLIN '01)*. Rodopi Amsterdam.
- [Levelt 1989] Levelt, W. J. M. 1989. *Speaking: From intention to articulation*. Cambridge, MA: The MIT Press.
- [Levin et al. 2000] Levin, E., R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialogue strategies. *IEEE Transactions on Speech and Audio Processing* 8(1):11–23.
- [Levow 1998] Levow, G. 1998. Characterizing and recognizing spoken corrections in human-computer dialogue. In *Proc. of COLING-ACL*.
- [Litman et al. 2000] Litman, D., J. Hirschberg, and M. Swerts. 2000. Predicting automatic speech recognition performance using prosodic cues. In *Proc. of NAACL*.
- [Litman et al. 2001] Litman, D., J. Hirschberg, and M. Swerts. 2001. Predicting user reactions to system errors. In *Proc. of EACL*, 362–369.
- [Litman and Pan 1999] Litman, D., and S. Pan. 1999. Predicting and adapting to poor speech recognition in a spoken dialogue system. In *Proceedings of the 7th International Conference of User Modelling*.
- [Litman et al. 1999] Litman, D., M. Walker, and M. Kearns. 1999. Automatic detection of poor speech recognition at the dialogue level. In *Proc. of ACL*, 309–316.
- [Manning and Schutze 1999] Manning, C., and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.
- [Màrquez 2000] Màrquez, L. 2000. Machine learning and natural language processing. Technical report, Polytechnic University of Catalonia, Barcelona, Spain.

- [Maynard et al. 2002] Maynard, D., V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. 2002. Architectural elements of language engineering robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*.
- [McKelvie 1998] McKelvie, D. 1998. The syntax of disfluency in spontaneous spoken language. Technical report, Human Communication Research Centre, University of Edinburgh, U.K.
- [Michalski et al. 1986] Michalski, R. S., J. G. Carbonell, and T. M. Mitchell (Eds.). 1986. *Machine learning: An artificial intelligence approach*. Vol. II. San Mateo, CA: Morgan Kaufmann.
- [Mitchell 1997] Mitchell, T. 1997. *Machine learning*. McGraw Hill.
- [Mitkov 2003] Mitkov, R. (Ed.). 2003. *The Oxford Handbook of Computational Linguistics*. Oxford University Press.
- [Montague 1974] Montague, R. 1974. *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press.
- [Nakano et al. 1999] Nakano, M., N. Miyazaki, J. Hirasawa, K. Dohsaka, and T. Kawabata. 1999. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *Proc. of ACL*.
- [Nakatani and Hirschberg 1994] Nakatani, C., and J. Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustical Society of America* 95(3):1603–1616.
- [Oostdijk 2002] Oostdijk, N. 2002. *The Design of the Spoken Dutch Corpus*. In: *New Frontiers of Corpus Research*. P. Peters, P. Collins and A. Smith (eds.), pages 105–112. Amsterdam: Rodopi.
- [Oviatt 1995] Oviatt, S. 1995. Predicting spoken disfluencies during human-computer interaction. *Computer Speech Language* 9:19–36.
- [Oviatt et al. 1996] Oviatt, S., G. Levow, M. MacEachern, and K. Kuhn. 1996. Modelling hyperarticulate speech during human-computer error resolution. In *Proc. of ICSLP*.
- [Oviatt et al. 1998] Oviatt, S., M. McEachern, and G. Levow. 1998. Predicting hyperarticulate speech during human-computer error resolution. *Speech Communication* 24:87–110.
- [Palmer and Ostendorf 2001] Palmer, D., and M. Ostendorf. 2001. Improving information extraction by modeling errors in speech recogniser output. In *Proc. of the First International Conference on Human Language Technology Research*.
- [Plauche and Shriberg 1999] Plauche, M., and E. Shriberg. 1999. Data-driven subclassification of disfluent repetitions based on prosodic features. In *Proc. International Congress of Phonetic Sciences*, Vol. 2, 1513–1516.

- [Pollard and Sag 1987] Pollard, C., and I. Sag. 1987. *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. Vol. 13 of *CSLI Lecture Notes*. Center for the Study of Language and Information.
- [Pollard and Sag 1994] Pollard, C., and I. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.
- [Popescu-Belis et al. 2003] Popescu-Belis, A., A. Clark, M. Georgescu, M. Starlander, and S. Zufferey. 2003. A thematic bibliography on dialogue processing. Technical report, ISSCO/TIM/ETI, Université de Genève, Switzerland.
- [Privat 2003] Privat, R. 2003. Age effect on ASR performances: which dialogue recommendations for adaptive strategies. In *Proc. of ISCA workshop on Error handling in spoken dialogue systems*, 59–64.
- [Provost et al. 1999] Provost, F., D. Jensen, and T. Oates. 1999. Efficient progressive sampling. In *Knowledge Discovery and Data Mining*, 23–32.
- [Qu et al. 1997] Qu, Y., B. DiEugenio, A. Lavie, L. Levin, and C. Rose. 1997. Minimizing cumulative error in discourse context. In *Dialogue Processing in Spoken Language Systems: Revised Papers from ECAI-96 Workshop*. Springer.
- [Quinlan 1986] Quinlan, J. 1986. Induction of Decision Trees. *Machine Learning* 1:81–206.
- [Quinlan 1993] Quinlan, J. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- [Rayner and Hockey 2003] Rayner, M., and B. Hockey. 2003. Transparent combination of rule-based and data-driven approaches in a speech understanding architecture. In *Proc. of EACL*.
- [Reithinger and Engel 2000] Reithinger, N., and R. Engel. 2000. Robust content extraction for translation and dialog processing. In W. Wahlster (Ed.), *VerbMobil: Foundations of Speech-to-Speech Translation*, 428–437. Springer.
- [Reithinger et al. 1996] Reithinger, N., R. Engel, M. Kipp, and M. Klesen. 1996. Predicting dialogue acts for a speech-to-speech translation system. In *Proc. of ICSLP*.
- [Reithinger and Maier 1995] Reithinger, N., and E. Maier. 1995. Utilizing statistical dialogue act processing in verbmobil. In *Proc. of ACL*.
- [Ringger and Allen 1997] Ringger, E., and J. Allen. 1997. Robust error correction of continuous speech recognition. In *Proc. of ESCA-NATO Workshop on Robust Speech Recognition for Unknown Communication Channels*.
- [Rissanen 1978] Rissanen, J. 1978. Modeling by shortest data description. *Automatica* 14:465–471.
- [Rotaru and Litman 2003] Rotaru, M., and D. Litman. 2003. Exceptionality and natural language learning. In *Proc. of CONNL*.

- [Samuel et al. 1998a] Samuel, K., S. Carberry, and K. Vijay-Shanker. 1998a. Computing dialogue acts from features with transformation-based learning. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 90–97.
- [Samuel et al. 1998b] Samuel, K., S. Carberry, and K. Vijay-Shanker. 1998b. Dialogue act tagging with transformation-based learning. In *Proceedings of COLING/ACL*, 1150–1156.
- [Shannon and Weaver 1949] Shannon, C., and W. Weaver. 1949. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.
- [Shimojima et al. 1998] Shimojima, A., H. Koiso, M. Swerts, and Y. Katagiri. 1998. An informational analysis of echoic responses in dialogue. In *Proc. 20th Annual Conference of the Cognitive Science Society*, 951–956, Madison, WI, USA.
- [Shriberg 1994] Shriberg, E. 1994. *Preliminaries to a theory of speech disfluencies*. PhD thesis, University of California at Berkeley, U.S.A.
- [Shriberg et al. 1998] Shriberg, E., R. Bates, A. Stolcke, P. Taylor, D. Jurafsky, K. Ries, N. Coccaro, R. Martin, M. Meteer, and C. V. Ess-Dykema. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and Speech. Special double issue on prosody and conversation*. 41(3-4):439–487.
- [Shriberg et al. 2001] Shriberg, E., A. Stolcke, and D. Baron. 2001. Can prosody aid the automatic processing of multi-party meetings? Evidence from predicting punctuation, disfluencies, and overlapping speech. In *Proc. of ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, 139–146.
- [Shriberg et al. 1992] Shriberg, E., E. Wade, and P. Price. 1992. Human-machine problem solving using spoken language systems: Factors affecting performance and user satisfaction. In *Proceedings of the DARPA Speech and Natural Language Workshop*, 49–54.
- [Spilker et al. 2001] Spilker, J., A. Batliner, and E. Nöth. 2001. How to Repair Speech Repairs in an End-to-End System. In *Proc. ISCA Workshop on Disfluency in Spontaneous Speech*, 73–76.
- [Spilker et al. 2000] Spilker, J., M. Klarner, and G. Görz. 2000. Processing self-corrections in a speech-to-speech system. In W. Wahlster (Ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*, 428–437. Springer.
- [Stolcke et al. 2000] Stolcke, A., K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics* 26:339–373.
- [Stolcke et al. 1998a] Stolcke, A., E. Shriberg, R. Bates, N. Coccaro, P. Taylor, D. Jurafsky, R. Martin, M. Meteer, K. Ries, and C. V. Ess-Dykema. 1998a. Dialog act modeling for conversational speech. In *Papers from the 1998 AAAI Spring Symposium*, 98–105.

- [Stolcke et al. 1998b] Stolcke, A., E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. 1998b. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proc. Int. Conf. on Spoken Language Processing*, Vol. 5, 2247–2250.
- [Streit 2003] Streit, M. 2003. Context-dependent error handling by a three-layered processing model. In *Proc. of ISCA workshop on Error handling in spoken dialogue systems*, 147–152.
- [Strik et al. 1997] Strik, H., A. Russel, H. van den Heuvel, C. Cucchiaroni, and L. Boves. 1997. A spoken dialog system for the Dutch public transport information service. *Int. Journal of Speech Technology* 2(2):121–131.
- [Swerts et al. 1998] Swerts, M., H. Koiso, A. Shimojima, and Y. Katagiri. 1998. On different functions of repetitive utterances. In *Proc. of ICSLP*, 1287–1290.
- [Swerts et al. 2000] Swerts, M., D. Litman, and J. Hirschberg. 2000. Corrections in spoken dialogue systems. In *Proceedings of ICSLP*, 615–618.
- [Taylor et al. 1998] Taylor, P., S. King, S. Isard, and H. Wright. 1998. Intonation and dialogue context as constraints for speech recognition. *Language and Speech. Special double issue on prosody and conversation*. 41(3-4):493–512.
- [Theune 2003] Theune, M. 2003. From monologue to dialogue: Natural language generation in OVIS. In *Proc. of AAAI 2003 Spring Symposium on Natural Language Generation in Written and Spoken Dialogue*.
- [Traum 1994] Traum, D. 1994. A computational theory of grounding in natural language conversation. Technical report, Department of Computer Science, University of Rochester, U.S.A.
- [Traum 2003] Traum, D. 2003. Semantics and pragmatics of questions and answers for dialogue agents. In *Proc. of International Workshop on Computational Semantics*, 380–394.
- [Traum and Heeman 1997] Traum, D., and P. Heeman. 1997. Utterance units in spoken dialogue. In E. Maier, M. Mast, and S. LuperFoy (Eds.), *Dialogue Processing in Spoken Language Systems*, 125–140. Springer.
- [Traum and Larsson 2003] Traum, D., and S. Larsson. 2003. The information state approach to dialogue management. In J. van Kuppevelt and R. Smith (Eds.), *Current and New Directions in Discourse and Dialogue*. Kluwer.
- [Uszkoreit 2002] Uszkoreit, H. 2002. New chances for deep linguistic processing. In *Proc. of COLING*.
- [Van den Berg et al. 1994] Van den Berg, M., R. Bod, and R. Scha. 1994. A corpus-based approach to semantic interpretation. In *Proc. of the Ninth Amsterdam Colloquium*.

- [Van den Bosch 2004] Van den Bosch, A. 2004. Wrapped progressive sampling search for optimizing learning algorithm parameters. In *Proc. of 16th Belgium-Netherlands Conference on Artificial Intelligence*, 219–228.
- [Van den Bosch and Buchholz 2002] Van den Bosch, A., and S. Buchholz. 2002. Shallow parsing on the basis of words only: A case study. In *Proc. of ACL*, 433–440.
- [Van den Bosch et al. 2001] Van den Bosch, A., E. Krahmer, and M. Swerts. 2001. Detecting problematic turns in human-machine interactions: Rule-induction versus memory-based learning approaches. In *Proc. of ACL*, 499–506.
- [Van der Wouden et al. 2002] Van der Wouden, T., H. Hoekstra, M. Moortgat, B. Renmans, and I. Schuurman. 2002. Syntactic analysis in the spoken dutch corpus. In *Proc. of LREC*, 768–773.
- [Van Noord et al. 1996] Van Noord, G., G. Bouma, R. Koeling, and M. Nederhof. 1996. Conventional natural language processing in the NWO priority programme on Language and speech technology. Technical report, Nr. 28, NWO Priority Programme Language and Speech Technology, October 1996 Deliverables.
- [Van Noord et al. 1999] Van Noord, G., G. Bouma, R. Koeling, and M. Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Journal of Natural Language Engineering* 5(1):45–93.
- [Van Rijsbergen 1979] Van Rijsbergen, C. 1979. *Information Retrieval*. London: Butterworth.
- [Veldhuijzen van Zanten 1996] Veldhuijzen van Zanten, G. 1996. Semantics of update expressions. Technical report, IPO, Eindhoven University.
- [Veldhuijzen van Zanten 1998] Veldhuijzen van Zanten, G. 1998. Adaptive mixed-initiative dialogue management. In *Proc. of IVTTA*, 65–70.
- [Veldhuijzen van Zanten et al. 1999] Veldhuijzen van Zanten, G., G. Bouma, K. Sima'an, G. van Noord, and R. Bonnema. 1999. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In *Computational Linguistics in the Netherlands 1998*, 213–229. Rodopi Amsterdam.
- [Vogten and Gigi 1998] Vogten, L., and E. Gigi. 1998. GIPOS (Graphical Interactive Processing of Speech). Technical report, Institute for Perception Research, Eindhoven University, Netherlands.
- [Wahlster 2000] Wahlster, W. (Ed.). 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer.
- [Walker et al. 2000a] Walker, M., I. Langkilde, J. Wright, A. Gorin, and D. Litman. 2000a. Learning to predict problematic situations in a spoken dialogue system: Experiment with how may i help you? In *Proc. of ACL*.

- [Walker et al. 2000b] Walker, M., J. Wright, and I. Langkilde. 2000b. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proceedings of the International Conference on Machine Learning*.
- [Weber and Wermter 1996] Weber, V., and S. Wermter. 1996. Towards learning semantics of spontaneous dialog utterances in a hybrid framework. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, 87–101. Springer.
- [Weegels 2000] Weegels, M. 2000. Users' conceptions of voice-operated information services. *International Journal of Speech Technology* 3:75–82.
- [Weintraub et al. 1996] Weintraub, M., K. Taussig, K. Hunnicke-Smith, and A. Snodgrass. 1996. Effect of speaking style on LVCSR performance. In *Proc. of ICSLP*, 16–19.
- [Weiss and Kulikowski 1991] Weiss, S., and C. Kulikowski. 1991. *Computer Systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann.
- [Wermter et al. 1996] Wermter, S., E. Riloff, and G. Scheler (Eds.). 1996. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer.
- [Zavrel et al. 2000] Zavrel, J., S. Degroeve, A. Kool, W. Daelemans, and K. Jokinen. 2000. Diverse classifiers for NLP disambiguation tasks. Comparison, optimization, combination, and evolution. In *TWLT 18. Learning to Behave. CEvoLE 2.*, 201–221.
- [Zechner and Waibel 1998] Zechner, K., and A. Waibel. 1998. Using chunk based partial parsing of spontaneous speech in unrestricted domains for reducing word error rate in speech recognition. In *Proc. of ACL*.

Appendix

Turn	Utterance
S1	van welk station naar welk station wilt u reizen
U1	van amsterdam
S2	wanneer wilt u vanuit almelo naar amsterdam centraal station reizen
U2	fout
S3	het spijt me ik heb u niet begrepen wilt u nog eens zeggen wanneer u vanuit almelo naar amsterdam centraal station wilt reizen
U3	terug klopt niet
S4	het spijt me maar ik heb u weer niet begrepen wilt u nog eens zeggen wanneer u vanuit almelo naar amsterdam centraal station wilt reizen
U4	ik wil van amsterdam naar emmen

Figure 1: Original transcription of example turns from dialogue nr. 002/005 from the OVIS corpus.

FREQUENCY-BASED TOP 100							
<i>freq</i>	<i>item</i>	<i>freq</i>	<i>item</i>	<i>freq</i>	<i>item</i>	<i>freq</i>	<i>item</i>
4886	naar	680	half	399	leeuwarden	245	jawel
4113	nee	637	amsterdam	393	zeven	244	vijf
3897	van	634	twee	387	voor	239	s-morgens
3284	wil	618	dertig	376	dertien	238	zaterdag
3267	ik	606	groningen	369	februari	235	nul
3254	uur	597	acht	368	uhm	235	maandag
2670	om	593	nee_dank_u	358	verbinding	233	en
2206	ja	588	maar	354	delft	231	wilde
1627	uh	557	is	339	vrijdag	219	woerden
1605	dat	542	het	333	in	219	arnhem
1485	dank	521	utrecht	324	zwolle	218	assen
1444	op	496	elf	313	dinsdag	216	ongeveer
1322	niet	487	klopt	309	voorburch	216	nijmegen
1305	reizen	485	haag	303	vier	215	achttien
1293	tien	482	twaaif	303	rond	213	alphen
1241	negen	478	station	295	donderdag	211	heerlen
1210	den	478	drie	290	januari	210	deventer
1201	de	473	een	285	maastricht	205	vijftien
1100	neen	437	centraal	278	twintig	198	trein
1062	u	428	leiden	274	aankomen	189	heb
989	graag	427	vandaag	271	ook	186	woensdag
948	vertrekken	427	hoorn	269	breda	186	juni
914	vanuit	423	rotterdam	266	zes	178	negentien
875	morgen	422	vanaf	262	zondag	173	veertien
754	xxx	404	haarlem	254	tussen	173	te

Table 1: The 100 most frequent words of the recognised input in the OVIS corpus.

Turn	Utterance	Ver. type
S1	goedemorgen dit is het automatische informatiesysteem van openbaar vervoer reisinformatie dit systeem geeft uitsluitend informatie over treinreizen van ns-station naar ns-station van welk station naar welk station wilt u reizen	
U1	ik wil van voorschoten naar den haag centraal station	
S2	wanneer wilt u vanuit voorschoten naar utrecht centraal station reizen	IMPL
U2	ik wil naar den haag centraal station	
S3	wanneer wilt u vanuit voorschoten naar den haag centraal station reizen	IMPL
U3	overmorgen	
S4	hoe laat wilt u overmorgen reizen	IMPL
U4	ergens in de avond	
S5	u wilt dus tussen achttien en vierentwintig uur vertrekken	EXPL
U5	ja	
S6	het spijt me ik heb u niet begrepen wilt u nog eens zeggen of u tussen achttien en vierentwintig uur wilt vertrekken	EXPL
U6	ja	
S7	er zijn vijf verbindingen dit is de vroegste (...) wilt u dat ik de verbinding nog eens herhaal	
U7	ja doe maar	
S8	het spijt me ik heb u niet begrepen kunt u nog eens zeggen of ik de verbinding moet herhalen	
U8	ja	
S9	ik herhaal de verbinding (...) zal ik de verbinding nog eens herhalen	
U9	nee	
S10	het spijt me ik heb u niet begrepen kunt u nog eens zeggen of ik de verbinding moet herhalen	
U10	nee niet herhalen	
S11	het spijt me maar ik heb u weer niet begrepen kunt u nog eens zeggen of ik de verbinding moet herhalen	
U11	nee	
S12	wilt u ook de eerstvolgende verbinding weten	
U12	nee	
S13	wilt u nog een andere verbinding weten	
U13	nee	
S14	het spijt me ik heb u niet begrepen kunt u nog eens zeggen of u een andere verbinding wilt weten	
U14	nee ik wil geen andere verbinding weten	

Figure 2: Original transcription of dialogue nr. 001/007 sampled from the OVIS corpus. System prompts are marked for verification type.

Turn	Utterance	Annotation
S1	goedenavond dit is het automatische informatiesysteem van openbaar vervoer reisinformatie dit systeem geeft uitsluitend informatie over treinreizen van ns-station naar ns-station van welk station naar welk station wilt u reizen	Q_VA
U1	ik moet volgende week dinsdag van schiphol naar nijmegen	S_VAD_PROB_OK
S2	van waar naar waar wilt u op dinsdag twaalf december reizen	Q_VA;I_LD
U2	van schiphol naar nijmegen	S_VA_OK_PROB
S3	hoe laat wilt u vanuit schiphol naar nijmegen reizen	Q_H;I_VA
U3	rond kwart over elf s avonds	S_TH_PROB_OK
S4	u wilt dus om elf uur achtendertig s ochtends vertrekken	E_TH
U4	nee s avonds	N;S_T_PROB_PROB
S5	wilt u mij alstublieft corrigeren	M
U5	nee ik wil om elf uur vijftien s avonds vertrekken	N;S_TH_PROB_PROB
S6	u wilt dus om drieentwintig uur achtendertig vertrekken	E_H
U6	zo ongeveer ja	A;Y_PROB_PROB
S7	het spijt me ik heb u niet begrepen wilt u nog eens zeggen of u om drieentwintig uur achtendertig wilt vertrekken	RE_H
U7	om drieentwintig uur vijftien	S_H_OK_PROB
S8	u wilt dus om drieentwintig uur vijftien vertrekken	E_H
U8	ja	Y_VOID_OK_OK

Figure 3: The original Dutch transcription of dialogue nr. 004/005 sampled from the OVIS corpus. User turns are annotated by our label set on the four components of SI, system prompts are analysed in terms of TRAs and slots.

Turn	Utterance
S1	ik moet nog uhm naar Hilde toe die nog steeds natuurlijk met 'r uh met 'r bovenlijf in 't gips zit.
S2	mm-hu.
S1	en ik moet nog tae-boën.
S2	ja dat weten ze niet hoe ze dat moeten schrijven. wedden?
S1	en ik moet nog... ggg nee. ggg. nee serieus.
S1	ja? tae-bo dat is T A E streepje B O maar die naam mag dus niet meer gebruikt worden.
S2	oh?
S1	omdat... heb je dat niet gehoord?
S2	nee.
S1	omdat uh Billy Blanks die vent die dus uh tae-bo heeft uitgevonden die uhm...
S2	je hoeft niet t- e- je hoeft uit te leggen. het gaat niet om inhoud.
S1	nee maar dat wil ik even zeggen tegen jou. xxx.
S2	xxx. ja oké ja maar ik weet wie je bedoelt ja.
S1	nou ja tae-bo... Billy Blanks die heeft dus dat uh heeft nu een rechtszaak.
S2	ja?
S1	en daarbij gaat 't d'rom dat dat dus die naam tae-bo die heeft hij dus zeg maar verzonnen en daar heeft ie vijftien jaar over gedaan om dat allemaal te ontwikkelen enzovoort.
S2	ja.
S1	en hij vindt dus dat hij dat alleen mag gebruiken. dus dat hij ook alleen maar die videobanden mag verkopen en dat sportscholen dus niet die naam tae-bo mogen gebruiken.
S2	zonder dat ze xxx ja.
S1	zonder dat ze aan hem heel veel geld ge-... hij wil dus gewoon heel veel geld verdienen d'raan. en hij verdient natuurlijk oh echt bakken met geld aan dat tae-bo.
S2	ja ja.

Figure 4: The first turns of a spontaneous dialogue sampled from the CGN corpus (sample nr. fn000451).

Turn	Utterance
S1	de Vlaamse regering wil vanaf midden volgend jaar starten met een zorgverzekering. zwaar hulpbehoevenden krijgen een financiële toelage voor hun niet-medische kosten. het bedrag schommelt tussen drieduizend vijfhonderd en zesduizend vijfhonderd frank per maand. het systeem wordt gefinancierd door de Vlaamse overheid maar ook door de burger. die betaalt dertig frank per maand via het ziekenfonds of een privé-verzekeringsfonds. Johnny Vansevenant.
S2	de vorige Vlaamse regering besliste al dat er een zorgverzekering moest komen. maar de concrete uitwerking ervan liet op zich wachten. nu zou het systeem worden georganiseerd via de ziekenfondsen of privé-verzekeringskassen. die zouden aan hun leden een bijdrage vragen van dertig frank per maand.

Figure 5: The first turns of a broadcasting discourse sampled from the CGN corpus (sample nr. fv600473).

Summary

The goal of this study is to develop and test a general method that can be implemented in an interpretation module of a spoken dialogue system (SDS). The interpretation process is called shallow since the material utilised is obtained directly from the speech recogniser and the dialogue manager of the SDS without performing deep linguistic processing.

Our approach integrates the components of the proposed shallow interpretation (SI) module in a machine learning framework where four pragmatic-semantic aspects of the user input are conceptualised as learning tasks: the detection of task-related acts (basic pragmatic acts exhibited by the user turn), information units (query slots for which information is provided by the user), forward-pointing problems (whether the user input is a source of communication problems in the interaction with the SDS) and backward-pointing problems (whether the user is aware that communication problems have occurred).

We train two supervised machine learning algorithms — MBL, a memory-based classifier, and RI, a rule induction classifier, considered as two extremes of working principle — on labelled data coming from a corpus of Dutch dialogues with a SDS in the travel domain. Dialogues are represented by a large amount of automatically extracted simple contextual features such as the wording and the history of system prompts, the full output of the speech recogniser, and acoustic-prosodic measurements of the speech signal, on the basis of which the user input is classified in terms of the four pragmatic-semantic components. An automatic algorithm parameter optimisation method [Van den Bosch 2004] is plugged into the module.

Our findings show that the ambitious task of simultaneously learning the four-level interpretation of spoken user turns (Chapter 4) yields significantly better results than an informed baseline strategy that draws on the most recently posed system prompt. Classification of task-related acts, slots, and backward-pointing problems is done better by MBL in these experiments than by RI.

In Chapter 5 we develop a method for improving the module's performance by means of class partitioning (i.e., the SI components are learnt in isolation and in different combinations with each other) and feature partitioning (i.e., classification draws on isolated information sources). We find that class partitioning has a substantial, positive influence on the scores produced by both classifiers for all SI components. The module's highest performance is attained on learning the task-related acts (MBL: 91.7 F-score, RI: 90.5), followed by the detection of backward-pointing problems (MBL: 90.8 F-score, RI: 88.5). The results for the remaining components — filled slot types (MBL: 87.7, RI: 85.5) and forward-pointing problems (MBL: 59.4, RI: 62.6) — are lower. The improvements gained

by optimising the class label of the learning tasks account for substantial error reductions, especially for RI, eliminating up to 50% of classification errors in terms of F-score. It is remarkable that in class partitioning MBL and RI produce statistically identical top performances concerning all four SI components. We observe that the various groups of information source contribute to a different extent to the classification tasks; primarily the speech recogniser output provides most information to classification. Using information coming from all available sources turns out to be best for extracting pragmatic-semantic information from spoken user input.

In Chapter 6 our aim is to block those pieces of information from the optimised learning algorithms that the literature supposes to negatively effect language processing. We design three, primarily machine learning-based methods to automatically filter the speech recogniser's output from disfluent words, from syntactically less dominant words, and from words that do not frequently occur in the recognition hypotheses. The experimental outcomes suggest that filtering cannot substantially improve learning performance on the SI task; we conclude that in our optimised experimental set-up the classifiers are enabled to internally cope with noise.

Besides answering our three research questions, the general aim of this work is to create an SI module that is robust in several respects. We conceptualise SI as a supervised classification task, and find that the proposed shallow approach leads to similar scale and performance tendencies of differently biased classifiers, suggesting that the method can with a similar success be implemented by other supervised learning techniques. In order to account for multi-layeredness in the input content, we extract information related to pragmatic and semantic levels of the user utterance. We conclude that an optimal component combination of such complex information is not trivial to determine, but is possible to find out via class partitioning, and that it would be useful to merge the backward-pointing problem component into the task-related act component of the SI module.

The module treats noisiness on several levels. The learning algorithms draw on approximative measurements, since the features extracted from the spoken user input include a large number of hypothetical values. Moreover, we find that when filtering techniques that incorporate higher-level linguistic information are applied to the noisy speech recogniser output, the cleaned data do not yield significantly better performance on the SI tasks than the noisy data. At the same time, we are also able to learn which factors of human-machine interaction can be identified as problem sources for the SI module itself (e.g., users accepting errors that the system has made), as well as for the examined SDS (the speech recogniser's difficulty in processing negative answers, aspects of the dialogue manager's prompting strategy).

The main finding of this study is that drawing on unsophisticated, potentially noisy features that characterise the dialogue situation, and by performing automatic optimisation of the formulated machine learning task at least in terms of class labels and algorithm parameter settings, it is possible to extract sophisticated information of practical pragmatic-semantic value from spoken user input with robust performance.

Samenvatting

De doelstelling van dit proefschrift is het ontwikkelen van een algemene methode die een zogenaamde ‘oppervlakkige interpretatie’ (*shallow interpretation*, SI) uitvoert van gebruikersuitingen in gesproken dialoogsystemen (*spoken dialogue systems*, SDSs). SDSs communiceren met een gebruiker in gesproken natuurlijke taal om een specifieke taak uit te voeren, bijvoorbeeld (zoals in ons onderzoek) het geven van informatie over treinreizen in Nederland. Interpretatie vindt in onze analysemodule plaats op vier pragmatisch-semantische niveau’s die de volgende aspecten van een gesproken gebruikersuiting (oftewel *spoken user turn*) beschrijven: basale acties die naar de onderliggende taak verwijzen (*task-related acts*), welke type informatie de gebruiker in zijn uiting geeft (*slots*, bijv. het invullen van vertrektijd of aankomststation), of de uiting communicatieproblemen oplevert tussen systeem en gebruiker (*forward-pointing problems*), en of uit de uiting blijkt dat de gebruiker zich bewust is van het ontstaan van communicatieproblemen (*backward-pointing problems*).

Het interpretatieproces wordt uitgevoerd door middel van gesuperviseerde lerende systemen die — op basis van geannoteerde voorbeelden uit het OVIS corpus — getraind worden om nieuwe uitingen te classificeren. In dit proefschrift gebruiken we twee algoritmen, *memory-based learning* (MBL) en *rule induction* (RI), die vaak gezien worden als twee extremen van het continuüm van lerende systemen. De aanpak is ‘oppervlakkig’ omdat de informatie die door de lerende algoritmen gebruikt wordt bestaat uit eenvoudige contextuele kenmerken (*features*) van de gesproken uiting, zoals akoestisch-prosodische metingen, de dialooggeschiedenis, en de woordhypotheses van de spraakherkenner (*automatic speech recogniser*, ASR) van het SDS. De parameters van MBL en RI worden bij elk experiment automatisch geoptimaliseerd met behulp van de methode van [Van den Bosch 2004].

De experimenten in Hoofdstuk 4 geven aan dat de complexe taak van het tegelijkertijd classificeren van alle vier SI niveau’s significant beter geleerd wordt door beide algoritmen dan door een geïnformeerde basisstrategie gebaseerd op de meest recente systeemvraag. Het classificeren van task-related acts, slots, en backward-pointing problems wordt significant beter gedaan door MBL dan door RI.

In Hoofdstuk 5 bekijken we of het mogelijk is de prestatie van de algoritmen te verbeteren door middel van *information partitioning*, d.w.z. het systematische verdelen en op een andere manier samenstellen van informatie in de klasse-componenten (*class partitioning*) en van de features voor een taak (*feature partitioning*). Wij zien dat class partitioning een positieve invloed heeft op beide algoritmen, zodat de beste resultaten (in termen van F-score) voor beide algoritmen op alle SI taken nagenoeg identiek zijn. De component die

in de module het meest succesvol voorspeld wordt is de task-related act (MBL: 91.7 F-score, RI: 90.5), gevolgd door backward-pointing problems (MBL: 90.8 F-score, RI: 88.5). De resultaten van de beide andere componenten — type slots (MBL: 87.7, RI: 85.5), en forward-pointing problems (MBL: 59.4, RI: 62.6) — liggen wat lager. Deze verbetering ten opzichte van het complexe experiment uit Hoofdstuk 4 levert een significante reductie van classificatiefouten op, vooral voor RI (tot 50% reductie op F-score). De experimenten met feature partitioning tonen aan dat onze informatiebronnen systematisch verschillen in hun bijdrage voor de SI taken, waarbij de ASR features het meest informatief blijken te zijn. In het algemeen worden de beste resultaten echter bereikt wanneer alle beschikbare informatie gebruikt wordt.

Hoofdstuk 6 beschrijft drie pogingen om de in de literatuur beschreven ‘ruis’ (*noise*) uit de ASR features te filteren: *disfluencies* (zoals zelfcorrecties, afgebroken woorden, enz.), *chunk non-heads* (d.w.z. syntactisch minder belangrijke woorden), en woorden met een lagere frequentie. In de drie experimenten wordt elk van deze bronnen van ruis automatisch uit de ASR features gefilterd, waarna de SI taken opnieuw geleerd worden. De resultaten lijken geen significante invloed aan te tonen van filtering op SI, waaruit afgeleid kan worden dat de geoptimaliseerde lerende algoritmen zelf al in staat zijn om met de ruis om te gaan.

Een belangrijke doelstelling van dit proefschrift is dat het model voor SI op verscheidene manieren robuust is. Allereerst leidt de formulering van de SI taak als een geoptimaliseerde leertaak er toe dat de twee algoritmen nagenoeg identieke prestaties kunnen behalen, waarbij we verwachten dat de hier beschreven aanpak en resultaten ook generaliseerbaar zijn naar andere gesuperviseerde leertechnieken. Daarnaast is de SI module in staat om alle vier componenten te interpreteren, waarbij de optimale samenstelling van de componenten automatisch gevonden kan worden. Tenslotte worden storingsfactoren in de interactie op meerdere manieren behandeld. De algoritmen gebruiken een groot aantal, mogelijk incorrecte, eenvoudige features voor het extraheren van complexe begrippen uit de gebruikersuiting. De resultaten van Hoofdstuk 6 laten zien dat er in principe geen verbetering optreedt in de module wanneer features gefilterd worden. Tegelijkertijd heeft het onderzoek praktische kennis opgeleverd over probleemfactoren, zowel voor de SI module (zoals acceptatie van systeemfouten door gebruikers) als voor de interactie met de SDS in het algemeen (zoals de problematische verwerking van negatieve uitingen, en diverse aspecten van de dialoogmanager).

De belangrijkste conclusie van ons werk is dat gebaseerd op eenvoudige, soms ruizige, contextuele kenmerken het mogelijk is om op robuuste wijze complexe informatie van pragmatisch-semantische aard te extraheren uit gesproken gebruikersuitingen met behulp van lerende algoritmen die geoptimaliseerd zijn op hun parameters en op samenstelling van de taakcomponenten.

Kivonat

Disszertációnk célja egy olyan általános eljárás kifejlesztése és tesztelése, amely beszélgető-rendszerbe (*spoken dialogue system*, SDS) ágyazva felhasználók hangzóbeszédének részleges értelmezését (*shallow interpretation*, SI) hajtja végre explicit grammatikai információ felhasználása nélkül. Az SDS-ek általános funkciója, hogy valamely szolgáltatást gépi verbális kommunikáció segítségével elérhetővé tegyenek, illetve arról — az általunk vizsgált esetben például vasúti menetrendről — felvilágosítást adjanak. Az ember-gép interakció során tehát a felhasználó beszédéből, mely többnyire az SDS által feltett kérdésre adott válasz, a gépnek ki kell nyernie a sikeres kommunikációhoz és a végeredményt adó adatbázis-lekérdezéshez szükséges információt. Az általunk fejlesztett modulban ez a mesterséges intelligencia módszereinek egyikével, felügyelt gépi tanuló algoritmusok (*supervised machine learning algorithms*) felhasználásával történik, melyet mind memórialapú tanulással (*memory-based learning*, MBL), mind szabálytanulással (*rule induction*, RI) elvégzünk.

Az MBL és az RI algoritmusokat — melyek a tanulás intenzitásának és a megszerzett tudás reprezentálásának szempontjából a felügyelt tanulási módszerek két végletének tekinthetők — holland nyelvű, ember-gép párbeszédet tartalmazó annotált gyakorlókorpuszon tanítjuk a felhasználó hangzóbeszédben bevitt szövegének (*spoken user input*) négy pragmatikai-szemantikai szinten való részleges értelmezésére. A szintek a következők: a felhasználó válaszában alapvető kommunikatív aktusa (*task-related act*, TRA), a felhasználó által megadott tartalmi információ típusa az SDS lekérdezési struktúrájában (*filled query slot*, SLOT), annak felismerése, hogy a bevitt mondat fog-e problémát okozni az adott párbeszédben (*forward-pointing problem*, FWD PR), valamint annak azonosítása, hogy amennyiben a diskurzus során probléma merült fel, a felhasználó tudatában van-e ennek (*backward-pointing problem*, BWD PR). A tanulás alapjául a párbeszédet egyszerű, az SDS-ben automatikusan hozzáférhető jellemzői szolgálnak, melyeket három csoportra oszthatunk: a beszédjel akusztikai-prozódiai attribútumai, az SDS dialógusmenedzsere által tárolt párbeszédelőzmények (*dialogue history*), valamint az SDS automatikus beszédfelismerője (*automatic speech recogniser*, ASR) teljes, szógráf-hipotézis formájú kimenete, melyet rendezetlen szóhalmazként (*“bag-of-words”*) ábrázolunk. A tanulás során mindkét algoritmus paramétereit automatikusan optimalizáljuk [Van den Bosch 2004] módszerével.

A disszertáció 1. fejezete a megválaszolni kívánt kutatási kérdéseket fogalmazza meg. A 2. fejezet más kutatócsoportok SI-re vonatkozó eredményeit és módszereit tárgyalja, különös tekintettel a korpuszannotálásra, valamint a pragmatikai-szemantikai információ kinyeréséhez felhasznált interakció-jellemzőkre. A 3. fejezet bemutatja a gépi tanulás

területét, ahol főként a memóriaalapú és a szabálytanuló algoritmusok belső mechanizmusára fordítunk figyelmet, valamint meghatározzuk tanulási kísérleteink metodikáját.

A 4. fejezetben leírt kísérletek alapján megállapítható, hogy az SI modul a bonyolult, négyszintű értelmezést eredményesebben képes elvégezni, mint az az alapeljárás (*baseline strategy*), amely az elemzéshez az SDS legutóbb feltett kérdését veszi figyelembe. A TRA, SLOT, és BWD PR részfeladatokat az MBL algoritmus az összetett feladat során jobb eredménnyel oldja meg, mint az RI algoritmus.

Munkánk 5. fejezetében az SI modult az információmegosztás (*information partitioning*) módszerével fejlesztjük tovább: az algoritmusok mind a négy SI alfeladatot az összes lehetséges kombinációban megtanulják (*class partitioning*), majd a kapott optimális kombinációt a három elkülönített attribútumcsoport alapján (*feature partitioning*). Megállapítjuk, hogy a *class partitioning* módszerrel a modul teljesítménye hatékonyan növekszik (néhány esetben a komplex feladathoz képest esetenként 50%-os hibacsökkenést is elérve) mind MBL, mind RI esetében. A legjobb eredményt a TRA komponensen (vagyis az alapvető beszédaktus-típus kinyerésén) érjük el (MBL: 91.7 F-score, RI: 90.5), melyet a BWD PR komponens követ (MBL: 90.8 F-score, RI: 88.5). A SLOT komponens klasszifikációja valamivel alacsonyabb eredménnyel végezhető el (MBL: 87.7, RI: 85.5). A legnehezebb feladatnak a FWD PR komponens meghatározása bizonyul (MBL: 59.4, RI: 62.6). Meglepő vizsgálati eredmény, hogy ezzel a módszerrel a két algoritmus az értelmezés egy-egy szintjét statisztikailag azonos teljesítménnyel képes megtanulni. Megállapítjuk, hogy a három információforrás közül az ASR attribútumcsoport bizonyul a leghasznosabbnak; általában azonosan legoptimálisabb az összes lehetséges attribútum figyelembe vétele a tanulás során.

A 6. fejezetben az eképpen optimalizált értelmezőmodul robusztusságának (*robustness*) további vizsgálatát hajtjuk végre. Három kísérleti módszer kerül alkalmazásra abból a célból, hogy az ASR "zajos" (*noisy*) kimeneti hipotéziséből kiszűrjünk háromféle — a szakirodalom által zavarónak feltételezett — elemet: a diszfluens elemeket (*disfluencies*, úgymint önjavítás, szóismétlés, hűmmögés, stb.), a grammatikailag alárendelt szavakat (*chunk non-heads*), és azokat a szavakat, amelyeket az ASR hipotézise alacsonyabb gyakorisággal tartalmazza. Az első két módszert ismét automatikusan, gépi tanulási technikák segítségével végezzük, esetenként grammatikai információ felhasználásával. A megszürt ASR kimeneten (a többi attribútummal egyetemben) elvégzett SI kísérletek azonban azt sugallják, hogy az értelmezési feladat nem igényli az attribútumok zajosságának csökkentését, mivel a kapott eredmények nem mutatnak jelentős változást az 5. fejezetben tárgyaltakhoz képest. Ebből arra következtetünk, hogy az SI modul az optimalizálás során belső robusztusságot ért el ezekkel az elemekkel szemben.

A leírt technikai megoldással nemcsak beszélt nyelvi szöveg részleges elemzését tudtuk elvégezni, de hasznos tudást nyertünk a gép és az ember között folyó párbeszéd számos gyakorlati jellemzőjéről. Noha a modell az utazás témakörében került bemutatásra, a kísérletek általános metodikája, valamint a kapott teszteredmények az eljárás általános alkalmazhatóságára utalnak. Kutatásunk során megállapítottuk, hogy a kifejlesztett modul képes arra, hogy zajos, egyszerű attribútumok alapján gépi tanulási módszerekkel felhasználók hangzóbeszédéből magasszintű, pragmatikai-szemantikai jellegű információt nyerjen ki stabil teljesítménnyel.

Bibliotheek K. U. Brabant



17 000 01543220 7

ISBN
90901
88746