Statistical Methods in Natural Language Understanding and Spoken Dialogue Systems

Von der Fakultät für Mathematik, Informatik und
Naturwissenschaften der RWTH Aachen University
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von:

Diplom–Informatiker Klaus Macherey
aus Düren-Birkesdorf

Berichter: Universitätsprofessor Dr.-Ing. Hermann Ney
Professor Dr.-Dr. Wolfgang Minker

Tag der mündlichen Prüfung: 22. September 2009

Abstract

Modern automatic spoken dialogue systems cover a wide range of applications. There are systems for hotel reservations, restaurant guides, systems for travel and timetable information, as well as systems for automatic telephone-banking services. Building the different components of a spoken dialogue system and combining them in an optimal way such that a reasonable dialogue becomes possible is a complex task because during the course of a dialogue, the system has to deal with uncertain information. In this thesis, we use statistical methods to model and combine the system's components. Statistical methods provide a well-founded theory for modeling systems where decisions have to be made under uncertainty. Starting from Bayes' decision rule, we define and evaluate various statistical models for these components, which comprise speech recognition, natural language understanding, and dialogue management.

The problem of natural language understanding is described as a special machine translation problem where a source sentence is translated into a formal language target sentence consisting of concepts. For this, we define and evaluate two models. The first model is a generative model based on the source-channel paradigm. Because the word context plays an important role in natural language understanding tasks, we use a phrase-based translation system in order to take local context dependencies into account. The second model is a direct model based on the maximum entropy framework and works similar to a tagger. For the direct model, we define several feature functions that capture dependencies between words and concepts. Both methods have the advantage that only source-target pairs in the form of input-output sentences must be provided for training. Thus, there is no need to generate grammars manually, which significantly reduces the costs of building dialogue systems for new domains.

Furthermore, we propose and investigate a framework based on minimum error rate training that results in a tighter coupling between speech recognition and language understanding. This framework allows for an easy integration of multiple knowledge sources by minimizing the overall error criterion. Thus, it is possible to add language understanding features to the speech recognition framework and thus to minimize the word error rate, or to add speech recognition features to the language understanding framework and thus to minimize the slot error rate.

Finally, we develop a task-independent dialogue manager using trees as the fundamental data structure. Based on a cost function, the dialogue manager chooses the next dialogue action with minimal costs. The design and the task-independence of the dialogue manager leads to a strict separation of a given application and the operations performed by the dialogue manager, which simplifies porting an existing dialogue system to a new domain. We report results from a field test in which the dialogue manager was able to choose the optimal dialogue action in 90% of the decisions. We investigate techniques for error handling based on confidence measures defined for speech recognition and language understanding. Furthermore, we investigate the overall performance of the dialogue system when confidence measures from speech recognition and natural language understanding are incorporated into the dialogue strategy. Experiments have been carried out on the TelDir database, which is a German in-house telephone directory assistance corpus, and on the Taba database, which is a German in-house train time scheduling task.

Zusammenfassung

Automatische sprachbasierte Dialogsysteme werden heutzutage in zahlreichen Anwendungen eingesetzt. So gibt es beispielsweise Dialogsysteme für Hotelreservierungen, Systeme zur Reise- und Fahrplanauskunft, sowie Dialogsysteme für das sogenannte Telefon-Banking. Damit ein für den Benutzer sinnvoller Dialog zustande kommt, muss ein Dialogsystem diejenigen Entscheidungen treffen, die zur Beantwortung der Benutzeranfrage führen. Da ein sprachbasiertes Dialogsystem nur ein unvollständiges Modell der Wirklichkeit besitzt und die Dekodierung von Benutzereingaben fehlerhaft sein kann, können die Entscheidungen eines Dialogsystems im allgemeinen nicht auf Basis von Faktenwissen getroffen werden, sondern müssen aufgrund unvollständigen Wissens erfolgen. Um die Unsicherheit in den Entscheidungen zu beschreiben, verwenden wir in dieser Arbeit statistische Methoden zur Modellierung der Komponenten eines Dialogsystems. Ausgehend von der Bayesschen Entscheidungsregel definieren und evaluieren wir verschiedene Modelle, mit deren Hilfe wir die Spracherkennungskomponente, die Sprachverstehenskomponente und den Dialog Manager modellieren.

Das Problem des Verstehens natürlicher Sprache wird als ein spezielles Problem der maschinellen Übersetzung beschrieben, wobei ein Satz der Quellsprache in einen Satz der Zielsprache übersetzt wird. Die Zielsprache ist dabei eine formale Sprache bestehend aus Konzepten. Hierzu untersuchen wir zwei Modelle: das erste Modell ist eine generatives Modell, welches auf dem Source-Channel Paradigma basiert. Da lokale Kontexte von Wörtern eine zentrale Rolle beim Verstehen natürlicher Sprache spielen, verwenden wir ein phrasenbasiertes Übersetzungssystem, dass Wörter im Kontext modellieren kann. Das zweite Modell ist ein direktes Modell, welches auf dem Maximum Entropie Ansatz basiert und ähnlich wie ein Tagger eingesetzt wird. Für das direkte Modell definieren wir zahlreiche Feature Funktionen, welche die komplexen Abhängigkeiten zwischen Wörtern und Konzepten erfassen. Beide Ansätze haben den Vorteil, dass nur Satzpaare in Form von Ein-Ausgabe Sätzen dem Trainingsalgorithmus zur Verfügung gestellt werden müssen. Dadurch entfällt die manuelle Generierung von Grammatiken, welche häufig im Kontext von Sprachverstehenssystemen eingesetzt werden.

Desweiteren stellen wir einen auf dem Minimum Error Rate Training basierenden Ansatz vor, der eine stärkere Kopplung zwischen Spracherkenung und Sprachverstehen erlaubt. Der Ansatz ermöglicht auf einfache Weise die Integration zahlreicher Feature Funktionen bei gleichzeitiger Minimierung des Evaluationskriteriums. Dadurch ist es möglich, die Wissensquellen der Sprachverstehenskomponente in die Spracherkennungskomponente zu integrieren und somit die Wortfehlerrate zu minimieren, beziehungsweise umgekehrt die Wissensquellen der Spracherkennungskomponente mit den Wissensquellen der Sprachverstehenskomponente zu kombinieren und somit die Konzeptfehlerrate zu minimieren.

Zusätzlich entwickeln wir einen domänenunabhängigen Dialog Manager, der auf Basis einer Kostenfunktion die nächstfolgende Dialogaktion bestimmt. Die Domänenunabhängigkeit des Dialog Managers führt zu einer strikten Trennung zwischen der konkreten Applikation und den Operationen, die der Dialog Manager ausführen kann. Dies vereinfacht die Portierung eines existierenden Dialog Managers auf eine neue Domäne. Wir zeigen empirisch, dass der Dialog Manager in einem Feldtest in der Lage war für ca. 90% aller Entscheidungen die jeweils optimale Entscheidung zu treffen.

Abschließend untersuchen wir Techniken zur Fehlerbehandlung in sprachbasierten Dialogsystemen basierend auf Konfidenzmaßen. Dabei untersuchen wir die Performanz des Dialogsystems für den Fall, dass Konfidenzmaße für die Spracherkennung und das Sprachverstehen in die Dialogstrategie eingebaut werden. Experimentelle Resultate werden für die TelDir und die Taba Datensammlung diskutiert. Die TelDir Datensammlung ist ein Korpus aus der Domäne Telefonbuchassistent für die deutsche Sprache, die Taba Datensammlung ist ein Korpus aus der Domäne Fahrplanauskunftssystem, ebenfalls für die deutsche Sprache.

Contents

List of Tables

List of Figures

Acknowledgment

"*You can know the name of a bird in all the languages of the world, but when you're finished, you'll know absolutely nothing whatever about the bird. So let's look at the bird and see what it's doing – that's what counts.*"

(Richard Feynman, physicist, 1918-1988)

First I would like to thank my supervisor Prof. Dr.-Ing. Hermann Ney, head of department of computer science 6, RWTH Aachen University, for the opportunity to realize this work as part of his team. He introduced me to the exciting field of pattern recognition in general and speech recognition, natural language understanding, and machine translation in particular. I am also grateful to my second supervisor Prof. Dr. Dr.-Ing. Wolfgang Minker who is Professor at the University of Ulm for agreeing to review this thesis and for his interest in this work.

My most grateful thanks go to my parents Peter and Ria Macherey for their unconditional support with my studies and their trust in all my decisions.

Special thanks go to Franz Josef Och who contributed to this thesis by providing software packages which in part became open source software like the alignment toolkit giza++ and the maximum entropy toolkit yasmet, and his non-public implementation of the alignment templates approach which I used for comparison experiments throughout this thesis. I would also like to thank him for the numerous interesting and enlightening discussions we had. I also want to thank Oliver Bender for his prior work on maximum entropy modeling and its application to natural language processing. Important results of this thesis would not have been possible without his contributions. I am also grateful to all the other people working at the department of computer science 6, including Gisela Gillmann for her valuable work and especially all my former colleagues from the "Aachener Kreis".

I want to thank Drs. Stefan Dobler and Klaus Reinhard who initiated this research project when they were with Ericsson Eurolab Deutschland GmbH and Drs. Andreas Kellner from Philips and Reinhard Blasig from Scansoft for providing text and speech data for the TABA corpus.

And finally a big thank you to Willi Geffers and his team from the *Rechnerbetrieb für Informatik* for granting me a special account with more disk space for my home directory and for setting up the SGE cluster which was vital for all the experiments I performed.

Chapter 1

Introduction

Spoken dialogue systems can be viewed as an advanced application of spoken language technology. They provide an interface between users and a computer-based application that permits spoken interaction with the application in a relatively natural manner. In so doing, spoken dialogue systems provide a stringent test for the major fields of spoken language technology, including speech recognition and speech synthesis, language understanding, and dialogue management.

Each of the listed fields is a scientific area of its own, and combining the different components into a spoken dialogue system as well as dealing with their interplay in an optimal manner such that a reasonable dialogue between a human and a machine becomes possible is a challenging task because during the course of a dialogue, the system has to deal with uncertain information.

In this thesis, we use statistical methods to model and combine the various components of a spoken dialogue system. We propose approaches for natural language understanding that avoid building grammars manually and investigate techniques that allow for combining the various knowledge sources from multiple system components such that the overall error criterion is minimized. Especially, we investigate the following points:

- **Natural language understanding**
  We present two approaches for analyzing the semantics of natural language inputs and discuss their advantages and drawbacks. The first approach uses a generative model based on the source channel paradigm whereas the second approach directly optimizes the posterior probability using the maximum entropy framework. Starting from an annotated corpus, we describe the problem of natural language understanding as a translation from a source sentence to a formal language target sentence. We investigate the quality of different alignment models and feature functions, and show that the direct maximum entropy model outperforms the generative source channel-based approach.

- **Confidence measures for speech recognition and language understanding**
  We define confidence measures derived from posterior probabilities for automatic speech recognition and spoken language understanding. The computation is done on word graphs and concept graphs, respectively, and is suitable for realtime decoding

systems. In addition, we investigate whether graph-based methods also outperform N-best list approaches for language understanding tasks. While the superiority of graph-based methods has been shown for speech-recognition tasks, it is unclear, which method is best suited for language understanding tasks, where we have to deal with word positions rather than time-aligned word boundaries.

- **Feature-based spoken language understanding**
  Based on the results from the previous points, we enrich the maximum entropy-based natural language understanding component with additional features extracted from an automatic speech recognition system and show that these features can improve the overall performance of a spoken language understanding system. The feature functions used comprise language and acoustic model probabilities, length models, posterior probabilities, spelling correction features, and features based on confidence measures.

- **Combining speech recognition and spoken language understanding**
  In order to yield a tighter coupling between speech recognition and spoken language understanding, we combine and weight the knowledge sources of both components such that the overall error criterion is minimized. For this, we employ the minimum error rate training framework that has become a defacto standard in statistical machine translation. By combining the different knowledge sources, we can minimize the word error rate, the slot error rate, or a combination of both error rates. By combining the error counts for both components, we show that we can simultaneously minimize both word and slot error rates.

- **Domain-independent dialogue management**
  Ideally, the dialogue manager should be application-independent. To achieve this, we distill the steps that many domains have in common and store the domain-specific task knowledge in a parameterizable data structure based on trees. Using several feature functions, all operations of the dialogue manager can be formulated as cost functions that operate directly on nodes, paths, and trees. We investigate the capability of the system to choose those actions that are likely to lead as directly as possible through the tree to the user's goal and show that different knowledge sources can be easily integrated into this framework without the necessity to rewrite the system.

- **Dialogue strategies**
  The actions chosen by the dialogue manager define its dialogue strategy. The goal of the dialogue manager is to choose a dialogue strategy that meets the user's request by minimizing the costs of the dialogue. A common approach to dialogue strategies is to use a finite state representation of the dialogue. Transitions between the states determine the actions taken by the dialogue manager. In this thesis, we use decision trees in order to compute the next state-action pair and measure the dialogue success

rate as well as the overall system performance within a field test. In order to keep the space of possible state-action pairs small, we define meta-states that significantly reduce the information stored in a dialogue state.

- **Error-tolerant dialogue modeling**
  For spoken dialogue systems, errors can occur on different levels of the system's architecture. One of the principal causes for errors during a dialogue session are erroneous recognition results, which often lead to incorrect semantic interpretations. Even if the speech input signal has been correctly recognized, the natural language understanding component can produce error-prone sentence meanings due to the limitations of its underlying model. To cope with this problem, we introduce a multi-level error-detection mechanism based on several feature functions in order to find erroneous recognitions, error-prone semantic interpretations as well as ambiguities and contradictions. Here, the confidence output of one component is passed as an additional input to the next component. The proposed features are passed to the dialogue manager who then determines the subsequent dialogue action.

Chapter 2

Fundamentals of Spoken Dialogue Systems

In this chapter, we describe the objective of a spoken dialogue system and introduce the basic terminology used throughout this thesis. Starting from Bayes' decision rule, we embed a spoken dialogue system into a statistical framework and describe the different components a dialogue system consists of. The chapter ends with a discussion of evaluating and measuring the performance of spoken dialogue systems.

2.1 Architecture of a Spoken Dialogue System

The objective of a spoken dialogue system is to let a human interact with a computer-based information system using speech as means of interaction. Because a spoken dialogue system can make errors and user requests can be incomplete or ambiguous, a spoken dialogue system must actively request for missing data and confirm information, resulting in a sequence of interactions between a user and a machine. This turn-based sequence of interactions between a user and a machine is called a *dialogue*. Figure 2.1 illustrates the course of a dialogue between a human (user) and a machine (system) by example of a telephone directory assistance task. In this example, the user can ask for telephone numbers, fax numbers, and email addresses of persons, departments, and companies. After each user input, the system must select an appropriate dialogue action and response to the user in a cooperative way such that finally the user's request can be answered.

Formally, a dialogue turn of the system can be represented as a tuple (S, A) where S denotes the state space and A denotes the action set of the dialogue system. A *state* $s \in S$ represents all the knowledge the system has about internal and external resources it interacts with [Levin & Pieraccini^{+} 98]. This comprises user input, database entries, dialogue histories, and so on. The *action set* of a dialogue system describes the transitions between dialogue states and includes all possible actions it can perform, such as asking the user for input, providing output to the user, confirming user input, or interacting with the database. A path through the state-action space can be described as a finite sequence of state-action pairs $s_0 a_0 s_1 a_1 \dots s_F$ starting from an initial state s_0 and ending in a final state s_F. A complete sequence of state-action pairs between a user and a machine is called

System:	Hello. This is the RWTH telephone directory assistant. How may I help you?
User:	Uhm, what shall I do now?
Manager:	select action: `ask_for_information`
System:	Do you need a telephone number, fax number, or email address?
User:	Can you give me the number of Mr. Macherey?
Manager:	select action: `disambiguate`
System:	I have two persons named Macherey in my database. Please refine your request.
User:	Mr. Wolfgang Macherey.
Manager:	select action: `present`
System:	The phone number of Mr. Wolfgang Macherey is 80 21613. Shall I repeat the number?
	. . .

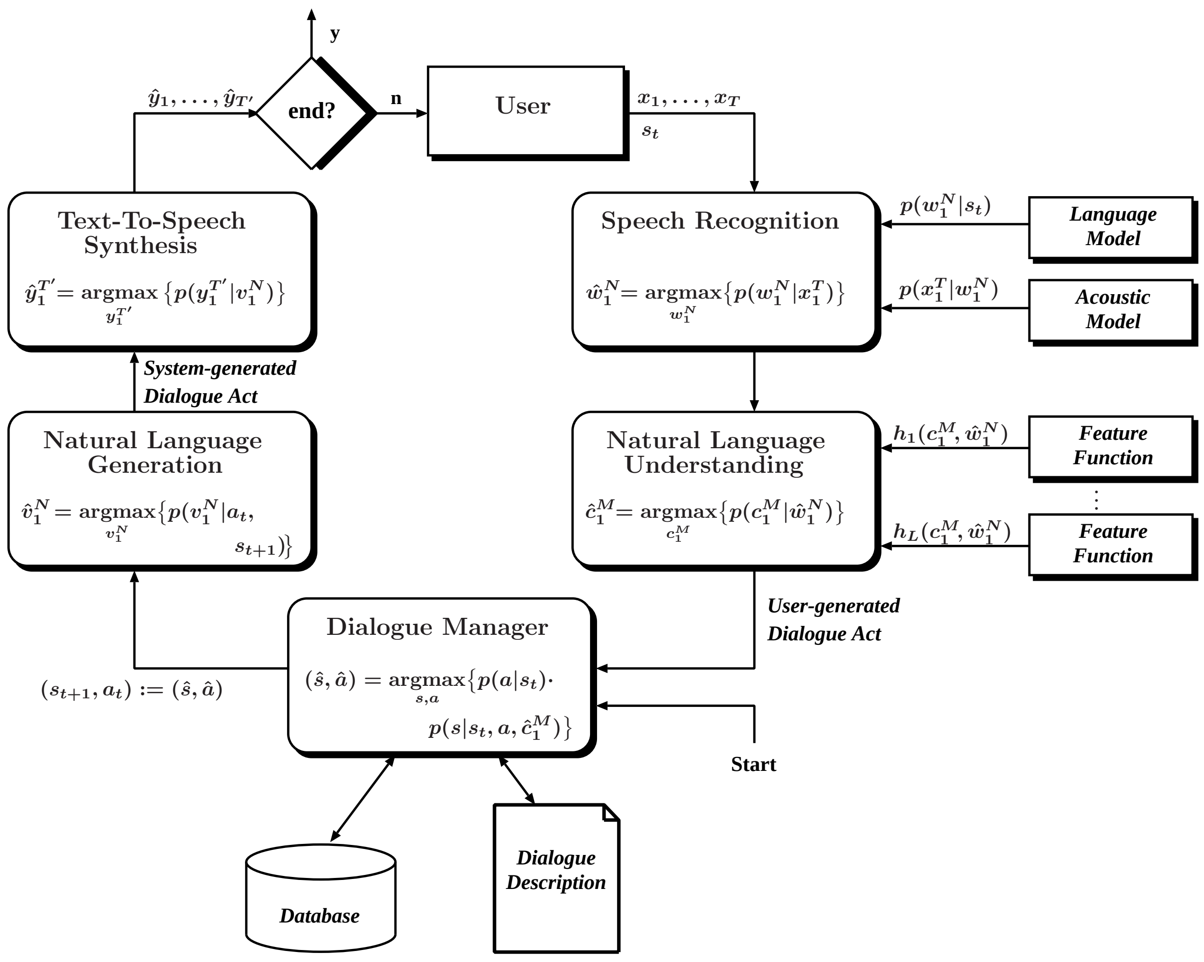
Application Database

Id	LastName	FirstName	PhoneNumber
⋮	⋮	⋮	⋮
170	Keysers	Daniel	8021610
189	Macherey	Klaus	8021617
190	Macherey	Wolfgang	8021613
⋮	⋮	⋮	⋮

Figure 2.1: Example of a dialogue between a human (User) and a machine (System). After each user input, the dialogue manager (Manager) determines the next dialogue action and the system outputs a response.

a *dialogue session*, a *dialogue transaction*, or in short a *dialogue*.

The main components of a sequential spoken dialogue system are depicted in Figure 2.2. The user provides input to the system by means of speech. The sort of input is referred to as *modality* of the dialogue system. In general a dialogue system can use other input modalities, such as text, gestures, touch-tones, and so on. Dialogue systems that employ more than one input modality are referred to as *multi-modal* systems. The user utterance is decoded into a sequence of words using automatic speech recognition. The natural language understanding component extracts all the information contained in the decoded word sequence that is relevant to the underlying task by converting its input into a semantic representation. This semantic representation serves as input to the dialogue manager who updates the current dialogue state and determines the next dialogue action. In general, the dialogue manager can choose between several actions for a given state s. A concrete selection is referred to as the *dialogue strategy* of the dialogue system. Both user utterances as well as system outputs can be classified and described by dialogue acts. A *dialogue act* represents the meaning of an utterance at the level of illocutionary force [Austin 62]. The illocutionary force of an utterance is the speaker's intention in producing that utterance. Dialogue acts can be useful for predicting the discourse of a dialogue. Examples for dialogue acts are statements, confirmations, rejections, yes-no questions, and so on. Depending on the action chosen and the current dialogue state a response must be generated by the natural language generation component. Response generation is essentially the inverse of the understanding process [Young 02]. The system



y
$\hat{y}_1, \ldots, \hat{y}_{T'}$
end?
n
User
$x_1, \ldots, x_T$
s_t
Text-To-Speech Synthesis
$\hat{y}_1^{T'} = \operatorname{argmax}_{y_1^{T'}} \{p(y_1^{T'} | v_1^N)\}$
Speech Recognition
$\hat{w}_1^N = \operatorname{argmax}_{w_1^N} \{p(w_1^N | x_1^T)\}$
$p(w_1^N | s_t)$
Language Model
$p(x_1^T | w_1^N)$
Acoustic Model
System-generated Dialogue Act
Natural Language Generation
$\hat{v}_1^N = \operatorname{argmax}_{v_1^N} \{p(v_1^N | a_t, s_{t+1})\}$
Natural Language Understanding
$\hat{c}_1^M = \operatorname{argmax}_{c_1^M} \{p(c_1^M | \hat{w}_1^N)\}$
$h_1(c_1^M, \hat{w}_1^N)$
Feature Function
$h_L(c_1^M, \hat{w}_1^N)$
Feature Function
User-generated Dialogue Act
Dialogue Manager
$(\hat{s}, \hat{a}) = \operatorname{argmax}_{s,a} \{p(a|s_t) \cdot p(s|s_t, a, \hat{c}_1^M)\}$
$(s_{t+1}, a_t) := (\hat{s}, \hat{a})$
Start
Database
Dialogue Description

Figure 2.2: Architecture of a sequential spoken dialogue system. The user utterance is represented as a sequence of vectors x_1^T that is decoded into a sequence of words $\hat{w}_1^N$. After translating the word sequence into the most likely sequence of concepts $\hat{c}_1^M$, the dialogue manager determines the subsequent dialogue action a_t and successor state s_{t+1}. Depending on the current dialogue state-action pair a response is generated and communicated to the user.

response consisting of the word sequence $v_1, \ldots, v_N$ is transformed into an acoustic signal $\hat{y}_1, \ldots, \hat{y}_{T'}$ using text-to-speech synthesis and is output to the user.

2.2 Decision Rules

From a statistical point of view we try to find in each turn the optimal state-action pair $(\hat{s}_{t+1}, \hat{a}_t)$ that maximizes the conditional probability $pr(s_{t+1}, a_t | s_t, x_1^T)$ where the predecessor state s_t and the user utterance x_1^T are given.[1]

$$(\hat{s}_{t+1}, \hat{a}_t) = \operatorname*{argmax}_{s_{t+1}, a_t} \left\{ pr(s_{t+1}, a_t | s_t, x_1^T) \right\} \tag{2.1}$$

$$\begin{aligned} pr(s_{t+1}, a_t | s_t, x_1^T) &= \sum_{c_1^M} pr(s_{t+1}, a_t, c_1^M | s_t, x_1^T) \\ &\cong \max_{c_1^M} \Big\{ \underbrace{\underbrace{p(a_t|s_t)}_{\text{Strategy}} \cdot \underbrace{p(s_{t+1}|s_t, a_t, c_1^M)}_{\text{Task Model}}}_{\text{Dialogue Manager}} \cdot p(c_1^M | x_1^T, s_t) \Big\} \end{aligned} \tag{2.2}$$

$$\begin{aligned} p(c_1^M | x_1^T, s_t) &= \sum_{w_1^N} p(c_1^M, w_1^N | x_1^T, s_t) \\ &\cong \max_{w_1^N} \Big\{ \underbrace{p(w_1^N | x_1^T, s_t)}_{\text{ASR}} \cdot \underbrace{p(c_1^M | w_1^N, x_1^T, s_t)}_{\text{NLU}} \Big\} . \end{aligned} \tag{2.3}$$

Introducing the sequence of concepts c_1^M as a hidden variable allows for decomposing $pr(s_{t+1}, a_t | s_t, x_1^T)$ into several submodels resulting in Equation 2.2. Both the strategy and the task model constitute the dialogue manager. The strategy model provides a probability for each action a_t given the state s_t. The task model describes possible state transitions given the predecessor state, the dialogue action, and the sequence of concepts. For the remaining probability $p(c_1^M | x_1^T, s_t)$, we ignore the dependence on the dialogue action. By introducing the word sequence w_1^N as another hidden variable, we can decompose $p(c_1^M | x_1^T, s_t)$ into an automatic speech recognition (ASR) term and a natural language understanding (NLU) term where the NLU term still depends on the acoustic observations. Plugging Equations 2.2 and 2.3 into Equation 2.1 yields the following decision rule for each dialogue turn:

$$\begin{aligned} (\hat{s}_{t+1}, \hat{a}_t) = \operatorname*{argmax}_{s_{t+1}, a_t} \Big\{ p(a_t|s_t) \cdot \max_{c_1^M} \Big\{ p(s_{t+1}|s_t, a_t, c_1^M) \cdot \\ \cdot \max_{w_1^N} \left\{ p(w_1^N | x_1^T, s_t) \cdot p(c_1^M | w_1^N, x_1^T, s_t) \right\} \Big\} \Big\} . \end{aligned} \tag{2.4}$$

Since solving Equation 2.4 is difficult, its solution can be approximated by modeling the dialogue system as a sequential operation. Figure 2.2 describes a spoken dialogue system as a sequential operation where the decision of one stage serves as input for the subsequent stage resulting in the update of the current dialogue state-action pair. This leads to the

[1] A detailed discussion on how this quantity is embedded in a decision rule for a complete dialogue transaction is presented in Section 2.5.

following chain of decisions:

$$\hat{w}_1^N = \underset{w_1^N}{\operatorname{argmax}} \left\{ p(w_1^N | x_1^T, s_t) \right\} \tag{2.5}$$

$$\hat{c}_1^M = \underset{c_1^M}{\operatorname{argmax}} \left\{ p(c_1^M | \hat{w}_1^N, s_t) \right\} . \tag{2.6}$$

Plugging Equations 2.5 and 2.6 into Equation 2.4 yields the following modified decision rule:

$$\begin{aligned}(\hat{s}_{t+1}, \hat{a}_t) = \underset{s_{t+1}, a_t}{\operatorname{argmax}} \Bigg\{ & p(a_t|s_t) \cdot p(s_{t+1}|s_t, a_t, \hat{c}_1^M) : \\ & \hat{c}_1^M = \underset{c_1^M}{\operatorname{argmax}} \left\{ p(c_1^M | \hat{w}_1^T, s_t) \right\} : \\ & \hat{w}_1^N = \underset{w_1^N}{\operatorname{argmax}} \left\{ p(w_1^N | x_1^T, s_t) \right\} \Bigg\} .\end{aligned} \tag{2.7}$$

That is, we first determine the optimal word sequence $\hat{w}_1^N$ and use it as input to the natural language understanding component, which yields the optimal sequence of concepts $\hat{c}_1^M$ that is used to obtain the next state-action pair. Although Equation 2.7 leads to a much simpler architecture, using this decision rule to compute the next dialogue state-action pair can turn out to be suboptimal because the interdependency of decisions is ignored. Knowledge sources that are introduced to model the probability distribution of a successor component might also be beneficial for modeling the predecessor components. However, by making independent decisions in the pipeline of Equation 2.7, these knowledge sources are often not considered in predecessor components. Furthermore, errors made in one component are propagated to the next component. Since a subsequent component considers its possibly erroneous input as its observation, errors often remain undetected. To mitigate this effect for the speech recognition and language understanding component, one can produce word graphs in a first pass recognition and perform the semantic analysis on the graph structure. The graph structure provides a compact representation of sentence hypotheses. By rescoring the graph with knowledge sources derived from the language understanding component, we can make a more informed decision when searching for the best word sequence.

An important question in the design of a spoken dialogue system is to decide who can take control of the conversation. Systems that completely control the conversation in a predefined and rigid manner are called *single initiative* or *system initiative* systems. If both the dialogue manager and the user can take control of the dialogue allowing the user to answer more than one question at a time, the system is called a *mixed initiative* system.

In the following sections, we will briefly describe each component of a spoken dialogue system separately. Section 2.3 overviews the fundamentals of automatic speech recognition followed by natural language understanding in Section 2.4, and dialogue management in

Section 2.5. System output generation is described in Sections 2.6 and 2.7. The chapter concludes with a discussion on how spoken dialogue systems are evaluated.

2.3 Automatic Speech Recognition

The aim of an automatic speech recognition system is to decode a given speech input into text. Nowadays, most automatic speech recognition systems follow the statistical paradigm using Bayes' decision rule. According to Bayes' decision rule [Bayes 63], that word sequence $w_1^N = w_1, \ldots, w_N$ should be chosen that maximizes the posterior probability for a given sequence of acoustic observations $x_1^T = x_1, \ldots, x_T$:

$$\begin{aligned} \hat{w}_1^N &= \operatorname*{argmax}_{w_1^N} \left\{ pr(w_1^N | x_1^T) \right\} \\ &= \operatorname*{argmax}_{w_1^N} \left\{ pr(w_1^N) \cdot pr(x_1^T | w_1^N) \right\} . \end{aligned} \tag{2.8}$$

Equation 2.8 transforms the recognition problem to maximizing the product of two probability distributions: the language model $pr(w_1^N)$ and the acoustic model $pr(x_1^T|w_1^N)$.[2] The architecture of a statistical speech recognition system is depicted in Figure 2.3. It mainly consists of four components:

- **Signal Analysis**
  The signal analysis module converts an acoustic signal into a sequence of acoustic vectors x_1^T suitable for automatic speech recognition. The parameterization of the input signal should be discriminative with respect to different sounds and robust with respect to noises in the transmission channel.

- **Acoustic Model**
  The acoustic model captures the acoustic properties of speech and provides the probability of the observed acoustic signal given a hypothesized word sequence. The acoustic model includes the pronunciation lexicon, which defines the decomposition of words into subword units, and the modeling of subword units, such as phonemes, which are usually modeled context-dependent.

- **Language Model**
  The language model provides the a-priori probability of a hypothesized word sequence based on the syntax, semantics, and pragmatics of the language to be recognized.

- **Search**
  The search module finally combines the two knowledge sources acoustic model and language model and determines the word sequence that maximizes Equation 2.8.

[2] For ease of notation, we ignore the dependence of the dialogue state from the previous section.

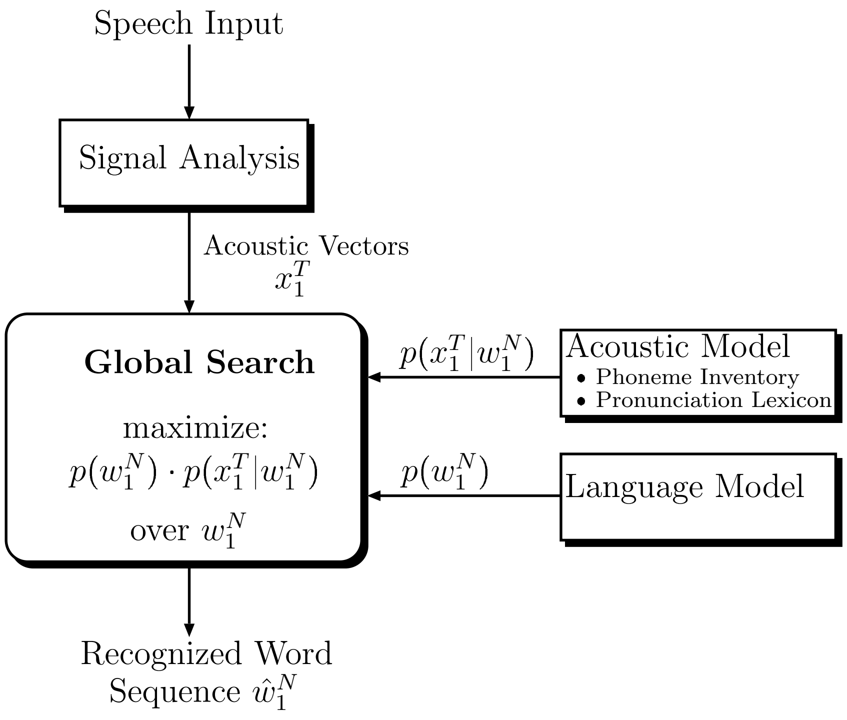


Figure 2.3: Basic architecture of a statistical automatic speech recognition system. The four main components comprise the *signal analysis*, the *acoustic model*, the *language model*, and the *global search* [Ney 90].

2.3.1 Signal Analysis

The objective of the signal analysis is to produce a parameterization of the speech input signal suitable for automatic speech recognition. Although, ideally the speech waveform should be modeled directly, today's modeling techniques are not suitable to process the raw speech waveform optimally. Signal analysis aims at separating information relevant to the recognition task from irrelevant information (e.g., speaker or channel characteristics) and at reducing the data that is presented to the speech recognizer.

The signal analysis module of the speech recognition system used in this thesis is based on a short-term spectral analysis [Rabiner & Schafer 78]. Starting point of the signal analysis is the digital representation of a speech input signal (*speech waveform*). The sampling rate of a speech waveform depends on the frequency bandwidth of the input signal. For telephone-bandwidth speech, a sampling rate of 8kHz is sufficient. The speech waveform is preprocessed using pre-emphasis in order to compensate for glottis mute and the high-pass filtering caused by the lips [Wakita 73]. After pre-emphasing, the waveform is further processed by computing the *Mel frequency cepstral coefficients (MFCC)* [Davis & Mermelstein 80], which requires a Fourier analysis of the speech waveform. The Fourier analysis requires the input signal to be periodic (stationary). Since speech is a

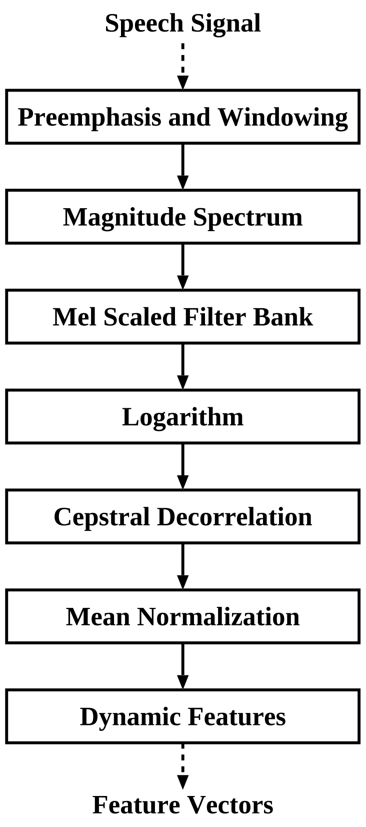


Figure 2.4: Mel frequency cepstral coefficient feature extraction.

highly non-stationary signal, the Fourier analysis must be carried out on short segments across which the speech signal is assumed to be stationary. These short segments can be obtained by applying a window function on the input signal. To avoid discontinuities caused by the truncation of the input signal, a Hamming window is used taping the input signal near zero at the borders of the window. With these segments the Fourier power spectrum of the speech waveform is computed for each time frame with a frame shift of 10ms and a window length of 25ms. The frequency axis of the power spectrum is warped according to the *Mel frequency* scale to adjust the spectral resolution to that of the human ear [Young 93]. The Mel frequency scale is linear up to 1000Hz and logarithmic thereafter. After the frequency warping, a set of overlapping Mel filters is applied. These Mel filters are made such that their center frequencies are equidistant on the Mel scale. For telephone-speech, the filter bank consists of 15 overlapping filters. The logarithm is applied to the output of the filter bank in order to reduce the dynamic range of the signal. From a physiological point of view, the logarithm mimics the nonlinear dependence between intensity and loudness of a signal as it is perceived by humans. The overlap between neighbored filters leads to a correlation between filter channels resulting in a correlation matrix that approximately has Toeplitz structure and that is decorrelated using a *discrete cosine transform* [Davis & Mermelstein 80]. In order to reduce the influence of the transmission channel, which is assumed to be constant, a mean normalization is applied to the log filter bank coefficients. The dimensionality of the cepstral vector is reduced by omitting the highest cepstral coefficients for smoothing, resulting in a total number of 12 cepstrum coefficients, where the 0th coefficient can be interpreted as the short-term energy of the speech signal.

Cepstral coefficients only describe short-term spectral properties of a signal and are therefore static quantities. However, the temporal dynamics of a speech signal are important for speech recognition [Ruske 82, Strange 83]. The temporal dynamics of the speech signal are taken into account by augmenting the cepstral feature vector with its derivatives. The derivatives are calculated by linear regression over 5 consecutive time frames. In the speech recognizer used in this thesis, the first derivative of all cepstral coefficients and the second derivative of the 0th cepstral coefficient are computed, resulting in a $12+12+1 = 25$ dimensional observation vector. Finally, a linear discriminant analysis is applied (see Section 2.3.6). So far, this is the only step in the signal analysis chain that takes class information into account.

2.3.2 Acoustic Modeling

The objective of acoustic modeling is to provide a stochastic model for $pr(x_1^T|w_1^N)$. The acoustic model is a concatenation of acoustic models for words or basic subword units the speech recognition system utilizes according to a pronunciation lexicon. With recognition tasks of increasing complexity and due to a limited number of training data it is unlikely that each word of the pronunciation lexicon can be observed sufficiently frequent during training. Therefore, words are split into subword units of which the most common ones are (context-dependent) phonemes. A *phoneme* is the smallest contrastive unit in the sound system of a language. In [Menzerath & de Lacerda 33], it was shown that the phonetic variants of a phoneme (*allophones*) are highly context-dependent, that is, the acoustic realization of a phoneme depends on its surrounding phonemes. In large vocabulary continuous speech recognition systems the mostly used subword units are phonemes in a context of one or two adjacent phonemes, so-called *triphones* or *quinphones*, respectively.

The acoustic realizations of a subword unit differ significantly with the speaking rate. To model different speaking rates, each context-dependent phoneme is represented by a *hidden Markov model (HMM).* An HMM is a stochastic finite state automaton consisting of a number of states and transitions between the states. Both the states and transitions are associated with probability distributions that can be derived from $p(x_1^T|w_1^N)$ by introducing an unobservable (hidden) sequence of states:

$$p(x_1^T|w_1^N) = \sum_{s_1^T:w_1^N} p(x_1^T, s_1^T|w_1^N) \ . \tag{2.9}$$

Here, the sum is taken over all possible state sequences that are consistent with the given word sequence w_1^N. Using the chain rule for joint probabilities yields

$$p(x_1^T|w_1^N) = \sum_{s_1^T:w_1^N} \prod_{t=1}^{T} p(x_t|x_1^{t-1}, s_1^t; w_1^N) \cdot p(s_t|x_1^{t-1}, s_1^{t-1}; w_1^N) \ . \tag{2.10}$$

The probabilities $p(x_t|x_1^{t-1}, s_1^t; w_1^N)$ and $p(s_t|x_1^{t-1}, s_1^{t-1}; w_1^N)$ are assumed not to depend on previous observations but only on the current state (Markov assumption) and on the immediate predecessor state (first-order model):

$$p(x_1^T|w_1^N) = \sum_{s_1^T:w_1^N} \prod_{t=1}^{T} p(x_t|s_t; w_1^N) \cdot p(s_t|s_{t-1}; w_1^N) \ . \tag{2.11}$$

Thus, the probability is split into the acoustic *emission probability* $p(x_t|s_t; w_1^N)$ denoting the probability to observe an acoustic vector x_t while being in state s_t and the *transition probability* $p(s_t|s_{t-1}; w_1^N)$ for transiting from state s_{t-1} to state s_t. For the ease of computation, the sum is often replaced by the maximum, which is called Viterbi or Maximum approximation [Ney 90]:

$$p(x_1^T|w_1^N) = \max_{s_1^T:w_1^N} \prod_{t=1}^{T} p(x_t|s_t; w_1^N) \cdot p(s_t|s_{t-1}; w_1^N) \ . \tag{2.12}$$

The Viterbi approximation has the advantage that Equation 2.12 can be solved in log-space, thus, reducing the dynamic range of a product of already small probabilities. Equations 2.11 and 2.12 can be efficiently solved using the forward-backward algorithm [Baum 72, Rabiner & Juang 86] or dynamic programming [Bellman 57, Viterbi 67, Ney 84]. An example of an HMM in Bakis topology [Bakis 76] is depicted in Figure 2.5.

Often, the emission probability $p(x_t|s_t; w_1^N)$ is modeled by Gaussian mixture densities:

$$p(x|s; w_1^N) = \sum_{l=1}^{L_s} c_{sl} \mathcal{N}(x|\mu_{sl}, \Sigma; w_1^N) \ , \tag{2.13}$$

where L_s denotes the number of mixtures and c_{sl} denotes the mixture weights with the constraint $\sum_{l=1}^{L_s} c_{sl} = 1$, and $\mathcal{N}(x|\mu, \Sigma)$ denotes the normal distribution with mean μ and covariance Σ.

2.3.3 Language Modeling

The language model $pr(w_1^N)$ provides an a-priori probability for a word sequence $w_1^N = w_1, \ldots, w_N$. The syntax, semantics, and pragmatics of the language to be processed are implicitly covered by this statistical model. Because of the unlimited number of possible word sequences it is not feasible to estimate $pr(w_1^N)$ for each word sequence given a finite training set. Therefore, an additional modeling step is necessary that models each word

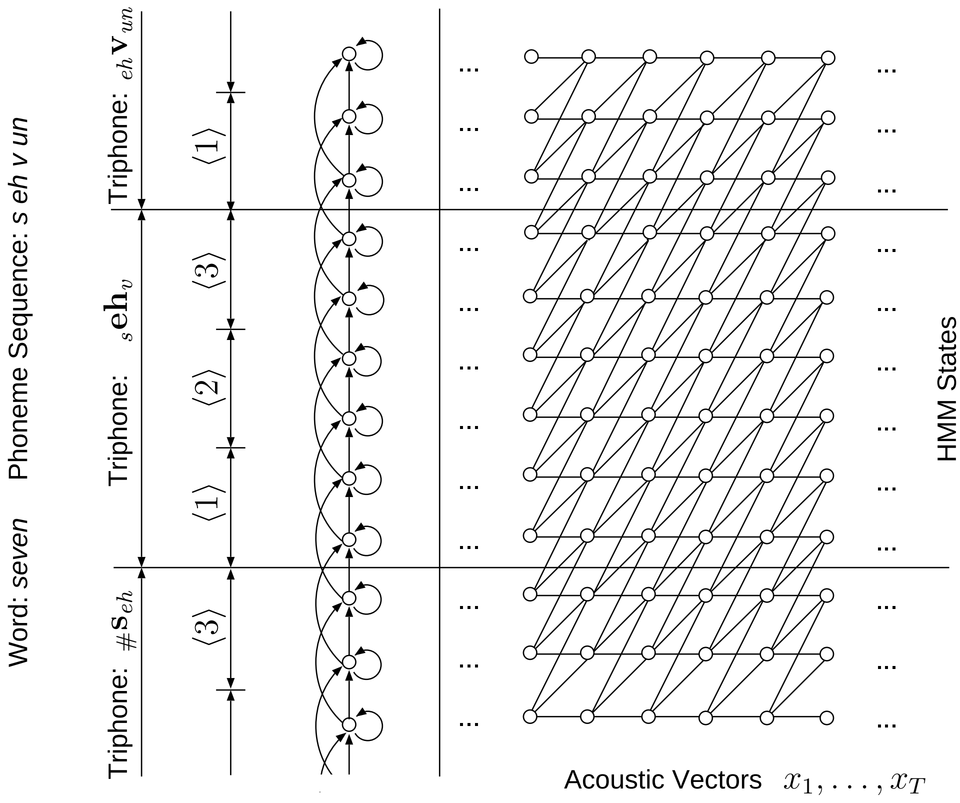

Word: seven
Phoneme Sequence: s eh v un
Triphone: $\#\mathbf{s}_{eh}$
Triphone: $_{s}\mathbf{eh}_{v}$
Triphone: $_{eh}\mathbf{v}_{un}$
⟨3⟩
⟨1⟩
⟨2⟩
⟨3⟩
⟨1⟩
HMM States
Acoustic Vectors $x_1, \ldots, x_T$

Figure 2.5: 6-state hidden Markov model in Bakis topology. Each state provides a loop, a forward, and a skip transition. Two contiguous states form a segment denoted by ⟨1⟩, ⟨2⟩, and ⟨3⟩.

sequence as a sequence of so-called *m-grams*. For m-gram language models, it is assumed that a word at position n only depends on its $(m-1)$ predecessor words. Using an $(m-1)$th order Markov process, the prior probability $pr(w_1^N)$ can be decomposed as follows:

$$\begin{aligned} pr(w_1^N) &= \prod_{n=1}^{N} pr(w_n|w_1^{n-1}) \\ &\underset{\text{Markov}}{=} \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \; . \end{aligned} \tag{2.14}$$

The predecessor words $h = w_{n-m+1}^{n-1} = w_{n-m+1}, \ldots, w_{n-1}$ are called the history of word w_n. The number of possible m-grams increases exponentially with the history length $m-1$. Thus, a considerable amount of m-grams will be unseen during training or have too few observations for a reliable estimation of $p(w|h)$. Therefore, smoothing methods have to be applied. A thorough investigation on different smoothing techniques can be found in

[Chen & Goodman 98].

A commonly used measure for the evaluation of language models is the *perplexity* PP

$$\mathrm{PP} = \left[\prod_{n=1}^{N} p(w_n | w_{n-m+1}^{n-1}) \right]^{-1/N} , \tag{2.15}$$

which can be interpreted as the average number of choices to continue a word sequence w_{n-m+1}^{n-1} at position n. Its logarithm is equal to the entropy of the model.

A variant of m-gram language models are the so-called *class-based* m-gram models. For a class-based m-gram model, the conditional probability of a word w_n based on its history is decomposed into the product of the probability of word w_n given class k_n and the probability of the class given the preceding classes:

$$p(w_n | w_{n-m+1}^{n-1}) = p(w_n | k_n) \cdot p(k_n | k_{n-m+1}^{n-1}) . \tag{2.16}$$

Equation 2.16 is a generalization of an ordinary m-gram language model because if we map each word onto a separate class, we obtain the usual m-gram model. Class-based m-grams can provide a more robust estimation of the conditional probability, especially if the training corpus used is very small. The classes can be chosen application-specific. A typical example is an airline information system where all occurring city names can be mapped onto the class City.

2.3.4 Search

The decision on the spoken word sequence is made during a search process by determining the word sequence that maximizes the product of the two knowledge sources acoustic model and language model given the sequence of observation vectors $x_1, \ldots, x_T$. If the language model is given by an m-gram and the acoustic model is an HMM then the following optimization problem has to be solved:

$$\begin{aligned} \hat{w}_1^N &= \underset{w_1^N}{\operatorname{argmax}} \left\{ \left[\prod_{n=1}^{N} p(w_n | w_{n-m+1}^{n-1}) \right] \cdot \left[\sum_{s_1^T : w_1^N} \prod_{t=1}^{T} p(x_t | s_t; w_1^N) \cdot p(s_t | s_{t-1}; w_1^N) \right] \right\} \\ &= \underset{w_1^N}{\operatorname{argmax}} \left\{ \left[\prod_{n=1}^{N} p(w_n | w_{n-m+1}^{n-1}) \right] \cdot \left[\max_{s_1^T : w_1^N} \prod_{t=1}^{T} p(x_t | s_t; w_1^N) \cdot p(s_t | s_{t-1}; w_1^N) \right] \right\} . \end{aligned} \tag{2.17}$$

As in all search problems, the search can be organized in two different ways: a *depth-first* and a *breadth-first* search. The depth-first strategy is used by the *A^*-search* or *stack-decoding* algorithm. Here the state hypotheses are expanded time-asynchronously dependent on a heuristic estimate of the cost to complete the path [Jelinek 69, Paul 91]. The performance of the A^*-search relies strongly on the quality of this estimate; the convergence to a global optimum is guaranteed if the estimate is a lower bound of the true costs. Additionally, the search space is minimal if the estimate is equal to the true costs.

The breadth-first search design is used by the *Viterbi* search where all state hypotheses are expanded time-synchronously [Vintsyuk 71, Baker 75, Sakoe 79, Ney 84]. In this approach the probabilities of all hypotheses up to a given time frame are computed and thus can be compared to each other. This allows to reduce the search space significantly by pruning unlikely hypotheses early in the search process. Especially in the breadth-first approach an efficient pruning is necessary as the number of possible word sequences grows exponentially in its length N. Thus, a full optimization of Equation 2.17 is only feasible for small vocabulary sizes. For large vocabulary sizes approximations have to be made. Instead of finding the exact optimal solution of Equation 2.17, the goal is changed to find a sufficiently good solution with much less effort. In the so-called *beam-search*, only that fraction of the hypotheses is expanded whose likelihood is sufficiently close to that of the best hypothesis of the given time frame [Lowerre 76, Ney & Mergel$^+$ 87, Ortmanns & Ney 95]. Beam-search does not guarantee to find the globally best word sequence. This optimal sequence may have been pruned at an intermediate search stage due to a poor likelihood. However, if the pruning parameters are adjusted properly no significant search errors occur and the search effort is reduced considerably.

2.3.5 Word Graphs and N-best Lists

Instead of generating only the first-best recognition result, the most likely sentence hypotheses can be combined in a word graph. A word graph is a directed acyclic weighted graph where each edge is labeled with a word hypothesis together with a score and each node is labeled with a time index.[3] Additionally, word graph nodes used in this thesis also include language model histories. Each word graph has a unique start node called source and a unique end node called sink. A path from the source to the sink defines a sentence hypothesis. An example is depicted in Figure 2.6.

Word graphs have proven useful as interface between continuous speech recognition and language understanding [Oerder & Ney 93]. They are often used as intermediate representations in multi-pass decoders where the employment of more complex knowledge sources is difficult or not feasible at all. Incorporating more complex models into the search process often yields higher recognition performance. However, this performance gain usually leads to a significant loss in decoding efficiency since the search space increases if more complex models are used. Besides the performance, the efficiency of a decoder is of prime importance for a spoken dialogue system because the dialogue system must respond in realtime to the user's request. Therefore, a trade-off between the complexity of a model and the performance gained with this model must be found. A reasonable trade-

[3] In some text books, this structure is called a word lattice. In [Schukat-Talamazzini 95], the difference between a word lattice and a word graph is described such that the latter one lacks the time information and only provides the adjacency information. However, this distinction is not done in this thesis: here, a word graph always includes time information.

off can be achieved if a multi-pass decoder is used where in the first pass a simple model is used that allows for an efficient decoding at the expense of a suboptimal recognition result. The first pass generates a word graph of the most likely word hypotheses. In the second pass, the preselected word hypotheses are rescored using a more complex model. If proper models for preselecting and rescoring are chosen the decoding will be efficient at the expense of only a small degradation in word accuracy.

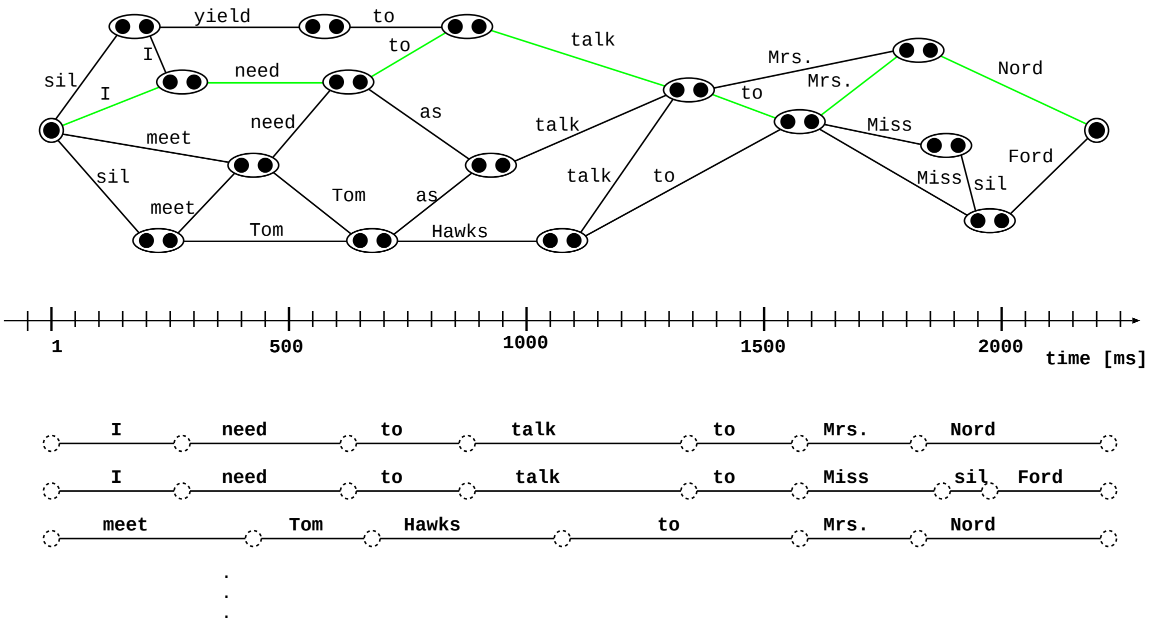


Figure 2.6: A word graph for the utterance ”I need to talk to Mrs. Nord”. The arcs of the graph represent words that are hypothesized during search. Each arc is associated with a word hypothesis and a cost vector containing at least the acoustic score and the language model score. The nodes contain the language model history and describe start and end times of word hypotheses. The N-best list can be extracted from the graph using an A^* search. The dashed circles are only included in order to find the corresponding arcs in the graph and are not part of the N-best list.

An alternative intermediate representation are N-best lists that contain the most probable N sentence hypotheses with respect to the underlying models [Chow & Schwartz 89]. N-best lists can be easily extracted from word graphs using an A^* search. Setting the rest costs to the forward scores of the graph’s edges while recombining sentence hypotheses with the same word sequence but different time alignments provides a list of the most probable, pairwise different sentence hypotheses. Compared to word graphs, N-best lists allow for an easy integration of sentence-based knowledge sources at the expense of a much more redundant and inefficient representation. As the most likely candidate solutions of the same decoding problem, N-best entries often differ only in a few word positions.

Figure 2.6 shows an example of a word graph and an N-best list.

2.3.6 Linear Transformations

Linear transformations are an important tool in pattern recognition tasks in general and in automatic speech recognition in particular. Typical linear transformations used in speech recognition are the linear discriminant analysis and the maximum likelihood linear regression. While the linear discriminant analysis aims at reducing the dimensionality while keeping those directions that are useful for discriminating the classes, the maximum likelihood linear regression method aims at adapting the system to different speakers and to varying environmental conditions.

Linear Discriminant Analysis

The design of a good classifier becomes more difficult as the dimensionality of the input space increases. One way of dealing with this problem is to preprocess the data so as to reduce its dimensionality before applying a classification algorithm. The *linear discriminant analysis (LDA)* introduced by [Fisher 36] aims to achieve an optimal linear dimensionality reduction by maximizing the proportion of the difference between the projected class means (between-class scatter) and the intra-class means (within-class scatter).

Feature Space Maximum Likelihood Linear Regression

A speaker-independent automatic speech recognition system has to cope with many variabilities in the acoustic signal. For example, varying transmission channels, noise, multiple speakers, and different speaking styles are sources of such variabilities. Therefore, the training material of such systems usually contains a wide range of different acoustic conditions. However, if the testing conditions strongly differ from the training conditions, the trained models do not fit well resulting in an increase in error rate. One way to cope with this mismatch conditions is to adapt the acoustic model to the testing conditions. An approach that is virtually used in all state-of-the-art speech recognition systems is called *maximum likelihood linear regression (MLLR)* and was proposed by [Leggetter & Woodland 95]. An affine transformation is applied to the mean vectors of the acoustic emission probabilities of the HMM in order to shift them to the “true” mean vectors of the testing conditions. This requires that the testing utterances are known. Since the word sequence of a testing utterance is usually unknown, we can approximate the correct sequence of words by a first-pass recognition result and use this to compute the affine transformation and to adapt the mean vectors, which are then used in a second pass recognition. If only one global transformation matrix is used for all mean vectors, the transformation can be equivalently applied to the observation vectors, which is called *feature-space maximum likelihood linear regression (F-MLLR).*

2.4 Natural Language Understanding

Natural language understanding aims at extracting all the information from an input sentence that is relevant to a specific task. If speech is used as means of input, natural language understanding is often referred to as spoken language understanding. If we want to automatically acquire meaning, we have to face two problems that in general cannot be treated independently: first, we need to define the meaning of a sentence and second, we need a meaning representation. Unfortunately, there is no generally accepted theory on what the meaning of an arbitrary input sentence should be. Therefore, language understanding systems are usually task-dependent and are only designed for limited application domains where the meaning of a user query can be described in a relatively simple manner.[4] The other principal characteristic of natural and spoken language understanding systems is the need for a meaning representation so that, once the meaning has been determined, these representations can be used in reasoning tasks. A first notion would be to use the sentence itself as a representation of its meaning. Although it is sometimes argued that this is not feasible because words can have multiple meanings [Allen 95][5], the main counterargument to this approach is its lack of generalization due to the limited size of training corpora: meaning representations for limited domains provide a certain structure to the problem of language understanding, which reduces the space of possible interpretations. If corpus-based methods are employed for natural language understanding, we usually have corpora of the size of at most some thousand sentences, which is still not sufficient to observe the most likely verbalizations of a user's intention and to generate appropriate replies to the user. In the context of spoken dialogue systems, intermediate meaning representations help to subdivide the dialogue task into several subproblems and to structure the information provided by the user so that replies can be generated from the current system state. Furthermore, if natural language understanding is investigated independently from a dialogue system, the meaning representation can be viewed as the result of a classification process. However, if the size of training corpora increases, using sentences themselves as meaning representations should be reconsidered.

Dependent on the complexity of the understanding task, different meaning representations have been proposed, among them the first order predicate calculus [Allen 95], case frames [Issar & Ward 93], semantic frames [Bennacef & Bonnea-Maynard$^+$ 94], semantic networks [Bates & Bobrow$^+$ 94], and concept representations [Miller & Bobrow$^+$ 94b, Levin & Pieraccini 95]. Concept representations have proven to be very effective in natural language understanding tasks and will also be used in this thesis. A concept is defined as the smallest unit of meaning that is relevant to a specific task [Levin & Pieraccini 95].

[4] For spoken dialogue systems, a way out to this problem could be to define the meaning of an utterance directly by means of the action the dialogue system shall perform. This could help to circumvent the need for an intermediate meaning representation. However, as will be discussed later, this requires large training corpora.

[5] The counterarguments stated in [Allen 95] do not apply here, because we are more interested in the meaning of phrases and sentences rather than the meaning of isolated words.

Concept representations can be used in a hierarchical manner (*nested concepts*) or in a non-hierarchical manner (*flat concepts*). Using flat concepts, natural language understanding can be described as a conversion from words w_1^N into a sequence of concepts c_1^M. Formally, the objective can be described as follows: given an input utterance represented as a sequence of acoustic vectors $x_1^T = x_1, \ldots, x_T$, we search for the sequence of concepts $\hat{c}_1^M = \hat{c}_1, \ldots, \hat{c}_M$ that has the highest probability among all possible concept sequences

$$\hat{c}_1^M = \operatorname*{argmax}_{c_1^M, M} \left\{ p(c_1^M | x_1^T) \right\} \tag{2.18}$$

$$= \operatorname*{argmax}_{c_1^M, M} \left\{ \sum_{w_1^N} p(c_1^M, w_1^N | x_1^T) \right\} \tag{2.19}$$

$$= \operatorname*{argmax}_{c_1^M, M} \left\{ \sum_{w_1^N} \underbrace{p(c_1^M | w_1^N)}_{\text{NLU}} \cdot \underbrace{p(w_1^N | x_1^T)}_{\text{ASR}} \right\}, \tag{2.20}$$

where we assume that $p(c_1^M | w_1^N)$ is independent of the sequence of acoustic vectors. Here, $p(w_1^N | x_1^T)$ denotes the common statistical approach in automatic speech recognition (ASR) and $p(c_1^M | w_1^N)$ is the statistical natural language understanding (NLU) component. Although both tasks are not independent, Equation 2.20 is often decoded sequentially,

$$\hat{c}_1^M = \operatorname*{argmax}_{c_1^M, M} \left\{ p(c_1^M | \hat{w}_1^N) : \hat{w}_1^N = \operatorname*{argmax}_{w_1^N} \left\{ p(w_1^N | x_1^T) \right\} \right\}, \tag{2.21}$$

that is, in the first step the optimal word sequence $\hat{w}_1^N$ is determined that maximizes $p(w_1^N | x_1^T)$, which is then passed to the NLU component. Since this chain of decoding steps is only suboptimal, the search for the optimal concept sequence $\hat{c}_1^M$ can be carried out on word graphs or N-best lists [Zue & Glass$^+$ 91, Oerder & Ney 93].

In order to perform an SQL query in a spoken dialogue system, it is not sufficient to determine the sequence of concepts, but also the values of the concepts, which are called *attributes*. Each concept is associated with a (possibly empty) set of attributes. An attribute is a pair consisting of a name and a value and is sometimes referred to as *slot*. The mapping from words to concepts and attributes is usually done using (partial) parsing [Seneff 92, Kellner & Rueber$^+$ 97, Minker & Waibel$^+$ 99], but also methods derived from statistical machine translation can be employed [Epstein & Papineni$^+$ 96, Macherey & Och$^+$ 01].

2.5 Dialogue Management and Strategies

The goal of the dialogue manager is to change the system from some initial uniformed state to a sufficiently informed state, such that the user's information need can be satisfied [Young 00]. Dependent on the sequence of concepts provided by the natural language understanding component and the dialogue history, the dialogue manager must choose that action that leads as quickly as possible to a final state that is likely to meet the user's request. If each dialogue action and state change are assumed to depend only on the current dialogue state, then the system can be modeled as a *Markov decision process (MDP)* [Levin & Pieraccini 97, Levin & Pieraccini$^+$ 98]. An MDP is a special case of the more general framework of *reinforcement learning* problems, where an *agent* (here: the dialogue system) tries to solve an optimization problem by finding an optimal policy that determines for each state the action he must perform in order to reach his goal. In contrast to supervised learning techniques, where functions are approximated by presenting correct input-output patterns, the knowledge about the correct output remains unknown. Therefore, the agent must interact with his *environment* (here: the user) and learns the optimal policy by trial and error.

Let S denote a set of states, A denote a set of actions, and R a set of rewards. Further, denote $t = 0, 1, 2, \ldots$ discrete points in time. Denote s_t the current state of the dialogue system and $\mathcal{A}(s_t)$ the set of possible actions available for state s_t. If the dialogue manager chooses action $a_t \in \mathcal{A}(s_t)$, the state will change to s_{t+1} and the dialogue system gets the reward r_{t+1}. In general, the reward and the successor state are modeled by a probability density function

$$pr(s_{t+1}, r_{t+1}|s_0^t, a_0^t, r_1^t) = p(s_{t+1}, r_{t+1}|s_t, a_t) \ , \tag{2.22}$$

where we assume the system to satisfy the Markov property. A Markov decision process is fully defined by the set of states, the action set, the transition probability $p(s_{t+1}|s_t, a_t)$, and the expected reward $\mathcal{R}_{t+1}(s', s, a)$ with

$$\begin{aligned} \mathcal{R}_{t+1}(s_{t+1}, s_t, a_t) &= E\{r_{t+1} \mid s_{t+1}, s_t, a_t\} & (2.23) \\ &= \sum_{r_{t+1}} r_{t+1} \cdot p(r_{t+1} \mid s_{t+1}, s_t, a_t) \ . & (2.24) \end{aligned}$$

A *policy* $\pi(a|s)$ is a probability distribution over states and actions and models the probability that action a is chosen in state s. Denote $[s_0^T, a_0^{T-1}] = s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$ a path through the state-action space, then the probability of this path can be factorized as follows:

$$p(s_0^T, a_0^{T-1}) = \prod_{t=0}^{T-1} p(s_{t+1}, a_t|s_t) = \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \cdot \pi(a_t|s_t) \ , \tag{2.25}$$

with $p(s_{t+1}, a_t|s_t)$ being the probability introduced in Equation 2.1, where we neglect the user utterance x_1^T. The goal is to maximize the expected reward in the long run, that is, we compute the expectation of rewards over paths times the probability of the path. This can be expressed by the following value functions:

State value function

$$\begin{aligned} V_\pi(s) &= \underset{\substack{[s_0^\infty, a_0^\infty]: \\ s_0=s}}{E_\pi} \left\{ \mathcal{R}(s_0^\infty, a_0^\infty) \right\} \\ &= \sum_{\substack{[s_0^\infty, a_0^\infty]: \\ s_0=s}} \left\{ \left(\sum_{\tau=0}^{\infty} \gamma^\tau \cdot \mathcal{R}_{\tau+1} \right) \cdot \left(\prod_{i=0}^{\infty} p(s_{i+1} \mid s_i, a_i) \cdot \pi(a_i|s_i) \right) \right\} , \end{aligned} \tag{2.26}$$

State-action value function

$$\begin{aligned} Q_\pi(s,a) &= \underset{\substack{[s_0^\infty, a_0^\infty]: \\ s_0=s, a_0=a}}{E_\pi} \left\{ \mathcal{R}(s_0^\infty, a_0^\infty) \right\} \\ &= \sum_{\substack{[s_0^\infty, a_0^\infty]: \\ s_0=s, a_0=a}} \left\{ \left(\sum_{\tau=0}^{\infty} \gamma^\tau \cdot \mathcal{R}_{\tau+1} \right) \cdot \left(\prod_{i=0}^{\infty} p(s_{i+1} \mid s_i, a_i) \cdot \pi(a_i|s_i) \right) \right\} , \end{aligned} \tag{2.27}$$

where a discount factor γ with $0 \leqslant \gamma < 1$ is introduced in order to handle infinite paths. This simplifies the modeling of dialogue transactions where the duration of the dialogue is a priori unknown. The optimal strategy can be found by maximizing the expected reward. With the Bellman equations for value functions

$$V_\pi(s_0) = \sum_{a_0 \in \mathcal{A}(s_0)} \pi(a_0|s_0) \cdot \sum_{s_1} p(s_1 \mid s_0, a_0) \cdot \left\{ \mathcal{R}_1 + \gamma V_\pi(s_1) \right\} \tag{2.28}$$

$$Q_\pi(s_0, a_0) = \sum_{s_1} p(s_1 \mid s_0, a_0) \cdot \left\{ \mathcal{R}_1 + \gamma \sum_{a_1 \in \mathcal{A}(s_1)} Q_\pi(s_1, a_1) \right\} , \tag{2.29}$$

the optimal value functions are defined as follows:

$$\begin{aligned} \hat{V}(s) \quad &= \max_\pi \; V_\pi(s) \qquad && \forall s \in S \\ \hat{Q}(s,a) &= \max_\pi \; Q_\pi(s,a) && \forall s \in S, a \in \mathcal{A}(s) \; . \end{aligned} \tag{2.30}$$

This leads to the optimal strategy $\hat{\pi}$ given by:

$$\hat{\pi}(s) = \underset{a}{\operatorname{argmax}} \left\{ \hat{Q}(s,a) \right\} . \tag{2.31}$$

If the transition function is unknown, methods based on sampling actual dialogues can be used [Young 02]. *Temporal difference learning* is a technique in which the Q function is updated after every dialogue turn $(s,a) \to (s',a')$ by comparing the actual one-step reward with the reward predicted by the Q function

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \left[r(s,a) + \gamma Q(s',a') - Q(s,a) \right] \; , \tag{2.32}$$

where α determines the learning rate. In order to ensure that Q is considered over all reasonable combinations of (s, a), it is necessary to allow the dialogue to deviate from the optimal policy occasionally. This can be achieved by adopting a stochastic ε-soft policy:

$$\pi(s,a) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|\mathcal{A}(s)|} & \text{if } a = \hat{a} \\ \dfrac{\varepsilon}{|\mathcal{A}(s)|} & \text{otherwise.} \end{cases} \tag{2.33}$$

A problem with the MDP approach is the assumption that the state space is fully observable. This assumption is unrealistic since at least the user state is unknown. Additionally, due to recognition errors, the system state is also uncertain. Modeling this uncertainty in a principled way leads to the so-called *partially observable Markov decision process (POMDP)* [Williams & Poupart$^+$ 05].

2.6 Natural Language Generation

Natural language generation is essentially the inverse of the natural language understanding process. A natural language generation module can be implemented via templates. Here, a template is a predefined natural language sentence that, besides natural language words, also contains some variables whose values will be extracted from the current dialogue state. While writing individual templates is convenient, they may not scale well to more complex domains.

Alternatively, a probabilistic model can be defined as follows: given a dialogue action and the current dialogue state, a sequence of words v_1^N must be generated that defines the system's answer. Formally, we choose that word sequence $\hat{v}_1^N$ maximizing the following probability:

$$\hat{v}_1^N = \underset{v_1^N}{\operatorname{argmax}} \left\{ pr(v_1^N | a_t, s_{t+1}) \right\} . \tag{2.34}$$

Since state s_{t+1} contains all the knowledge the dialogue system has collected so far, it seems reasonable to use the same models for language generation that were employed for language understanding.[6] However, as pointed out in [Young 02], a generation model based on $pr(v_1^N|a_t, s_{t+1})$ lacks syntactic constraints and tends to produce ungrammatical sentences. To cope with this problem, the model can be smoothed with a simple m-gram model, leading to the modified criterion in Equation 2.35:

$$\hat{v}_1^N = \underset{v_1^N}{\operatorname{argmax}} \left\{ pr(v_1^N | a_t, s_{t+1}) + \gamma \cdot pr(v_1^N) \right\} . \tag{2.35}$$

Here, $pr(v_1^N|a_t, s_{t+1})$ produces a set of candidate sentences and $pr(v_1^N)$ selects the most likely word sequence.

[6] The dialogue state s_{t+1} especially contains the attribute value pairs associated with the concepts and the current dialogue act.

In [Ratnaparkhi 00] and [Zhou & Gao$^+$ 02], the generation of target sentences is based on a maximum entropy statistical model. The model is trained on attribute-value pairs obtained from semantic representations of input texts. This modeling process is largely domain independent and speeds up the development of systems for different application domains.

2.7 Speech Synthesis

Once the response $\hat{v}_1^N$ has been generated, it must be converted into an acoustic signal. This can be achieved by employing a large database of context-dependent phone models and searching for that phone sequence $\hat{y}_1^{T'}$ maximizing the following criterion:

$$\hat{y}_1^{T'} = \underset{y_1^{T'}}{\operatorname{argmax}} \left\{ pr(y_1^{T'} | \hat{v}_1^N) \right\} . \tag{2.36}$$

Current state-of-the art speech synthesis systems are concatenative systems consisting of three components [Allauzen & Mohri$^+$ 04].

The first component is a text analysis front end generating a sequence of feature vectors for a given input text. Typical components of a feature vector are the predicted phones, phonetic contexts, acoustic features, including pitch and duration, and prosodic features.

The second component of a concatenative speech synthesis system is the unit selection algorithm that determines for a given set of recorded acoustic units the sequence of units with minimal distance to the sequence of feature vectors predicted by the text analysis front end.

The third component produces an acoustic signal by concatenating the unit sequence determined by the unit selection algorithm.

Unit selection can be modeled statistically by defining two probability distributions: the target probability $pr(z_1^{T'} | y_1^{T'})$ that estimates how well the features y_t of a recorded unit match the specified feature vectors z_t, and the concatenation probability $pr(y_1^{T'})$ that estimates how well two units will be perceived to match when appended. This leads to the following decision rule:

$$\begin{aligned} \hat{y}_1^{T'} &= \underset{y_1^{T'}}{\operatorname{argmax}} \left\{ pr(y_1^{T'} | z_1^{T'}(\hat{v}_1^N)) \right\} \\ &= \underset{y_1^{T'}}{\operatorname{argmax}} \left\{ pr(y_1^{T'}) \cdot pr(z_1^{T'}(\hat{v}_1^N) | y_1^{T'}) \right\} , \end{aligned} \tag{2.37}$$

where $z_1^{T'}(\hat{v}_1^N)$ denotes the sequence of feature vectors produced by the text analysis front end.

2.8 Measuring System Performance

Because of the interactive nature of spoken dialogue systems, evaluating and measuring system performance is a difficult problem. An important criterion for measuring system performance is the *dialogue success rate* that describes the average proportion of successfully finished dialogues. A dialogue is considered successful if the user obtains the desired information after a finite number of dialogue turns. However, this measure is not sufficient because it does not take into account certain aspects of a dialogue, such as the duration of the dialogue, whether the dialogue system explicitly confirms information and thus extends the duration of the dialogue, and whether the system is able to detect errors and acts appropriately. This leads to a more subjective evaluation criterion for measuring the system's quality as seen from the user's point of view. Subjective measures aimed at assessing the user's opinion on the system are obtained through direct *interview by questionnaire filling*. User satisfaction is regarded as the ultimate evaluation criterion, and subjective measures are helpful to detect weak points in the design of spoken dialogue systems.

The interactive nature of spoken dialogue systems reveals another problem: since spoken dialogue systems are a realization of human-machine interfaces and therefore interact with humans by definition, a proper evaluation is only feasible in large field tests. However, interactions with humans are expensive and not always practicable during a testing phase. Furthermore, the results obtained are in general not reproducible since users behave differently if they are asked to interact with a dialogue system once more.

In order to get a more objective assessment, the individual components of a spoken dialogue system can be evaluated separately:

- **Automatic speech recognition**
  The automatic speech recognition system is evaluated according to word- and graph error rates. The *word error rate* is defined as the Levenshtein distance between the single-best recognition result and the spoken word sequence. The Levenshtein distance is computed using dynamic programming and determines the optimal alignment of the words in both sequences in terms of number of word deletions, insertions, and substitutions. The graph error rate and N-best error rate, respectively, reflects the error rate with the smallest Levenshtein distance that is obtained by an oracle-based search on word graphs or N-best lists. In general, this minimal error word sequence does not correspond to the best-scored sentence hypothesis. Graph- and N-best error rate measure to what extend the spoken word sequence is included in the graph or N-best list structure. In case of a multi-pass decoding, the graph error rate is a lower bound to the word error rate. Although the single-best word error rate is of less importance for a combined speech recognition and spoken language understanding system, it is useful to estimate the error of the speech recognition system. If confidence measures are computed, their quality can be described by *confidence error rates* and *detection error tradeoff curves*.

- **Natural language understanding**
  Similar to the word error rate for automatic speech recognition, we can define the *concept error rate* or *slot error rate* for the natural language understanding component. The slot error rate is defined as the Levenshtein distance between the single-best decoded concept sequence and a given reference concept sequence. Since each concept is associated with a set of attributes, we define the *attribute error rate* as the proportion of attributes that have the wrong value with respect to given reference attribute values.

- **Dialogue management**
  Besides the dialogue success rate, the number of dialogue turns, the frequency of confirmation questions, and the overall duration of a dialogue transaction are important. These quantities can be combined to a cost function, where the weights of the cost function are usually chosen empirically.

- **Full system**
  In order to circumvent large field tests for evaluating the full dialogue system and to make results reproducible, a system can be evaluated approximately using user simulations. Here, a user is simulated by a stochastic model that, in place of a human, interacts with the system. Over the past years, stochastic user models have found increasing attention in order to learn dialogue strategies and to evaluate dialogue systems automatically. Stochastic user models have the advantage that dialogue transactions can be generated offline and are reproducible. Therefore, studying the expressiveness and limitations of user models (i.e., answering the question what can be learned from it) has become an active research field.

Chapter 3

State-of-the-Art

The following sections describe the state-of-the-art in spoken language understanding and dialogue management. Section 3.1 provides and overview of different natural language understanding corpora and tasks that where used and investigated in the past. Section 3.2 discusses techniques used in natural language understanding and recent efforts to combine speech recognition with spoken language understanding. Section 3.3 delineates different approaches to dialogue management and dialogue strategies. Section 3.4 deals with error handling in spoken dialogue systems.

3.1 Applications and Corpora

Although several spoken dialogue systems are developed for commercial purposes and are therefore trained on in-house data, there are a few natural language corpora that are publicly available and that can be used to develop and evaluate the language understanding components of spoken dialogue systems. The domains of these corpora cover mainly scheduling and appointment tasks.

3.1.1 Air Travel Information Service (ATIS)

From 1990 - 1994, the Air Travel Information Service (ATIS) task was the subject of annual benchmark evaluations sponsored by DARPA. The task is to retrieve airline schedules, fares, and related information from a relational database for a restricted set of cities within the United States and Canada using a spoken language environment [Price 90].

3.1.2 Speech Understanding in Dialogue (SUNDIAL)

The Speech Understanding in Dialogue (SUNDIAL) project started in 1988 and ended in 1993. The goal of the SUNDIAL project was to build real-time integrated computer systems that are capable of maintaining co-operative dialogues with users over standard telephone lines [Peckham 91]. The domains covered by the SUNDIAL project were flight arrivals, schedules, and reservations. Systems were planned for four different languages: English, French, German, and Italian.

3.1.3 Automatic Railway Systems for Europe (Arise)

The goal of the Arise project, which ran from 1996 - 1998, was to develop and evaluate spoken language dialogue systems for restricted domain information [den Os & Boves$^+$ 99]. During the project, spoken dialogue systems for a train timetable information task were developed in three different languages: Italian, French, and Dutch.

3.1.4 Trains and Trips

The Trains project and its successor, Trips, started in 1991 and is an ongoing research effort on spoken dialogue systems. One of the early achievements of the Trains project was the collection of a corpus of task-oriented spoken dialogues, which is now available through the LDC. Over the course of the Trains project, a series of demonstration systems were built. The domains for these systems involved routing and scheduling of freight trains, hence the project name. An overview of the project is given in [Allen & Miller$^+$ 96] and [Ferguson & Allen$^+$ 96].

3.2 Spoken Language Understanding

Today, most state-of-the-art spoken dialogue systems perform a rule-based semantic analysis using so-called *semantic grammars*. Semantic grammars are (probabilistic) context free grammars consisting of application-dependent handcrafted rules. The first semantic grammars were originally developed for text-based dialogue systems in the domains of question answering and intelligent tutoring [Brown & Burton 75]. Besides semantic grammars, other approaches to semantic analysis were developed that are briefly overviewed in the following. However, it should be pointed out that, to a certain extent, even these methods fall back to rule-based methods, especially when dealing with numbers and time expressions.

3.2.1 Rule-based Approaches

The Tina system [Seneff 92] developed at MIT included a rule-based natural language understanding component. A set of context-free rewrite rules is first transformed into a network structure. Afterwards, probabilities derived from a set of parsed sentences are assigned to the arcs of the network. The parser uses a stack decoding search strategy with a top-down control flow.

Other rule-based systems, which were developed at LIMSI-CNRS, are the L'Atis system [Bennacef & Bonnea-Maynard$^+$ 94], which is a French version of the American Atis task, as well as the Mask [Gauvain & Bennacef$^+$ 97] and the Arise [Lamel & Rosset$^+$ 98] systems both covering train travel-based domains. All systems employ a rule-based case frame parser. A frame can be thought of as a network of nodes and relations [Minker & Waibel$^+$ 99]. The top levels of a frame are fixed and represent facts that are always true about the supposed situation. The lower levels have terminals or slots that need to be filled by specific instances of the data.

The CMU-Phoenix system [Issar & Ward 93, Ward & Issar 95] also employs a case grammar-based implementation. The parser was originally designed for extracting meaning representations from spoken utterances in the Atis task. Its strategy is to apply grammatical constraints at the phrase level rather than on the full sentence, thus, strings of phrases that do not form a grammatical English sentence are still parsable by the system. The CMU-Phoenix system participated in the the official Atis-3 Dec94 evaluation. All systems were evaluated using a measure called common answer specification (CAS), which defines the minimal and maximal SQL answer that is considered to be correct. On the Atis-3 Dec94 test set, the CMU-Phoenix system obtained a CAS of 96.2%, which was, along with AT&T's Chronus system, the best scored system for text input.

3.2.2 Statistical Approaches

Stochastic Parsing

The most employed approach to spoken language understanding is based on stochastic context free grammars (PCFG). In [Kellner & Rueber$^+$ 97] individual concepts are modeled by attributed handcrafted PCFGs. A top-down chart parser is used to search an input word graph for meaningful phrases. The result of a parse is stored in a concept graph. In order to cope with ungrammatical utterances, a partial parsing strategy is used that only parses fragments of an utterance that are likely to contain meaningful words, whereas unparsable parts of the utterance are replaced by so-called filler arcs. Experiments were conducted on the Philips automatic telephone switchboard and directory information system (Padis), which is a natural-language user interface to a telephone directory database. The authors report a concept error rate of 26.9% with an initial word error rate of 24.4% and a dialogue success rate of about 95%.

In [Minker 98], the portation of a stochastically-based method for natural language understanding from the Atis task to Mask is described. The author compares a rule-based approach with a stochastic approach for both tasks. While the stochastic approach outperforms the rule-based approach on Mask, the rule-based approach works better on the Atis task.

Markov Model-based Approaches

In AT&T's Chronus system [Levin & Pieraccini 95], a set of flat concepts is used as semantic representation. The natural language understanding component is based on an HMM-like process whose hidden states correspond to the concepts. The sequence of words generated by this model constitutes the observed sentence.[1] Along with CMU's

[1] This generative model is therefore a simple version of a translation model. In [Levin & Pieraccini 95], this model is misleadingly called a stochastic language model.

PHOENIX system, the described system achieved the best result during the official ATIS-3 evaluation with a CAS of 96.2% on the ATIS-3 Dec94 test set.

In [Miller & Bobrow$^+$ 94a, Miller & Bates$^+$ 95], a tree-structured meaning representation is used. Concepts can be nested forming a tree structure for a given input sentence. The underlying statistical model for the language understanding component is called hidden understanding model. This generative model is a two folded statistical process where in the first step the model chooses the meaning to be expressed and in the second step selects the word sequences generating that meaning. The first step corresponds to a semantic language model producing meaning expressions, the second step produces lexical realizations based on the meaning expressions. A limitation of the nested concept approach is that linguistic movement phenomena make it difficult to align words of a sentence to any tree structured meaning representation without introducing crossings. For the official ATIS-3 Dec-94 evaluation, a CAS of 90.6% was reported.

Semantic Classification Trees

In [Kuhn & de Mori 95], semantic classification trees (SCTs) are applied to natural language understanding. SCTs are specialized decision trees where the nodes contain patterns of possible input word sequences. The semantic rules are learned automatically from an annotated corpus and are robust against errors caused by the speech recognizer as well as disfluencies and ungrammatical utterances produced by the speaker. The SCT approach was integrated into the CHANNEL system developed at the Centre de Recherche Informatique de Montréal (CRIM). The authors report a CAS of 87.7% on the ATIS-3 Dec-93 corpus.

Machine Translation-based Approaches

Another approach to natural language understanding is derived from the field of statistical machine translation and was proposed in [Epstein & Papineni$^+$ 96]. Given an annotated corpus, the problem of natural language understanding is described as a translation from a source sentence to a formal language target sentence. Experiments were carried out on the ATIS-3 corpus. The target sentences are natural language paraphrases of SQL queries. Both the SQL queries and their natural language paraphrases are provided by the ATIS-3 corpus. The authors report a CAS of 72% using the IBM 1 translation model. The model used lacks local context information, which is essential for natural language understanding. This problem is addressed in [Della Pietra & Epstein$^+$ 97], where the model is refined by an additional fertility model. The authors report a CAS of 83.0% on the context-independent subcorpus of the ATIS-3 Dec93 test set.

Most of the approaches described so far are based on the source-channel paradigm and employ a generative model. A different approach to natural language understanding, which directly optimizes the posterior probability of the formal language target sentence, was introduced in [Papineni & Roukos$^+$ 97]. Similar to [Epstein & Papineni$^+$ 96], natural language understanding is regarded as translating a natural language sentence into a formal target language. To model the posterior probability, the authors define several feature functions that capture correlations between automatically determined key phrases in both languages. The features and their associated weights are selected using a training corpus of matched pairs of source and target language sentences to maximize the entropy of the resulting conditional probability model. The authors report a CAS of 85.8% for the context independent sentences on the ATIS-3 Dec94 test set.

Hidden Vector State Model

The hidden vector state model was introduced by [He & Young 03a] and is used for hierarchical semantic parsing. The model associates each state of a push-down automaton with the state of an HMM. State transitions are factored into separate stack push and pop operations and are further constrained to give a tractable search space. The model is complex enough to capture hierarchical structure and is trained on so-called abstract semantics that can be generated semi automatically from SQL queries provided by the ATIS-3 corpus. For the hidden vector state model the authors report a CAS of 91.5% on the ATIS-3 Dec94 test set. In [He & Young 03b], the method is compared with a finite state transducer approach. Both approaches show a similar CAS performance on the ATIS-3 Dec94 test set.

Table 3.1: Common answer specification (CAS) results of various systems applied on the ATIS-3 Dec93 and Dec94 test sets. All results were obtained on the context-independent sentences of the ATIS-3 test corpus, that is, answering these sentences does not require information contained in previous sentences. The table contains results that were produced on pure text data (NLU) and on speech data (SLU). Rows preceded by a ⋆ refer to results obtained during the official ATIS-3 evaluations.

	ATIS-3 Dec93		ATIS-3 Dec94	
	NLU CAS[%]	SLU CAS[%]	NLU CAS[%]	SLU CAS[%]
⋆AT&T	92.6	77.9	96.2	93.0
⋆BBN	90.4	86.2	90.6	88.1
Cambridge	–	–	91.5	86.1
⋆CMU (case frames)	94.0	91.1	96.2	92.6
CRIM (SCT)	87.7	–	–	–
IBM (hidden clump.)	83.0	–	–	–
IBM (feature-based)	86.8	–	85.8	–
LIMSI-CNRS (rules)	–	–	83.1	–
LIMSI-CNRS (stoch.)	–	–	81.3	–
⋆SRI	85.7	83.5	93.0	89.4

3.2.3 Combining Speech Recognition and Language Understanding

Combining automatic speech recognition and natural language understanding in a simple cascaded manner according to Equation 2.7 often leads to suboptimal results because the interdependence of decisions is ignored. Errors that may occur in the automatic speech recognition component, are propagated to the natural language understanding module, which then treats the incoming sentence hypotheses as observations rather than hypotheses. Therefore, methods that aim at a tighter coupling between speech recognition and natural language understanding have been studied in various articles.

Word Graphs and N-best Lists

Several methods have been proposed in order to get a tighter integration of speech recognition and language understanding. Employing word graphs and N-best lists as interface between the automatic speech recognition module and the natural language understanding component was already proposed in [Chow & Roukos 89, Oerder & Ney 93] and [Schwartz & Austin 91]. Word graphs and N-best lists provide a simple interface that allows for an easy integration of additional knowledge sources, such as semantic grammars or natural language understanding-specific language models. Once a word graph has been generated, these knowledge sources can be incorporated into the graph structure. The search for the best-scored hypothesis is carried out on the graph structure through a rescoring step resulting in a multi pass decoding strategy. In [Chow & Roukos 89], the authors describe a system for speech understanding that uses hidden Markov models for acoustic modeling, a unification grammar for the syntax of English, and a higher order intensional logic for the semantic representation. To maximize speech understanding performance, the constraints of the linguistic models are incorporated into the search process to find the most probable interpretation of the user's utterance. The approach parses a word lattice that is produced during a first pass decoding.

Transducer-based Approach

Another way to combine speech recognition and understanding is to define two finite-state transducers, one for the speech recognition component and one for the language understanding module. If both transducers are composed, one yields a finite-state transducer for a combined recognition-understanding system. Using finite-state transducers to integrate speech recognition with other areas of natural language processing is very common in machine translation. In [Vidal 97, Casacuberta & Llorens$^+$ 01, Matusov & Kanthak$^+$ 05], finite-state transducer approaches are employed in order to combine acoustic recognition with translation models. [Matusov & Kanthak$^+$ 05] show consistent improvements using this approach compared with translations obtained from single-best recognition results. Because natural language understanding can be viewed as a translation between a natural source language and a formal target language, the finite-state transducer approach can directly be applied for combining speech recognition and language understanding.

Concept-based and Grammar-based Language Models

In [Sudoh & Tsukada 05], a standard word-based m-gram language model is replaced by a word-to-concept translation model. This model can be directly integrated into an existing speech recognition framework. The authors compare their proposed method with a naïve cascaded spoken language understanding approach, where the best scored hypothesis output produced by a speech recognition system is used as input for the natural language understanding component. The authors show on a Japanese train timetable information task that their approach yields better results in a precision recall graph compared to the single-best approach.

In [Moore & Appelt$^+$ 95], a grammar-based natural language understanding system is used as additional language model, which is combined with a trigram model and a fourgram as well as a class-based fourgram model. The authors show that employing this enriched language model reduces the word error rate from 2.5% to 2.1% and also reduces the spoken language understanding error rate from 13.7% to 12.7%.

Enriching Language Understanding with Speech-based Confidence Measures

If word graphs or N-best lists produced by an automatic speech recognition system are used as intermediate representations, the natural language understanding component can incorporate the speech-based knowledge sources. In [Hazen & Burianek$^+$ 00], speech-based confidence measures are incorporated into the understanding process. The understanding grammar is augmented by additional reject and confirmation markers leading to a concept error rate reduction of 35% for the Jupiter weather information system. [Tur & Wright$^+$ 02] improve their spoken language understanding system by exploiting word-level confusion networks obtained from speech recognition word graphs. Using these networks together with confidence measures improved the classification error rate for AT&T's "How may I help you" dialogue system by 38%.

3.3 Dialogue Management and Strategies

The dialogue manager is the central component of a dialogue system where all the information from the different components, such as speech recognition and language understanding, must be collected and evaluated so that a reasonable dialogue action can be selected. The sequence of dialogue action defines the dialogue strategy of a dialogue system.

3.3.1 Domain-Independent Dialogue Management

Often, dialogue management components are designed for only one particular application. This can become a bottleneck if a dialogue system needs to be ported to a new domain. To tackle this problem. various designs have been proposed that aim at making the dialogue management component domain-independent.

Construct Algebra

In [Abella & Gorin 99], an approach for creating dialogue management systems based on a so-called *Construct Algebra* is described. The dialogue manager consists of three main components: a task knowledge representation, a construct algebra, and a collection of dialogue motivators that determine what actions need to be taken by the dialogue system. The task knowledge is encoded as a hierarchy of constructs, which themselves are represented as trees. The construct algebra defines a collection of relations and operations on these constructs. The authors apply their framework to two different tasks: AT&T's "How may I help you?" and a voice post query application.

Tree-based Representations

In [Potamianos & Ammicht$^+$ 00] and [Ammicht & Potamianos$^+$ 01], trees are used for representing semantic ambiguities, which often occur in spoken natural language tasks. The tree is constructed from the ontology of the domain. Nodes encode concepts and edges between nodes encode *is-a* or *has-a* relationships. Two types of ambiguities are modeled: value ambiguities, where the value for a particular attribute is uncertain, and positional ambiguity, where the attribute to a particular value is ambiguous. A key point of the described approach is that, similar to [Abella & Gorin 99], task knowledge, user input, and attribute value pairs are kept separate in related data structures. During a dialogue transaction, the system extracts data values from user utterances and places them in instances of the tree. Candidate attribute-value pairs are scored based on supporting evidence for or against the candidate. The described algorithms are independent of the domain. The authors report an error reduction of around 50% in a travel reservation task using their new ambiguity resolution strategy compared to an old system that lacked the described methods.

3.3.2 Learning Dialogue Strategies

If the new dialogue state-action only depends on the current dialogue state, the system can be modeled as a Markov Decision Process (MDP). Markov decision processes provide a theoretical framework for automatic learning of dialogue strategies.

Fully observable Markov Decision Processes

The application of MDPs in the context of spoken dialogue systems was described for the first time in [Levin & Pieraccini 97]. In [Levin & Pieraccini$^+$ 00], the problem of learning a dialogue strategy is interpreted as an optimization problem. Starting with a dialogue system that can be described within the MDP framework together with an objective function representing the dialogue costs, the optimal dialogue strategy can be computed using a Monte Carlo algorithm by minimizing the expected dialogue costs. A fundamental problem of employing MDPs for learning dialogue strategies is that the state space increases exponentially with the information stored in a single state. Another problem that one is confronted with arises from the huge number of example dialogue sessions that are necessary in order to compute the optimal dialogue strategy. In [Goddeau & Pineau 00], techniques are introduced that reduce the number of data and that accelerate the training of the models. A moot point with all fully observable MDPs is that they assume the current state of the environment to be known exactly. Thus, the uncertainty introduced by automatic speech recognition cannot be captured.

Partially observable Markov Decision Processes

The assumption of an always determined state is dropped for a partially observable Markov decision process (POMDP). In POMDPs, the current state of the system is not explicitly defined but can be probabilistic. Because a spoken dialogue system can make errors on different levels of the system's architecture, such as speech recognition errors, semantic interpretation errors, and so on, the current state of the dialogue system is uncertain. Therefore, POMDPs provide a more realistic framework for modeling dialogue systems compared to standard MDPs. In [Williams & Poupart$^+$ 05], the authors use POMDPs for modeling dialogue strategies. The authors show how to incorporate discrete and continuous components into the proposed framework and report significant improvements in a testbed dialogue scenario from the travel domain compared with several handcrafted dialogue managers.

3.3.3 Dialogue Simulation

Evaluating dialogue turns from real human-machine interactions is a time-consuming and costly effort. Therefore, using simulated dialogues is a reasonable alternative to evaluate dialogue systems and to learn dialogue strategies automatically. In [Eckert & Levin$^+$ 97], dialogue simulations are used for evaluation. In [Scheffler & Young 02, Scheffler 02], a simulation tool is proposed for dialogue evaluation that is applied during the design process of a dialogue system. Additionally, the simulation tool is applied in a framework for automatically designing and optimizing dialogue management strategies.

Despite the advantages of simulated dialogues, it is still unclear how to quantitatively evaluate whether simulated user responses are realistic, generalize well to unseen dialogue situations, and resemble the variety of the user population. These questions were addressed in part in [Schatzmann & Georgila$^+$ 05], where different simulation techniques are investigated and systematically tested and evaluated on real data. The authors show that none of the currently available techniques can realistically reproduce the variety of human user behavior and that simple statistical measures are sufficient to distinguish synthetic dialogues from real dialogues.

3.4 Error Handling in Spoken Dialogue Systems

For spoken dialogue systems, errors can occur on different levels of the system's architecture. One of the principal causes for errors during a dialogue session are erroneous recognition results, which often lead to incorrect semantic interpretations. Even if the speech signal has been recognized correctly, the natural language understanding component can produce erroneous sentence meanings. Therefore, mechanisms for error detection and error handling play an important role in the design of spoken dialogue systems.

3.4.1 Word Confidence Measures

State-of-the-art statistical speech recognition systems use Bayes' decision rule in order to determine the most likely sequence of word hypotheses. Typically, the score function used does not provide a measure for the certainty of the recognition result. Therefore, confidence measures can be employed that quantify this certainty. Word confidence measures based on word posterior probabilities have proven to be very successful in automatic speech recognition. In [Macherey 98, Wessel & Macherey$^+$ 98], word posterior probabilities are estimated on word graphs using a forward-backward algorithm. That way, word posterior-based confidence measures employ the same knowledge sources used for finding the best-scored hypothesis during the search process. By accumulating the posterior probabilities of all edges that share the same word index and that overlap with each other, the problem of different time alignments for the same sentence hypotheses is mitigated. In [Wessel & Macherey$^+$ 98], the authors report a reduction of the confidence error rate ranging between 15% and 30% for different speech recognition corpora.

In [Kemp & Schaaf 97], posterior probabilities used as confidence measures yield the largest performance. Similar to [Wessel & Macherey$^+$ 98], the posterior probability is estimated on word graphs using a forward backward-algorithm. However, the exact algorithm is not further specified. For the German Verbmobil corpus, the authors report a classification error reduction of 18.4%.

3.4.2 Semantic Confidence Measures

For spoken language understanding, word confidence measures can be augmented by semantic features. In [Sarikaya & Gao$^+$ 03], tag and arc probabilities obtained from a statistical classer and parser tree as well as probabilities estimated from a maximum entropy-based semantic structured language model are used as semantic features. Experiments were conducted on an air travel domain. Together with the word posterior probability, the authors report an improvement of around $13-14\%$ for correct acceptance at a fixed false acceptance rate of 5%.

3.4.3 Multi-Level Error Handling

In [Komatani & Kawahara 00], confidence measures for spoken dialogue systems are proposed for two levels: the first level comprises confidence measures for content words based on N-best lists, the second level comprises confidences measures for semantic attributes based on inverse document frequencies. The content word-based confidence measure directly controls the confirmation strategy of the dialogue manager. For this, the authors define two thresholds. Depending on the interval the confidence value is located, the user utterance is accepted, explicitly confirmed, or rejected. Although the proposed ideas are reasonable, an analysis of the false acceptance rate and slot error rate shows contradictory results where the slot error rate increases if an N-best list is used and a confirmation strategy is employed.

In [Macherey & Bender$^+$ 03], confidence measures are determined on the speech recognition level and the semantic level. Since the natural language understanding component is based on a maximum entropy framework, the word posterior-based confidence measures can directly be used as additional features. Both confidence values are propagated to the dialogue manager who then determines whether the utterance should be accepted, implicitly, or explicitly confirmed. The authors report a dialogue success rate of 88.6%.

In [Litman & Walker$^+$ 99], an approach similar to classification trees is used in order to detect erroneous recognitions. The approach uses several features, including confidence measures from the speech recognition unit, features measuring dialogue efficiency, dialogue quality features that take the number of played rejection prompts and played timeout prompts into account, and textual features.

Chapter 4

Scientific Goals

The scientific goals addressed in this thesis are outlined in the following. A reference pointing to a more detailed description is given at the end of each goal.

- **Fast confidence computation for realtime speech recognition systems**
  We investigate confidence measures for automatic speech recognition that are suitable for spoken dialogue systems. Similar to [Wessel & Macherey$^+$ 98] the computation is based on word graphs. However, in contrast to [Wessel & Macherey$^+$ 98, Wessel 02], we do not interpret word graphs in a time-conditioned manner, but keep the word-conditioned structure, which results in a decreased number of computational steps. The word graphs are an exact representation of all hypotheses that have survived the pruning. The construction of word graphs during search is organized such that unreachable hypotheses are automatically eliminated during the search process. Hence, no time consuming postprocessing steps for eliminating dead paths are necessary. This reduced computational effort makes posterior confidence measures especially suitable for spoken dialogue systems. We show that keeping the word-conditioned graph structure yields a similar performance reported in [Wessel & Macherey$^+$ 98]. Details can be found in Section 5.3.

- **F-MLLR for realtime speech recognition systems**
  We investigate how the F-MLLR transformation can be successfully applied within a spoken dialogue system where adaptation data is limited and a full second pass decoding is not feasible due to realtime constraints. We compare two approaches: an incremental F-MLLR that adapts on data received from the previous utterance and a word graph-based approach that is based on acoustic rescoring. We also show that speech recognition-based confidence measures are beneficial for applying F-MLLR adaptation in spoken dialogue systems. Details can be found in Section 5.4.

- **Natural language understanding is a machine translation problem**
  We show that the problem of natural language understanding can be described as a special machine translation problem where often no reordering is necessary. We investigate two approaches: a source channel-based approach and a direct model based on maximum entropy. In contrast to [Epstein & Papineni$^+$ 96] where almost

no context was taken into account, we show that local contexts are crucial to this task. The maximum entropy approach to natural language understanding described in [Papineni & Roukos$^+$ 97] requires a decoder that supplies a list of candidate translations on which then the maximum entropy model is trained. In this thesis, we directly train the maximum entropy model on bilingual training data. Thus, no additional decoding step is necessary for the training procedure. Results presented in [Papineni & Roukos$^+$ 97] were produced on pure text data. In this thesis, we also investigate the robustness of both approaches if speech is used as input modality. The tree-structured concepts defined in [Miller & Bobrow$^+$ 94a, Miller & Bates$^+$ 95] can introduce so-called crossing effects. Therefore, we use flat unstructured concepts, which are easier to handle and more suitable for machine translation tasks. We empirically show that flat concepts are sufficient for modeling different natural language understanding tasks. Details can be found in Chapter 6.

- **Tighter coupling between spech recognition and language understanding**
  Although many authors have investigated enriching natural language understanding with speech recognition features in order to get a tighter coupling between both tasks, the combinations do not aim at minimizing an objective function, such as the number of word errors or slot errors. In this thesis, we employ the minimum error training algorithm in order to combine speech recognition and natural language understanding features such that the error criterion used is minimized. A related approach was investigated in [Matusov & Kanthak$^+$ 05] in the context of machine translation. Here, we use a different optimization technique that computes for each feature function the corresponding error surface. Furthermore, we investigate a larger set of feature functions that take also sentence-based features into account. Details can be found in Section 6.8.

- **Domain-independent dialogue management**
  We present a new approach to domain-independent dialogue management where trees are used in order to represent the task knowledge. The operations of the dialogue manager can be entirely formulated as cost functions that operate directly on nodes, paths, and trees. We investigate the capability of the system to choose those actions that are likely to lead as directly as possible through the tree to the user's goal and show that different knowledge sources can be easily integrated into this approach without the necessity to rewrite the system. A related approach was proposed in [Abella & Gorin 99]. However, the paper presents only examples, but does not describe quantitative results. In this thesis, we investigate how the dialogue manager performs in actual field tests. Another difference is that the dialogue manager described in this thesis is entirely based on feature functions derived from speech recognition, language understanding, and task-based knowledge sources. We analyze the system's behavior for choosing the subsequent dialogue action based on a foregoing assessment of the current dialogue state. Especially, we investigate

whether the proposed features are able to determine the best path that leads as quickly as possible to a final state that is likely to meet the user's request. Details can be found in Chapter 7.

- **Error-tolerant dialogue components**
  We propose a strategy for detecting errors that may occur during a dialogue transaction. The strategy employs confidence measures on different levels of the system's architecture. We show that this strategy can be easily integrated into a tree-based spoken dialogue system and evaluate the strategy within a field test. Furthermore, we enrich the maximum entropy-based language understanding component with speech recognition-based confidence measures and analyze if the language understanding component can benefit from the speech recognition-based confidence values in case of recognition errors. Details can be found in Chapter 8.

Chapter 5

Corpus Description and Automatic Speech Recognition Results

In this chapter, the corpora are described and recognition results are presented that serve as a baseline for spoken language understanding results that will be presented in subsequent chapters. Section 5.1 introduces the corpora used. Word graphs and N-best lists are specified in Section 5.2. Section 5.3 briefly reviews the computation of word posterior-based confidence measures that will be incorporated as an additional knowledge source into the understanding process. Speaker adaptation techniques suitable for spoken dialogue systems are outlined in Section 5.4, and results are presented in Section 5.5.

5.1 Corpus Description

All experiments were conducted on two different corpora, the TELDIR corpus and the TABA corpus. Both corpora are German in-house corpora and were collected over several months in telephone-based applications.

5.1.1 TELDIR Corpus

The TELDIR corpus is a German in-house telephone directory assistance corpus. The objective is to answer naturally spoken requests for telephone numbers, fax numbers, and email addresses of persons, companies, and organizations. The TELDIR corpus also covers utterances for spelling units. The data for training the speech recognition system and the natural language understanding component were recorded over several months from fixed telephones as well as wireless and mobile phones under changing conditions. The conditions cover clean speech, office noise, and traffic noise. The corpus allocation is summarized in Table 5.1.

Besides acoustic transcriptions, each utterance of the TELDIR corpus provides a flat concept annotation as meaning representation together with its associated attribute values. A concept is defined as the smallest unit of meaning that is relevant to a specific task [Levin & Pieraccini 95]. Figure 5.1 (a) depicts an example of a concept-based meaning representation for the utterance "Can you give me the email address of Mr. Florian

Table 5.1: Corpus allocation for the TelDir corpus. The TelDir corpus is a German in-house corpus covering the domain of a telephone directory assistance task. Within this task, users can ask for telephone numbers, fax numbers, and email addresses of persons, companies, and organizations.

corpus TelDir-Speech	training	development	evaluation
number of acoustic data [h]	2:00	0:36	0:28
silence portion [%]	58.4	60.7	62.7
# speakers	161	49	41
# sentences	1 399	405	308
# running words	8 840	2 575	1 950
# running phonemes	36 390	10 572	8 040
covered vocabulary	883	443	369
# lexicon entries	1 348		
# pronunciations	1 403		
# unknown words	–	0	0
perplexity (trigram LM)	5.2	22.8	18.7
perplexity (class-trigram LM)	16.4	23.1	22.6

Table 5.2: Bilingual, parallel corpus allocation for the natural language understanding part of the TelDir corpus. The ‘source’ entries describe the natural language source side, the ‘target’ entries mark the formal concept language target side of the corpus.

corpus	training		development		evaluation	
TelDir-NLU	source	target	source	target	source	target
# sentences	1 399		405		308	
# running words	8 840	3 564	2 575	1 054	1 950	801
covered vocabulary	883	22	443	21	369	19
# singletons	0	1	–	–	–	–
# unknown words	–	–	257	0	168	0
perplexity (trigram LM)	5.2	4.1	22.8	4.5	18.7	4.6

Petzhold”. The first line shows the source sentence, the second line depicts the target sentence consisting of concepts marked by the preceding @-symbol. The links describe alignments between words and concepts. Although the corpus already provides a word-concept reference alignment, we keep this information hidden and determine the alignment between words and concepts automatically. The fixed corpus alignment is only used for computing alignment error rates in order to determine whether words are mapped onto the correct concepts. Table 5.2 summarizes the corpus information for the natural language

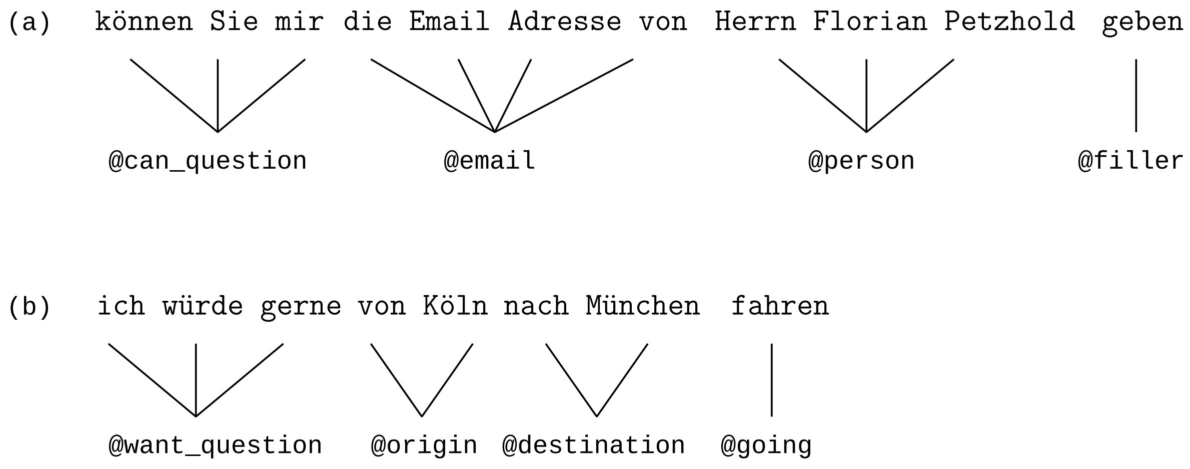

(a) können Sie mir die Email Adresse von Herrn Florian Petzhold geben
@can_question
@email
@person
@filler
(b) ich würde gerne von Köln nach München fahren
@want_question
@origin
@destination
@going

Figure 5.1: Concept-based meaning representations for the utterances (a) "Can you give me the email address of Mr. Florian Petzhold" taken from the TelDir corpus and (b) "I would like to go from Cologne to Munich" taken from the Taba corpus. In each subfigure, the upper text line denotes the word sequence, the lower text line denotes the sequence of concepts. The links describe alignments between words and concepts.

understanding part. The target vocabulary of the TelDir corpus is specified by a total number of 22 concepts, including a filler concept that is used for phrases and sentences that either do not contain useful information or are out of domain sentences. A list of all concepts together with their associated attributes is contained in appendix B.1. In contrast to the speech recognition vocabulary, the vocabulary used in the natural language understanding component is not closed.

5.1.2 Taba Corpus

The Taba corpus is a German in-house corpus covering the domain of a train timetable scheduling task. The speech data was collected from real users interacting with a fully functional automatic spoken dialogue system. The system was introduced in 1994 by Philips Speech Processing, Aachen, Germany, and was the world's first publicly available natural language dialogue system. The system is available under the telephone number +49-241-604020 and provides information on connections between 1,000 German cities. The Taba corpus used in this thesis was kindly provided by Philips Speech Processing and Scansoft. The corpus allocation is summarized in Table 5.3.

Similar to the TelDir corpus, each utterance of the Taba corpus is annotated using a set of flat concepts as meaning representation. Figure 5.1 (b) depicts an example of a concept-based meaning representation for the utterance "I would like to go from Cologne to Munich". Equivalently, the corpus already provides the affiliated alignments between natural language words and formal concepts, which are used for computing alignment

Table 5.3: Corpus allocation for the TABA corpus. TABA is a German in-house corpus covering the domain of a train timetable information system. The corpus was kindly provided by Philips and Scansoft.

corpus TABA-Speech	training	development	evaluation
number of acoustic data [h]	13:09	1:41	1:44
silence portion [%]	46.6	46.7	48.0
# speakers	2 585	324	324
# sentences	17 771	2 247	2 390
# running words	58 098	7 502	7 709
# running phonemes	254 362	33 293	33 659
covered vocabulary	1 678	670	669
# lexicon entries	2 972		
# pronunciations	2 988		
# unknown words	–	0	0
perplexity (trigram LM)	7.7	18.0	16.3
perplexity (class-trigram LM)	11.7	17.9	16.1

Table 5.4: Bilingual, parallel corpus allocation for the natural language understanding part of the TABA corpus. The 'source' entries describe the natural language source side, the 'target' entries mark the formal concept language target side of the corpus.

corpus	training		development		evaluation	
TABA-NLU	source	target	source	target	source	target
# sentences	17 771		2 247		2 390	
# running words	58 098	31 751	7 502	4 083	7 709	4 254
covered vocabulary	1 678	27	670	26	669	26
# singletons	518	0	–	–	–	–
# unknown words	–	–	183	0	169	0
perplexity (trigram LM)	7.7	4.4	18.0	4.8	16.3	4.6

error rates. Otherwise, this information is kept hidden and the alignments are computed automatically. Table 5.4 summarizes the corpus information for the natural language understanding part. A list of all concepts and attributes is contained in appendix B.2. Again, the speech recognition vocabulary is closed whereas the vocabulary used in the natural language understanding component is kept open.

5.2 Word Graph and N-best List Generation

Word graphs are generated during the integrated trigram search. Each node in the graph contains the language model history of the outgoing edges. The search implementation uses so-called reference pointers in order to store traceback information. Thus, it is guaranteed that the graph only contains hypotheses that remain valid at the end of the search when the endpoint detection has determined the end of an utterance. The word graphs are characterized using the following quantities:

- **Word graph density**
  The *word graph density (WGD)* is defined as the total number of edges contained in a word graph, including silence and noise words, divided by the number of spoken words contained in the utterance transcription, where the utterance transcription may contain non-speech annotations, such as hesitations and other phenomena.

- **Node graph density**
  The *node graph density (NGD)* describes the node density of the word graph and is defined as the total number of nodes contained in the word graph, divided by the number of words contained in the utterance.

- **Graph error rate**
  The most important quantity is the *graph word error rate (GER)*. This quantity reflects the word error rate with the smallest Levenshtein distance that is obtained by an oracle-based search on the word graph. The graph word error rate measures to what extend the spoken word sequence is included in the graph. It is also a lower bound to the word error rate obtained in subsequent passes that employ word graphs as intermediate representations.

In order to reduce the graph density, a modified forward-backward pruning is applied. In contrast to the implementation described in [Sixtus & Ortmanns 99] we do *not* transform the word-conditioned word graph into a time-conditioned word graph, but keep the original structure. Doing this requires an additional sorting step of all edges but has a couple of advantages that were not described in [Sixtus & Ortmanns 99]. The advantages only become evident if the original structure is kept. According to [Sixtus & Ortmanns 99], we first compute for each edge the score of the best path passing through this edge. As a modification to the original algorithm, we then sort all edges with respect to their forward-backward path scores in ascending order. This modification was proposed by [Och 07]. Now, the pruning parameter is not a predefined score but the desired number of edges to be kept, that is, the desired word graph density. For a specific word graph, the desired word graph density defines the absolute number of edges to be kept. Everything else is discarded. The modified forward-backward pruning algorithm has the following properties and advantages:

- **Paths are preserved**
  Because each arc in the word graph is assigned the score of the best path passing through this arc, the pruning step preserves paths. This is an important property because it guarantees that each edge in the pruned graph is reachable and that there is at least one path from the source to the sink going through that arc. An often used alternative to forward-backward pruning is the so-called posterior pruning where each arc is pruned if its posterior score falls below a certain threshold. In contrast to forward-backward pruning, posterior pruning does not guarantee that each edge in the pruned graph is reachable. In fact, it is possible that posterior pruning actually prunes edges from the first best recognition result. This usually requires complicated "repair" steps in which unreachable hypotheses must be eliminated and where the first best sentence hypothesis needs to be reinserted into the graph. The forward-backward pruning does not have these disadvantages. For time-conditioned word graphs, this is irrelevant because those graphs have at most one node per timeframe. Thus, unreachable edges do not occur.

- **Computation is more efficient**
  In comparison with time-conditioned word graphs, the forward-backward pruning can be computed much more efficiently on word-conditioned graphs. The reason for this is that only the backward scores need to be computed. The forward scores are already computed during the time-synchronous word-conditioned search. The transformation from word-conditioned to time-conditioned graphs as described in [Sixtus & Ortmanns 99] invalidates the forward scores because new transitions between arcs are introduced. Therefore, the forward scores must be recomputed after the transformation has been applied. This step is not necessary if word-conditioned graphs are used.

- **Language model scores are features of the edges**
  Time-conditioned word graphs cannot store the language model scores as part of the edges because the language model contexts have been discarded. For word-conditioned graphs, language model scores can be directly stored as features of the edges. Thus, the time consuming language model lookups that are necessary for time-conditioned graphs can be omitted.

- **Pruning parameter is the desired Word Graph Density**
  The actual pruning parameter for the modified forward-backward pruning algorithm is the desired graph density. This basically defines an upper bound for the density of word graphs and, thus, is independent of fluctuations of the actual scores across utterances and speaker changes.

The N-best lists are constructed using an A^* search on word graphs. The algorithm starts at the word graph's sink and integrates partial sentence hypotheses into a priority queue. As rest cost estimation, the accumulated forward score of each partial sentence hypothesis starting from the graph's source up to the edge under consideration is used. The implemented algorithm guarantees that the final N-best list only contains sentence hypotheses that are pairwise different and that for two N-best list entries ℓ_i, ℓ_{i+1} there is no sentence hypothesis contained in the graph that

1. either has the same word sequence as ℓ_i but a better score
2. or has a different word sequence and a score that is between the scores of ℓ_i and ℓ_{i+1}.

A plot showing the effect of the length of N-best lists on the N-best error rate is presented for both corpora in Section 5.5.

5.3 Confidence Measures for Speech Recognition

Given a sequence of acoustic vectors x_1^T, statistical automatic speech recognition systems employ Bayes' decision rule in order to determine the most likely sequence of words $\hat{w}_1^N$ with respect to the underlying models:

$$\hat{w}_1^N = \underset{w_1^N}{\operatorname{argmax}} \left\{ pr(w_1^N | x_1^T) \right\} \tag{5.1}$$

$$= \underset{w_1^N}{\operatorname{argmax}} \left\{ \frac{pr(w_1^N) \cdot pr(x_1^T | w_1^N)}{pr(x_1^T)} \right\} \quad \text{with } pr(x_1^T) = \sum_{v_1^M, M} pr(v_1^M) \cdot pr(x_1^T | v_1^M) \ . \tag{5.2}$$

Since $pr(x_1^T)$ does not depend on the word sequence w_1^N, it is neglected for practical reasons because evaluating this expression requires a summation over an exponential growing number of word sequences. As a result, the system's decisions are based on joint probabilities rather than on posterior probabilities [Young 94]. Although it is still possible to determine the most likely word sequence given the language model and the acoustic model, we are unable to measure the *quality* of the recognition. In other words: by giving up the posterior probability, we lose a measure for the certainty of the recognition. If we find a suitable approximation for the denominator expression, we will obtain a score for the certainty of the recognition because the posterior probability can be interpreted as a confidence measure in a natural manner.

The posterior probability in Equation 5.1 can be used as confidence measure for the sentence hypothesis w_1^N. However, we are more interested in word-based confidence measures rather than in sentence-based confidence measures. This can be easily obtained if we sum up all posterior probabilities for a word hypothesis w_n that contain the word under consideration at the corresponding word position:

$$pr(w_n | x_1^T) = \sum_{w_1^{n-1}} \sum_{w_{n+1}^N} pr(w_1^{n-1}, w_n, w_{n+1}^N | x_1^T) \ . \tag{5.3}$$

This quantity can be efficiently approximated on word graphs using a modified forward backward algorithm [Valtchev & Odell$^+$ 97, Wessel & Macherey$^+$ 98]. Since we may also employ other features as confidence measures, we define a confidence measure as a function $\mathcal{C}$ that, given a vocabulary $\mathcal{V}$ and a d-dimensional feature space $\mathcal{F}^d$, determines for each word hypothesis $w_{n,r}$ at position n for the rth utterance whether $w_{n,r}$ has been recognized correctly or wrongly:

$$\begin{aligned}\mathcal{C}: \mathcal{V} \times \mathcal{F}^d &\longrightarrow \{0,1\} \\ \mathcal{C}(w_{n,r}, f^d_{n,r}) &\longmapsto \begin{cases} 1 & \text{if } w_{n,r} \text{ was recognized correctly} \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{5.4}$$

5.3.1 Computation of Word Posterior Probabilities

According to Section 2.3.5, a word graph contains boundary times for its word hypotheses. Let $[w;\tau,t]$ denote a hypothesis for word w with starting time τ and ending time t and let $[w;t]_1^N = [w_1; t_0+1, t_1], \ldots, [w_N; t_{N-1}+1, t_N]$ be a sequence of word hypotheses, where $t_0 = 0$ and $t_N = T$ by definition. Further, define $\tau_n \coloneqq t_{n-1}+1$. Then we can rewrite Equation 5.3 as follows:

$$p([w;\tau,t]|x_1^T) = \sum_{[w;t]_1^N} \sum_{\substack{n=1: \\ [w_n;\tau_n,t_n]=[w;\tau,t]}}^{N} p([w;t]_1^N|x_1^T) \ . \tag{5.5}$$

Instead of explicitly summing the posterior probabilities of all sentence hypotheses that contain a particular word hypothesis, a forward-backward algorithm is used based on dynamic programming. Let h_1^{m-1} be the $m-1$ predecessor words of word w, referred to as the *history* of word w. For ease of notation, the probability $p(w|h_i^j)$ is equated with $p(w)$ if $j < i$, for example, $p(w|h_1^0) = p(w)$.

The forward probability $\Phi(h_2^{m-1}; [w;\tau,t])$ is the probability that the last hypothesis of a word hypothesis sequence is $[w;\tau,t]$ and that its $m-2$ predecessor words are h_2^{m-1}:

$$\Phi(h_2^{m-1}; [w;\tau,t]) = \sum_{\substack{n,\ [w;t]_1^n: \\ [w_n;\tau_n,t_n]=[w;\tau,t]}} \left\{ \delta(w_{n-m+2}^{n-1}, h_2^{m-1}) \cdot \prod_{i=1}^{n} p(x_{t_{i-1}+1}^{t_i}|w_i) \cdot p(w_i|w_{i-m+1}^{i-1}) \right\} . \tag{5.6}$$

This quantity can be efficiently computed using the following recursive formula:

$$\Phi(h_2^{m-1}; [w;\tau,t]) = p(x_\tau^t|w) \cdot \sum_{h_1} \sum_{\tau'} \Phi_{\tau-1}(h_1^{m-2}; [h_{m-1}; \tau', \tau-1]) \cdot p(w|h_1^{m-1}) \ . \tag{5.7}$$

Similarly, the backward probability $\Psi([w;\tau,t]; f_1^{m-2})$ is the probability that the first hypothesis of a word hypothesis sequence is $[w;\tau,t]$ and that its future is f_1^{m-2}:

$$\Psi([w;\tau,t];f_1^{m-2}) = \sum_{\substack{N,\ [w;t]_n^N:\\ [w_n;\tau_n,t_n]=[w;\tau,t]}} \left\{ \delta(w_{n+1}^{n+m-2}, f_1^{m-2}) \cdot \prod_{i=n}^{N} p(x_{\tau_i}^{t_i}|w_i) \cdot \right. \\ \left. \cdot \prod_{j=n+\max\{1,m-1\}}^{N} p(w_j|w_{j-m+1}^{j-1}) \right\} . \tag{5.8}$$

This quantity can also be efficiently computed using the following recursive formula:

$$\Psi([w;\tau,t];f_1^{m-2}) = p(x_\tau^t|w) \cdot \sum_{f_{m-1}} \sum_{t'} \Psi_{t+1}([f_1;t+1,t'], f_2^{m-1}) \cdot p(f_{m-1}|w, f_1^{m-2}) . \tag{5.9}$$

With the definitions in Equations 5.5, 5.7, and 5.9 the posterior probability can now be computed by summing over all histories and futures of the word hypothesis $[w;\tau,t]$ and by incorporating the language model probabilities. Thus, we arrive at the following equation:

$$p([w;\tau,t]|x_1^T) = \sum_{h_2^{m-1}} \sum_{f_1^{m-2}} \left\{ \frac{\Phi(h_2^{m-1};[w;\tau,t]) \cdot \Psi([w;\tau,t];f_1^{m-2})}{p(x_1^T) \cdot p(x_\tau^t|w)} \cdot \right. \\ \left. \cdot \prod_{n=1}^{m-2} p(f_n|h_{n+1}^{m-1}, w, f_1^{n-1}) \right\} . \tag{5.10}$$

5.3.2 Computation of Word-Based Confidence Measures

A word confidence $\widetilde{p}(\cdot)$ is now computed by summing over all word hypothesis probabilities $p([w;\tau,t]|x_1^T)$ that share the same word label and that have at least one common time frame t:

$$\mathcal{C}_{\text{ASR}} := \widetilde{p}([w;\tau,t]|x_1^T) = \max_{\tilde{t}:\tau\leqslant\tilde{t}\leqslant t} \sum_{(\tau',t'):\tau'\leqslant\tilde{t}\leqslant t'} p([w;\tau',t']|x_1^T) . \tag{5.11}$$

The basic principle is depicted in Figure 5.2. The maximization over $\tilde{t}$ is motivated by the following property: for each time frame $\tilde{t}$, all word hypothesis probabilities that are contained in the word graph and that intersect this time frame must sum to unity. For a word hypothesis $[w;\tau,t]$ of the recognized word sequence, the word graph may contain several word hypotheses sharing the same word label but having slightly different time boundaries. Thus, the probability mass is distributed over several edges. Although word hypotheses sharing the same word label may indicate the correctness of the word hypothesis under consideration, they belong to different paths and compete with each other. To mitigate this effect, we maximize over the time frame $\tilde{t}$, for which the sum of the word hypothesis probabilities with the same word label and approximately the same time boundaries is maximal, and use this value as word confidence.

5.3.3 Evaluating Confidence Measures

After computing the confidence score, each generated word hypothesis is tagged as either *correct* or *wrong*, depending on whether its confidence exceeds a tagging threshold that must be optimized on a development set beforehand. The performance of the ASR confidence measure is evaluated using two different metrics:

- **Confidence Error Rate**
  The *confidence error rate (CER)* is defined as the number of incorrectly assigned tags divided by the total number of generated words in the recognized sentence. The baseline CER is given by the number of substitutions and insertions, divided by the number of generated words. The CER strongly depends on the tagging threshold. Therefore, the tagging threshold is adjusted beforehand on a development corpus *distinct* from the test set.

- **Detection Error Tradeoff curve**
  The *detection error tradeoff (DET)* curve plots the *false rejection rate* versus the *false acceptance rate* for different values of the tagging threshold. The false rejection rate is defined as the number of correctly recognized words that have been tagged as wrong, divided by the total number of correctly recognized words. It is also referred to as *type I error*. The false acceptance rate, also referred to as *type II error*, is calculated as the number of incorrect words that have been accepted, divided by the total number of incorrectly recognized words.

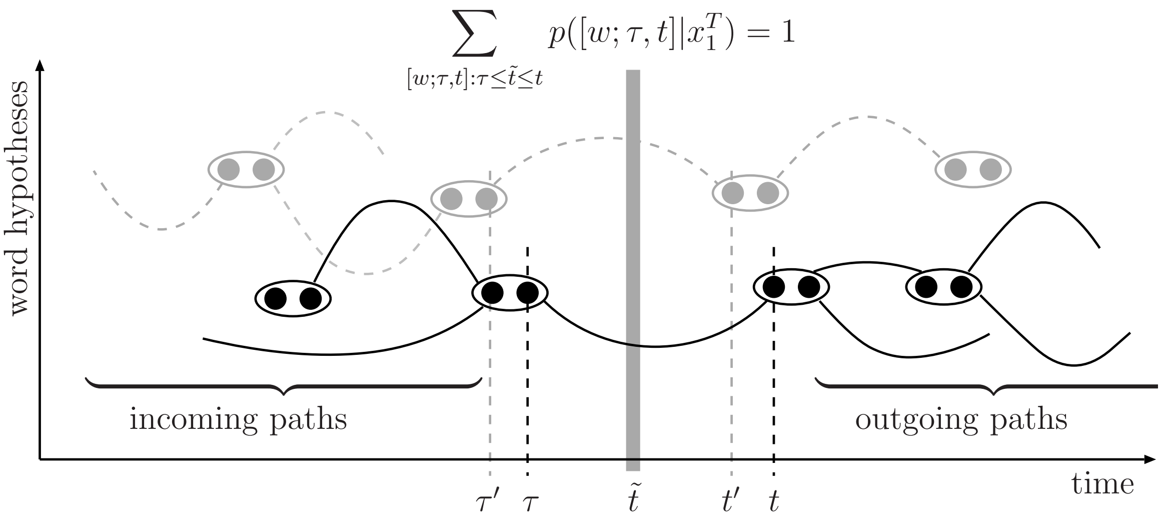


Figure 5.2: Principle of the word graph-based confidence computation. The word posterior probabilities are computed within the framework of a forward backward algorithm with the property that for each time frame $\tilde{t}$, all word hypothesis probabilities that intersect this time frame must sum to one.

5.3.4 Modifications to former Implementation

Although the computation of confidence measures on the basis of word graphs was already introduced in [Wessel & Macherey^{+} 98], the implementation used in this thesis differs significantly from the former implementation described in [Macherey 98] and [Wessel 02]. Furthermore, the modifications used are more suitable for a realtime speech recognition system. The differences are:

- **Word-conditioned word graphs**
  The automatic speech recognition module performs an integrated word-conditioned m-gram search where m depends on the complexity of the language model used. This word dependency is carried over into the structure of the word graph. Each node of a word graph is associated with a time frame and a language model history. Thus, each time frame may contain multiple nodes that pairwise differ in their language model histories. In contrast to this, the former implementation condensed multiple nodes for one time frame into a single node leading to a time-conditioned interpretation of the word graph. Thus, the final word graph allowed for more word contexts than actually survived the search process. The new implementation only contains those contexts that were originally hypothesized and survived the pruning during the search process.

- **Exact representation of the pruned search space**
  Caused by the word-conditioned graph structure, the new implementation guarantees that a word graph is an exact representation of all word hypotheses that have survived the pruning of the search space.

- **Inherent elimination of unreachable hypotheses**
  The book keeping of word hypotheses is based on a technique called reference counting, that is, edges, from which the final node cannot be reached, are eliminated automatically during the search process. Thus, the time consuming removal of unreachable hypotheses in a postprocessing step, which was necessary in the former implementation, is omitted.

- **Generic implementation**
  The graph structure is implemented using a graph container class. Because of the language model histories stored in a graph's nodes, there is only one generic implementation necessary that covers all language models with arbitrary history lengths.

5.4 Speaker Adaptation

In the following section, we will briefly summarize the fundamentals of feature-space maximum likelihood linear regression and discuss how this adaptation technique can be efficiently employed in a realtime speech recognition system.

5.4.1 F-MLLR

Denote η a hidden Markov Model. Then the goal of the *feature-space maximum likelihood linear regression (F-MLLR)* is to affinely transform the features of the test data such as to maximize the likelihood of the hidden Markov Model η:

$$\hat{x}_t = Ax_t + b, \quad A \in \mathbb{R}^{D\times D}, b \in \mathbb{R}^D \tag{5.12}$$

$$(\hat{A}, \hat{b}) = \underset{(A,b)}{\operatorname{argmax}} \left\{ \log pr(\hat{X}|\eta) \right\}, \quad \hat{X} = \hat{x}_1, \ldots, \hat{x}_T \ . \tag{5.13}$$

Integrating the affinely transformed acoustic features into the auxiliary function of the EM algorithm leads to:

$$Q(A,b) = \frac{1}{pr(X|\eta)} \cdot \sum_Q p(X,Q|\eta) \log p(\hat{X},Q|\eta) \tag{5.14}$$

$$= \sum_{i=1}^{D} \sum_{t=1}^{T} \gamma_t(i) \left[\log |A| - \frac{1}{2} \operatorname{tr} \left\{ \Sigma_i^{-1} (\tilde{A}\tilde{x}_t - \mu_i)(\tilde{A}\tilde{x}_t - \mu_i)^\top \right\} \right] , \tag{5.15}$$

where $\tilde{A} = [A|b]$ and $\tilde{x}_t = \begin{bmatrix} x_t \\ 1 \end{bmatrix}$. Maximizing Q with respect to the affine transformation $\tilde{A}$ leads to the following sufficient statistics for the F-MLLR:

$$T = \sum_{i=1}^{D} \sum_{t=1}^{T} \gamma_t(i) \tag{5.16}$$

$$K = \sum_{i=1}^{D} \sum_{t=1}^{T} \gamma_t(i) \Sigma_i^{-1} \mu_i \tilde{x}_t^\top \tag{5.17}$$

$$G^{(j)} = \sum_{i=1}^{D} \sum_{t=1}^{T} \frac{\gamma_t(i)}{\sigma_{ij}^2} \tilde{x}_t \tilde{x}_t^\top \ . \tag{5.18}$$

For each row j, we have to solve

$$T([A^{-\top}|0])_j = \tilde{a}_j G^{(j)} - k_j \ , \tag{5.19}$$

which is a $(D+1)$ dimensional vector equation. This leads to the following algorithm:

Algorithm 1 F-MLLR Calculation

```
Input: posteriors $\gamma_t(i)$
       features $x_t$
       Gaussians $\{\mu_i, \Sigma_i\}$
Output: $\tilde{A} = [A|b]$
  Compute sufficient statistics $T$, $K$, and $G^{(i)}$
  Invert $G^{(i)}$ statistics for each dimension
  Initialize $\tilde{A} = [I_n|0]$
  repeat
    Compute $A^{-\top}$ and $A^{*\top}$
    for all dimensions $d$ do
      solve Equation 5.19
    end for
  until convergence
```

where A^* is the adjoint matrix of A.

5.4.2 F-MLLR in Realtime Speech Recognition Systems

Since Equations 5.16 - 5.18 depend on the state posterior probabilities $\gamma_t(i)$, applying the F-MLLR actually requires a two-pass recognition, that is, in the first pass the most likely word sequence together with its alignment is determined. Using this data as input for the F-MLLR algorithm, we can compute the affine transformation. After applying the affine transformation matrix $\hat{A}$ onto the feature vectors x_1^T resulting in the transformed feature vectors $\hat{x}_1^T$, we have to do a search in the transformed feature space in a second pass. Of course, performing the time consuming acoustic search two times in a spoken dialogue system that is bound to certain realtime constraints is not feasible. Therefore, the following approximations can be applied.

Incremental F-MLLR

In general, a dialogue transaction of a turn-based dialogue system contains more than one user utterance. Under the assumption that the speaker does not change during a dialogue transaction we can apply an incremental F-MLLR as follows: after recognizing the user's first utterance, we compute the transformation matrix according to Algorithm 1. However, instead of applying it on the first utterance, we shall use this transformation matrix in order to transform the feature vectors obtained from the user's second utterance. The adaptation data collected from the first and second utterance is used to estimate a transformation matrix for the third utterance, and so on. This incremental update of the F-MLLR transformation matrix and its delayed application has the advantage that there are no further computation costs involved because the otherwise expensive second recognition is avoided. However, since the quality of the F-MLLR depends on the number of adaption data available for estimating the transformation matrix, this approximation can lead to a suboptimal solution.

Restriction on Word Graphs

A refinement to the incremental F-MLLR is the F-MLLR restricted on word graphs. Instead of performing a full acoustic search on the transformed feature vectors, the search can be restricted to the hypotheses stored in a word graph that was obtained during the first recognition pass. If the time boundaries of the word hypotheses are also kept, the acoustic search is equivalent to an acoustic rescoring on word graphs. This significantly reduces the time needed for the second recognition pass. Doing this, we can apply the F-MLLR starting with the first user utterance. Note that we obtain the incremental F-MLLR algorithm if we do not perform an acoustic rescoring.

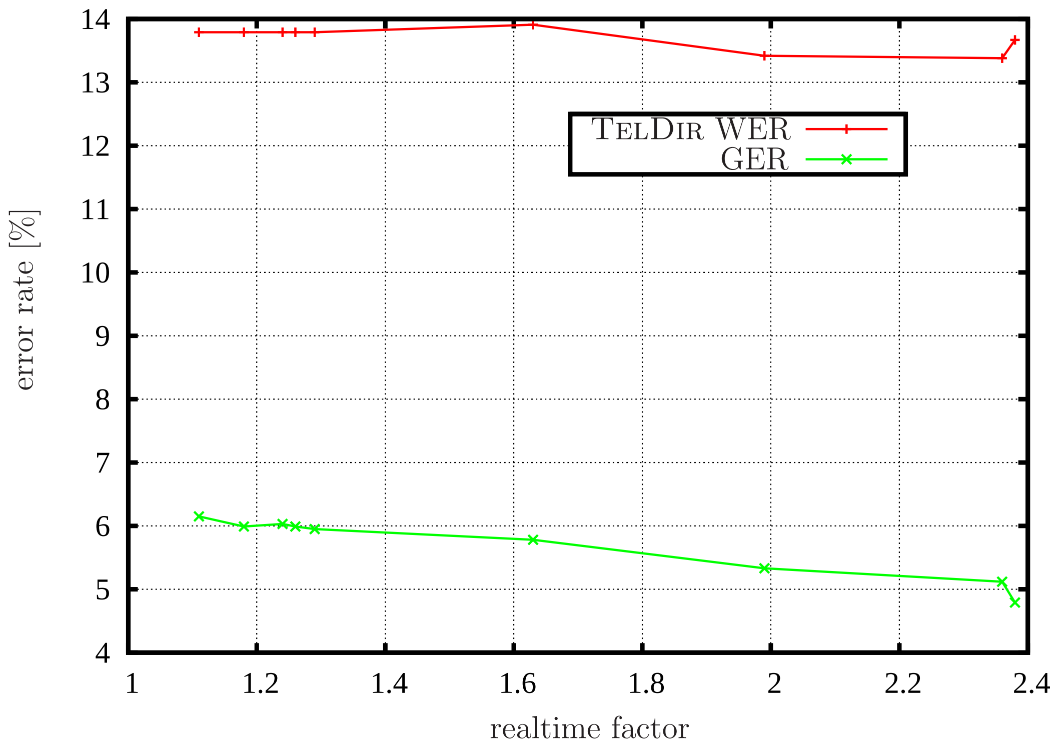
5.5 Results

For both the TelDir and the Taba task, the automatic speech recognition system uses a time-synchronous Viterbi beam search algorithm based on the concept of word-conditioned tree copies [Ney & Welling+ 98]. The acoustic model is based on maximum likelihood-trained Gaussian mixture models. We use 6 state hidden Markov models (HMM) in Bakis topology modeling triphones where two contiguous states are identical. With a frame shift of 10ms, the average length of such a triphone is 60ms, which approximately corresponds to the average length of a phoneme. A single globally pooled diagonal covariance matrix is employed.

For the language model a class-based trigram model was integrated into the Viterbi beam search. The estimation and smoothing method applied for training the language model is absolute discounting with leaving one out. A summary of the models used is given in Table 5.5.

Table 5.5: Summary of the models used for the TelDir and the Taba speech recognition system. For an acoustic observation vector (feature vector), “cmp” refers to the number of mel frequency cepstral coefficients (MFCC), “1st deriv” refers to the first derivativ of these coefficients, and “2nd deriv” refers to the second derivativ of the first MFCC component, which corresponds to the energy of the signal.

	TelDir corpus	Taba Corpus
dimension of feature vector (cmp + 1st deriv + 2nd deriv)	$12 + 12 + 1 = 25$	$12 + 12 + 1 = 25$
frame shift [ms]	10	10
LDA window, dimension	± 1 frame, 75×33	± 1 frame, 75×33
CART leafs + silence state	$400 + 1$	$800 + 1$
HMM	6 state Bakis, within-word triphone models	6 state Bakis, within-word triphone models
Gaussian mixture models	gender independent 19 779 densities 1 pooled variance	gender independent 89 490 densities 1 pooled variance
Search	Viterbi beam search integrated class trigram	Viterbi beam search integrated class trigram
Language model	class-based trigram absolute discounting with leaving one out	class-based trigram absolute discounting with leaving one out



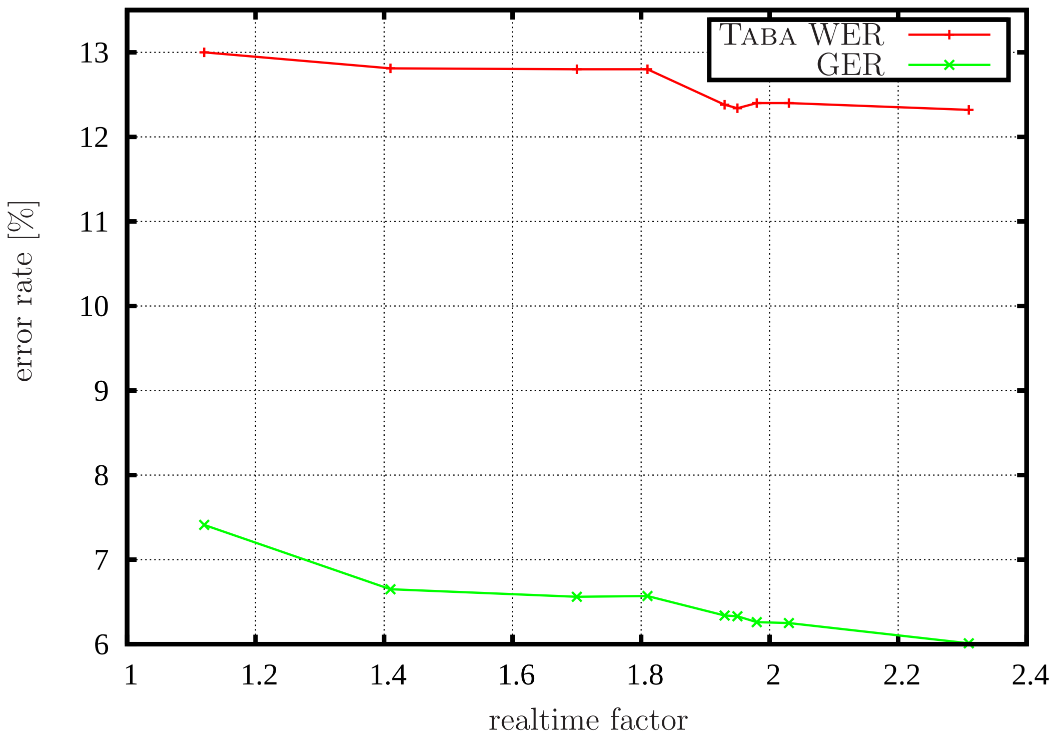


Figure 5.3: Word and graph error rates in course of realtime factors for the TelDir and Taba development corpus. The measurement points for both tasks where obtained using a Pentium IV processor with a clock rate of 2.8GHz and a first level cache size of 512KB.

5.5.1 Realtime Speech Recognition

Because a spoken dialogue system must satisfy certain realtime constraints, we have to adjust various search and pruning parameters for the automatic speech recognition module, which may result in additional search errors.

To document the performance loss, Figure 5.3 depicts word error rates as well as graph error rates in course of the realtime delay. The values for both tasks were obtained using a Pentium IV, 2.8GHz processor with a first level cache of 512kByte. As shown in the graphs, the performance loss for the TelDir task is around 0.4% in terms of word error rate and 0.9% in terms of graph error rate if the realtime factor is in the range 1.1 - 2.0. For the Taba task, the performance falls by 0.6% in terms of word error rate and 1.1% in terms of graph error rate, respectively. We have adjusted the search and pruning parameters such that a realtime factor of 1.1 is obtained on both tasks where the additional 0.1 delay is caused by a mean normalization step that uses an asymmetric window that is shifted over the incoming audio data. The curves shown in Figure 5.3 already include the generation of word graphs and the computation of confidence measures. Table 5.6 and 5.8 document recognition results under these settings.

5.5.2 Realtime Speaker Adaptation

The rows in Table 5.6 and Table 5.8 describe different strategies for collecting adaptation data during a dialogue transaction. The "incremental F-MLLR" incrementally adds adaptation data from a user's utterances and applies the estimated transformation matrix on subsequent utterances. The "incremental F-MLLR on word graphs" applies the affine transformation within an acoustic rescoring step starting with the user's first utterance. The last row lists results for the 2-pass F-MLLR where we use all utterances for adaptation that were input by a user during a dialogue transaction. This strategy is of course not feasible for a spoken dialogue system and is listed here as a contrastive result. Because an F-MLLR transformation is an unsupervised adaptation technique, it is not guaranteed that the 2-pass F-MLLR described in the previous section always outperforms the iterative variant.

An important point when applying an incremental F-MLLR is to use confidence measures in order to decide which user utterances shall be used for adaptation, once the recognition step has finished. The motivation for this is that if the recognition result for the first utterance already contains recognition errors, the transformed observation vectors may trigger further errors in subsequent user utterances because they were adjusted to wrong transcriptions during the estimation process. This effect is usually mitigated in a two-pass recognition approach where all user utterances are used for adaptation because on average, the correctly recognized utterances outweigh the wrongly recognized utterances, given that the baseline word error rate is $\ll 50\%$. In an incremental approach, the transformation matrix is affected by the previous estimation. Therefore, erroneous recognitions in early utterances can have a much higher impact on the overall performance

Table 5.6: Recognition and confidence measure results for the TelDir corpus. The F-MLLR rows describe results using different strategies for collecting adaptation data during a dialogue transaction.

corpus TelDir-Speech development	SER [%]	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline	38.5	13.7	6.2	11.9	10.3
with incr. F-MLLR	37.3	12.4	6.2	10.9	8.9
with incr. F-MLLR on WG	35.6	12.3	6.0	10.8	10.3
with 2-pass F-MLLR	34.3	12.2	5.9	10.6	8.5

corpus TelDir-Speech evaluation	SER [%]	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline	45.1	14.9	8.3	12.5	10.4
with incr. F-MLLR	42.5	13.6	7.5	11.2	9.4
with incr. F-MLLR on WG	41.2	13.5	7.2	11.1	9.5
with 2-pass F-MLLR	41.2	13.7	7.4	10.9	8.3

and sometimes may even underperform the baseline result. For the TelDir task, a confidence value of 0.6 was chosen, that is, whenever a word hypothesis of the first best sentence hypothesis falls below this threshold the word is discarded and not used for adaptation. This parameter is corpus and performance-dependent and was optimized on the development set beforehand. For the Taba task, using confidence measures in adaptation did not yield better results. Thus, the threshold was set to 0.0.

In order to reduce the computation time, the incremental F-MLLR in combination with acoustic rescoring on word graphs uses a cache for edges that share the same word index and the same boundary times but occur in different language model contexts. However, since this F-MLLR approach gave only a slight improvement in terms of word error rate compared to the incremental F-MLLR, we chose the incremental F-MLLR without the additional rescoring step and used this as baseline result for the following experiments.

5.5.3 Word Graphs and N-best Lists

Word graphs are constructed during beam search. The word graphs produced are an exact representation of all hypotheses that remain valid at the end of the search process. An edge in a word graph is called valid, if there is a path from the source to the sink containing this edge. Table 5.7 and 5.9 present word graph statistics for the TelDir and the Taba corpus.

After computing the confidence scores, a forward-backward pruning on the word graphs is applied. This pruning step significantly reduces the word graph density. The reduced word graphs are used in order to extract N-best lists using an A^* search. Figure 5.4

Table 5.7: Word graph and N-best list statistics for the TelDir corpus. The table contains statistics before and after applying a forward-backward pruning.

corpus	TelDir-Speech development	WER [%]	GER [%]	WGD	NGD	100-best ER [%]
	baseline	12.4	6.2	1218	262	6.2
	with F/B pruning	12.4	6.2	301	101	6.3

corpus	TelDir-Speech evaluation	WER [%]	GER [%]	WGD	NGD	100-best ER [%]
	baseline	13.6	7.4	1187	256	7.7
	with F/B pruning	13.6	7.5	300	100	7.8

Table 5.8: Recognition and confidence measure results for the Taba corpus. The F-MLLR rows describe results using different strategies for collecting adaptation data during a dialogue transaction.

corpus Taba-Speech development	SER [%]	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline	20.7	13.0	7.4	11.2	9.2
with incr. F-MLLR	20.3	12.3	7.1	10.6	8.9
with incr. F-MLLR on WG	20.3	12.1	7.1	10.5	9.3
with 2-pass F-MLLR	19.4	11.8	7.0	10.1	8.5

corpus Taba-Speech evaluation	SER [%]	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline	19.2	12.8	7.6	11.0	9.2
with incr. F-MLLR	18.6	12.5	7.4	10.7	9.0
with incr. F-MLLR on WG	18.6	12.4	7.5	10.6	9.1
with 2-pass F-MLLR	17.7	12.1	7.4	10.3	8.5

shows the effect of the average N-best list length on the N-best error rate. Note that by using a 1000-best list we can get very close to the graph error rate. This effect is not caused by search errors. The plots in Figure 5.3 describe the search errors we obtain by pruning more hypotheses due to realtime constraints. For both tasks, this is less than 1% in terms of word error rate. However, the large difference between the first-best sentence hypothesis and the oracle sentence hypothesis determined on the word graph indicates that our baseline models are still to weak in order to find the correct word sequence. Therefore, we will use additional knowledge sources in the next chapter and analyze their influence on the word error rate.

Table 5.9: Word graph and N-best list statistics for the Taba corpus. The table contains statistics before and after applying a forward-backward pruning.

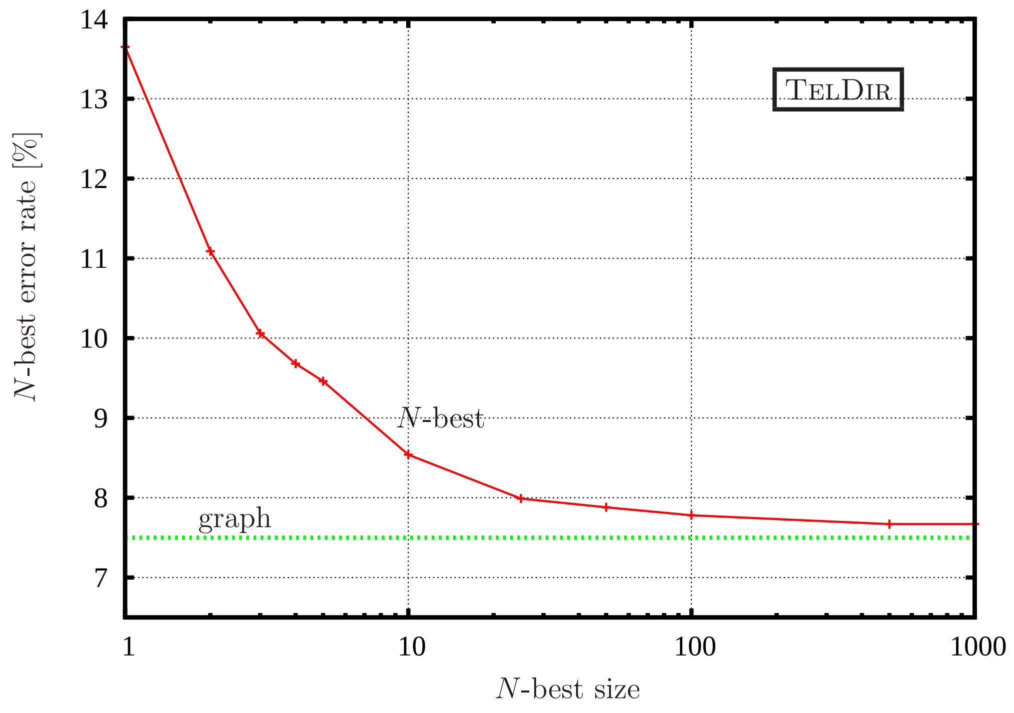
corpus	Taba-Speech development	WER [%]	GER [%]	WGD	NGD	100-best ER [%]
	baseline	12.3	7.1	723	269	7.3
	with F/B pruning	12.3	7.2	131	40	7.4

corpus	Taba-Speech evaluation	WER [%]	GER [%]	WGD	NGD	100-best ER [%]
	baseline	12.5	7.4	730	276	7.5
	with F/B pruning	12.5	7.4	131	41	7.5

The exceptionally high word and node graph densities for the TelDir corpus are mainly caused by two reasons. First, the speech recognizer performs a word-conditioned tree search, that is, many equally labeled edges occur multiple times within different language model contexts. Second, the speech recognition vocabulary also contains entries for non-speech effects like hesitations and noise. In contrast to the silence word, these words are not penalized by forward transitions. Thus, the search space is parqueted by small short non-speech words. After applying forward-backward pruning, most of these edges are eliminated. In contrast to the TelDir pronunciation lexicon, the Taba lexicon provides less non-speech models. Therefore, the initial edge and node densities are much smaller. With the forward backward pruning the word and node graph densities significantly decrease whereas the graph and N-best error rates deteriorate only slightly.

5.5.4 Confidence Measures

According to Section 5.3, the definition of confidence measures for speech recognition is based on posterior probabilities. The posterior-based confidence measures are computed before the forward-backward pruning is applied. As shown in Table 5.6 and 5.8, posterior-based confidence measures perform very well, even after applying an F-MLLR. Rescaling acoustic and language model scores is vital when computing posterior-based confidence measures. For both tasks, the language model score was set to 1.0. The acoustic scaling factor was set to $\frac{1}{15}$ for the TelDir task and to $\frac{1}{9}$ for the Taba task, respectively. Both values have been optimized on the development sets beforehand. In order to classify a word hypothesis to be correct or wrong, a task-specific threshold was used that was optimized on the development set as well. The optimal thresholds for the evaluation sets differ only slightly compared to the optimal thresholds adjusted on the development sets.



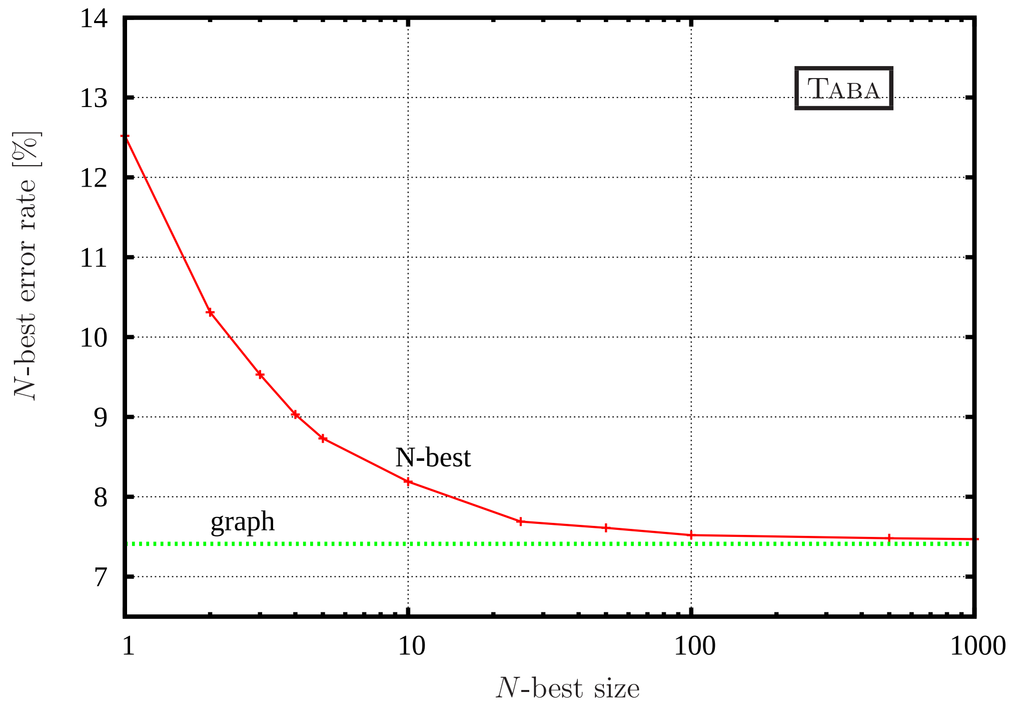


Figure 5.4: Effect of the N-best list size on the N-best error rate for the TelDir and Taba evaluation corpus. The N-best lists were extracted from the word graphs described in Tables 5.7 and 5.9. The horizontal dashed lines denote the lower boundaries defined by the corresponding graph error rates.

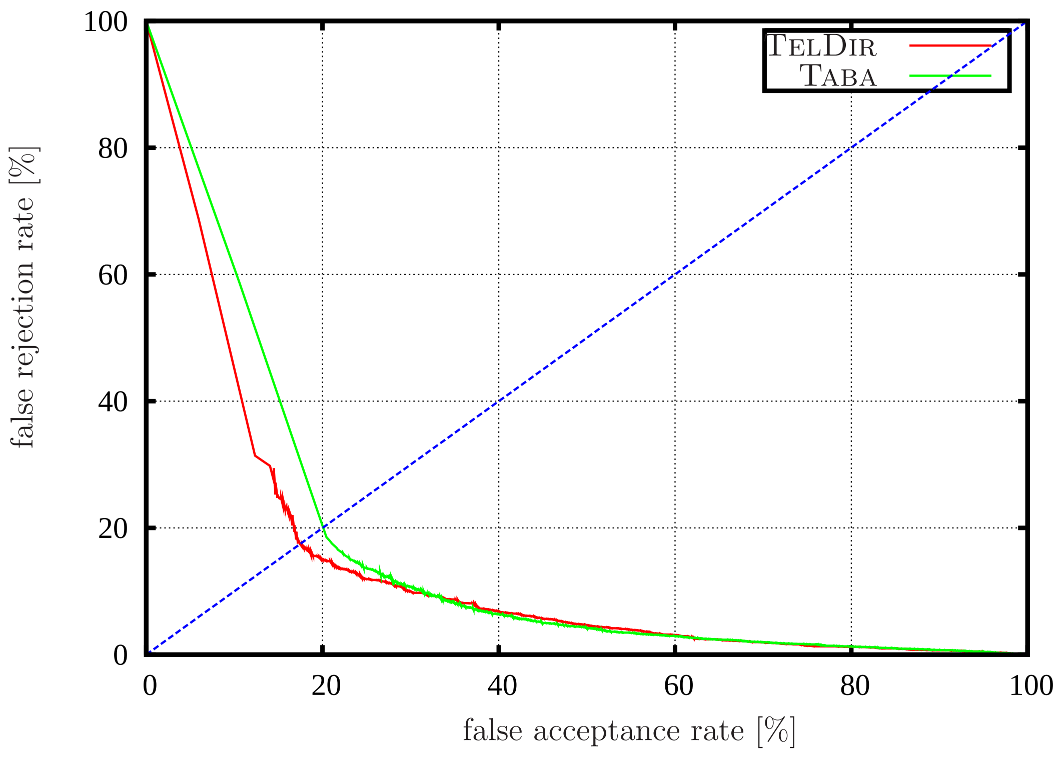


Figure 5.5: Detection error tradeoff curve for the full TelDir and Taba test corpus. The points of intersection with the diagonal line define the equal error rate, that is, the point where both error types are equal.

5.6 Summary

In this chapter, we have defined the corpora used and described an algorithm for confidence measures based on word-conditioned word graphs. The word graphs were pruned using a modified forward-backward pruning algorithm. The modified forward-backward algorithm has several useful properties that make it well suited for realtime recognition tasks. Furthermore, we presented and analyzed different strategies for F-MLLR adaptation that are suitable for realtime speech recognition systems. We compared their performance on the Taba and the TelDir task and provided recognition results together wtih realtime analyses. We reported word graph and N-best list results that we shall use as baseline results for some of the investigations presented in the next chapters.

Chapter 6

Natural Language Understanding

In this chapter, we compare two approaches to natural language understanding that are based on statistical machine translation.[1] The first approach employs the source-channel paradigm whereas the other uses the maximum entropy framework. Starting with an annotated corpus, we describe the problem of natural language understanding as a translation from a source sentence to a formal language target sentence. We analyze the quality of different alignment models and feature functions and show that the direct maximum entropy approach outperforms the source channel-based method. Furthermore, we investigate how both methods perform if the input sentences contain speech recognition errors. Finally, we investigate a new approach for combining automatic speech recognition with natural language understanding using the minimum error criterion.

6.1 A General View of Natural Language Understanding

The objective of natural language understanding (NLU) is to extract all the information from a natural language-based input that is relevant to a specific task. Often, stochastic grammars are used for this task requiring handcrafted rules, that is, dependencies between words and concepts must be modeled explicitly. Although partial parsing techniques are well suited to parse ungrammatical sentences, the sole usage of rule-based methods can turn out to be inflexible. Especially, when extending the application scenario or the application domain itself, many rules must be rewritten by hand to adjust them to new phrases and keywords, thus, leading to the problem of reusability. Therefore, the question is how this process can be simplified and whether the complex dependencies between words and concepts can be learned automatically by just providing pairs of sentences and concept strings.

To achieve this goal, we will take a closer look at the natural language understanding problem. Basically, natural language understanding can be viewed as a translation from a source language to a formal target language, that is, given a natural language source sentence $w_1^N = w_1, \ldots, w_N$ we choose the formal language target sentence $\hat{c}_1^M = \hat{c}_1, \ldots, \hat{c}_M$ among the space of all possible translations that has the highest probability, leading to

[1] A part of the work presented here was done in cooperation with Franz Josef Och and Oliver Bender.

the following decision rule:

$$\begin{aligned}
\hat{c}_1^M &= \operatorname*{argmax}_{c_1^M} \left\{ pr(c_1^M|w_1^N) \right\} && (6.1)\\
&= \operatorname*{argmax}_{c_1^M} \left\{ \frac{pr(c_1^M) \cdot pr(w_1^N|c_1^M)}{pr(w_1^N)} \right\} \\
&= \operatorname*{argmax}_{c_1^M} \left\{ pr(c_1^M) \cdot pr(w_1^N|c_1^M) \right\} \ . && (6.2)
\end{aligned}$$

Here, the argmax operation denotes the search problem, that is, the generation of the most likely sequence of formal semantic concepts in the target language. Using Bayes' theorem, Equation 6.1 can be rewritten to Equation 6.2, where the denominator can be neglected. Equations 6.1 and 6.2 induce two different approaches: either we can directly optimize the posterior probability $pr(c_1^M|w_1^N)$ using, for example, the maximum entropy framework or we can employ the classical source-channel approach using Bayes' theorem and decompose the posterior probability into two probability distributions: the translation probability $pr(w_1^N|c_1^M)$ and the concept language probability $pr(c_1^M)$. Once we have substituted these probability distributions with suitable model distributions, we can learn the model parameters with supervised learning techniques. Before we describe the methods in more detail we will first discuss the concept language used and review some basic ideas from machine translation, such as alignment models, that we employ for both approaches in order to automatically align words to concepts.

6.2 Concept Language

A concept c is defined as the smallest unit of meaning that is relevant to a specific task [Levin & Pieraccini 95]. Concept representations are sometimes used in a hierarchical manner [Miller & Bobrow$^+$ 94b], that is, concepts can be nested and subsume other concepts. Here, we use non-hierarchical concepts, which we call *flat* concepts. By using flat concepts as formal target language, the annotation of the input sentences can easily be provided because for a human annotator no structural knowledge about concepts and their interplay is necessary. Additionally, only a small set of sentence pairs is needed for training in order to obtain useful results. Although not a strict requirement, we assume that the concepts occur in the same order as given by the source text. An example is provided in Figure 6.1. The links between the sentences indicate to which concepts the words are aligned. Note that these alignments between words and concepts are not given explicitly but must be determined automatically from the training data, which is provided as pairs of source sentences and concept strings. Each concept is associated with a (possibly empty) set of attributes and each attribute is associated with a more or less complex rule describing how to extract the value from the words aligned to the concept.

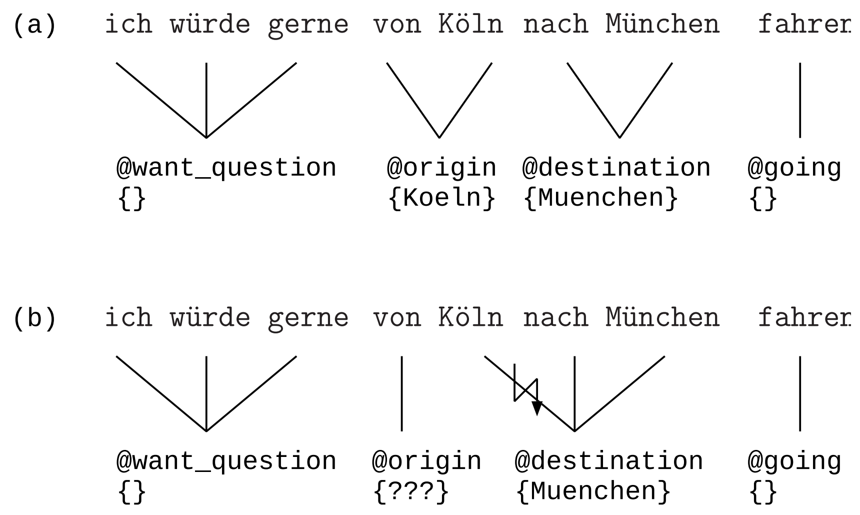


Figure 6.1: Example of a correctly and wrongly aligned word sequence for the utterance "I would like to go from Cologne to Munich". For each concept, its attribute values are enclosed in braces. In (a), the phrase 'from Cologne' is correctly aligned to the concept `@origin`. In (b), the city name 'Cologne' is wrongly aligned to the concept `@destination`. If each concept together with its associated words is considered separately, it will be impossible to extract the attribute value for the concept `@origin` in case (b).

6.3 Alignment Models and Local Context Dependencies

As depicted in Figure 6.1, the meaning of a word and thus its alignment to the correct concept depends on its context. For example, the allocation of the city name *Köln (Cologne)* to the concept `@origin` is only apparent when looking at the preceding preposition *von (from)* in the source sentence. Although the sequence of concepts is identical in both cases, only case (a) allows for a correct extraction of the attribute values. If each concept together with its aligned words is considered independently from all the other concepts, it will be impossible to extract the correct attribute value from the concept `@origin` in case (b). Therefore, finding correct alignments automatically is essential for both approaches. In the following, we will briefly review different alignment models that are used in machine translation and that we will use for either approach. A consequence of words and concepts having the same order is that the generated alignments will be monotonic. Also, we enforce that every word is aligned to exactly one concept. This might be problematic if the source language contains compounds that could be aligned to more than one concept. For example, the German compound "Privatadresse (home address)" as opposed to employer's address could be aligned to two target concepts `@private` and `@address` to distinguish it from "Büroadresse (office address)", which could be mapped to `@office` and `@address`. To enforce the one-word-to-one-concept constraint, we can design the target language appropriately and shift the distinction between home and office address into the attributes of the `@address` concept. Words not contributing to the meaning of a sentence are aligned to filler concepts.

For the introduction of the alignment models, we switch our notation and identify a word sequence w_1^N with f_1^J and a sequence of concepts c_1^M with e_1^I in order to use the same notation that is common in machine translation. If we rewrite the translation probability $pr(f_1^J|e_1^I)$ by introducing a hidden alignment $a_1^J = a_1, \dots, a_j, \dots, a_J$, with $a_j \in \{1, \dots, I\}$, we obtain the following identity:

$$pr(f_1^J|e_1^I) = \sum_{a_1^J} pr(f_1^J, a_1^J|e_1^I) \tag{6.3}$$

$$= pr(J|e_1^I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{6.4}$$

$$= \underbrace{pr(J|e_1^I)}_{\text{length probability}} \cdot \sum_{a_1^J} \prod_{j=1}^{J} \underbrace{pr(a_j|a_1^{j-1}, f_1^{j-1}, e_1^I)}_{\text{alignment probability}} \cdot \underbrace{pr(f_j|f_1^{j-1}, a_1^j, e_1^I)}_{\text{lexicon probability}} . \tag{6.5}$$

The different translation models result from different decompositions of Equation 6.5. In the following, we will briefly introduce these translation models, which were proposed in [Brown & Della Pietra+ 93] and [Vogel & Ney+ 96], and are the basis for extracting word-concept phrase pairs.

Model 1

By replacing the dependency in Equation 6.5 from a_1^{j-1} to j, we obtain a zero-order hidden Markov model. Assuming a uniform alignment probability $p(i|j, I) = \frac{1}{I}$, we obtain the much simpler model 1, which we use as starting point:

$$pr(f_1^J|e_1^I) = \frac{p(J|I)}{I^J} \cdot \sum_{a_1^J} \prod_{j=1}^{J} \left[p(f_j|e_{a_j}) \right] . \tag{6.6}$$

Due to the uniform alignment probability, model 1 cannot take into account where words appear in both strings. This very simple model has the important property that its training criterion is a convex function, which has only one maximum. Therefore, we can find the global maximum from any non-zero starting point.

HMM Alignment

If we assume a first-order dependence for the alignment a_j in Equation 6.5, restrict the dependent quantities of the translation probability only to a_j, and assume a simple length model $pr(J|e_1^I) = p(J|I)$, we arrive at the HMM alignment [Vogel & Ney+ 96]:

$$pr(f_1^J|e_1^I) = p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \right] . \tag{6.7}$$

As starting point, we use the solution to model 1. In this thesis, the HMM alignment is used instead of model 2 proposed by [Brown & Della Pietra$^+$ 93].

Model 3, Model 4, and Model 5

Since a detailed description of models 3-5 would go beyond the scope of this thesis we only sketch some basic ideas. For further details, see [Brown & Della Pietra$^+$ 93]. In model 3 and 4, fertilities of target words are introduced. The fertility ϕ_i of a target word in position i is the number of source words it can generate,

$$\phi_i = \sum_{j=1}^{J} \delta(i, a_j) \ . \tag{6.8}$$

Here, $\delta(\cdot,\cdot)$ denotes the Kronecker-function. Similar to model 2, model 3 is a zero-order alignment model including additional fertility parameters. A problem that occurs in both model 3 and 4 is their *deficiency*, that is, the same position can be chosen twice in the source string. Also, a position before the first or beyond the last position may be chosen in model 4. The deficiency problem is solved in model 5 by keeping track of vacant positions in the source string. Models 3-5 are alignment models with increasing complexity. Their exact definitions can be found in [Brown & Della Pietra$^+$ 93] but are not important for the basic ideas presented in this chapter. Starting from model 1, the sequential application of the more complex alignment models together with a search process finally yields the Viterbi alignment a_1^J between words and concepts.

6.4 Natural Language Understanding Using the Source-Channel Paradigm

According to Equation 6.2, the source-channel approach decomposes the posterior probability $pr(c_1^M|w_1^N)$ into two different probability distributions: the translation probability $pr(w_1^N|c_1^M)$ and the language probability $pr(c_1^M)$. Both probability distributions are combined during a search process, which leads to the architecture depicted in Figure 6.2. For the source channel-based approach, we use a phrase-based translation system based on so-called alignment templates [Och & Tillmann$^+$ 99]. Alignment templates have been proven to be very effective in statistical machine translation because they allow many-to-many alignments between source and target words.

Due to $a_j \in \{1, \ldots, I\}$, the basic alignment models introduced in the previous section can capture only 1-to-many alignments. A phrase-based translation system goes one step further and allows many-to-many alignments by providing a two-level alignment: a phrase level alignment and a within-phrase many-to-many word-level alignment. The key idea for obtaining many-to-many word alignments is to symmetrize the training directions. By performing a training in both translation directions (from source to target and from target

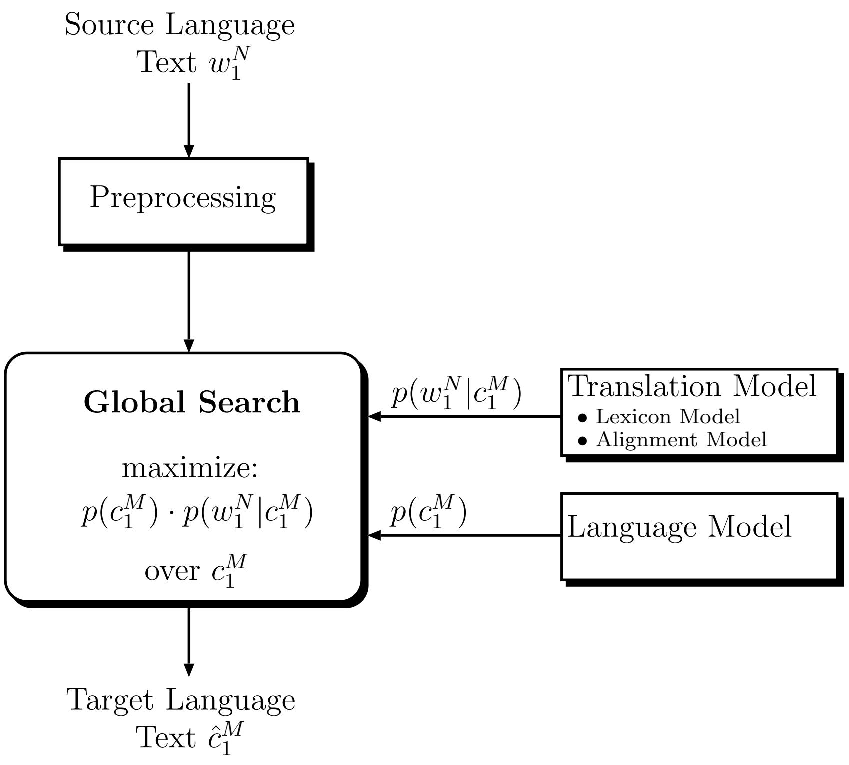


Figure 6.2: Architecture of the natural language understanding component using a statistical machine translation approach based on the source-channel paradigm.

to source), we obtain two Viterbi alignments a_1^J and b_1^I for each sentence pair, which can be merged. Although 1-to-many alignments are sufficient because of the restriction that each word must align to exactly one concept, alignments obtained through symmetrization and merging show a higher precision with respect to the alignment quality [Och & Ney 03]. Let the source and target sentence be segmented into K word-groups describing the phrases:

$$\begin{aligned} e_1^I &= \widetilde{e}_1^K, \;\; \widetilde{e}_k = e_{i_{k-1}+1}, \ldots, e_{i_k}, \;\; k = 1, \ldots, K \\ f_1^J &= \widetilde{f}_1^K, \;\; \widetilde{f}_k = f_{j_{k-1}+1}, \ldots, f_{j_k}, \;\; k = 1, \ldots, K \ . \end{aligned}$$

By decomposing the translation probability with the above-mentioned definitions, we arrive at the following first-order decomposition:

$$pr(f_1^J|e_1^I) = \sum_{\widetilde{a}_1^K, K} \prod_{k=1}^{K} p(\widetilde{a}_k|\widetilde{a}_{k-1}) \cdot p(\widetilde{f}_k|\widetilde{e}_{\widetilde{a}_k}) \ . \tag{6.9}$$

With $z = (\widetilde{e}\,', \widetilde{f}\,', \widetilde{a}\,')$ denoting an alignment template, we obtain $p(\widetilde{f}_k|\widetilde{e}_{\widetilde{a}_k}) = \sum_z p(z|\widetilde{e}_{\widetilde{a}_k}) \cdot p(\widetilde{f}_k|z, \widetilde{e}_{\widetilde{a}_k})$. The phrase translation probability $p(\widetilde{f}_k|z, \widetilde{e}_{\widetilde{a}_k})$ is then decomposed according

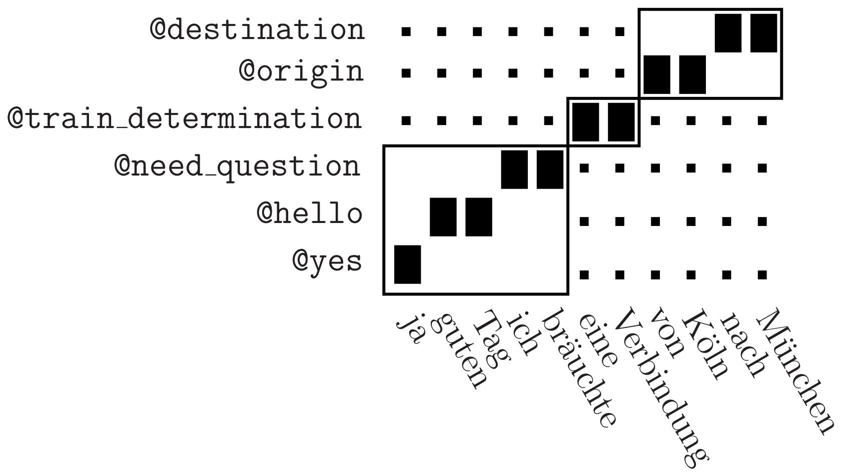


Figure 6.3: Phrase-based translation with alignment templates for the utterance "yes, hello, I need a connection from Cologne to Munich". The words on the x-axis denote the source words, the words on the y-axis are the concepts. The rectangles mark phrase boundaries, the solid boxes within the rectangles describe word-to-concept alignments.

to the following equation:

$$p(\widetilde{f}_k|(\widetilde{e}\,',\widetilde{f}\,',\widetilde{a}\,'),\widetilde{e}_{\widetilde{a}_k}) = \delta(\widetilde{e}_{\widetilde{a}_k},\widetilde{e}\,')\cdot\delta(\widetilde{f}_k,\widetilde{f}\,')\cdot\prod_{j=j_{k-1}+1}^{j_k} p(f_j|\widetilde{a}\,',\widetilde{e}_{\widetilde{a}_k})\,. \tag{6.10}$$

The Kronecker functions $\delta(\cdot,\cdot)$ ensure that only those alignment templates z are chosen that are consistent with $\widetilde{f}_k$ and $\widetilde{e}_{\widetilde{a}_k}$. The probability $p(f_j|\widetilde{a}\,',\widetilde{e})$ can be decomposed in the following way:

$$p(f_j|\widetilde{a}\,',\widetilde{e}) = \sum_{i=1}^{I} p(i|j;\widetilde{a}\,')\cdot p(f_j|e_i) \tag{6.11}$$

$$p(i|j;\widetilde{a}\,') = \frac{\widetilde{a}\,'(i,j)}{\sum_{i\,'}\widetilde{a}\,'(i\,',j)}\,, \tag{6.12}$$

$$\widetilde{a}\,'(i,j) \coloneqq \begin{cases} 1 & \text{if } (i,j) \text{ are linked in the merged alignment} \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

The merged alignment in Equation 6.13 is computed from the alignment sets $A = \{(a_j,j)|a_j > 0\}$ and $B = \{(i,b_i)|b_i > 0\}$ that are combined into one alignment matrix $\mathcal{A}$ by applying the intersection of both alignments. This intersection is afterwards refined by adding further alignment links that satisfy certain neighborhood constraints. Details can be found in [Och & Ney 03]. Doing this, we can explicitly model many-to-many alignments. Figure 6.3 depicts an example of a phrase-based translation for the input sentence "yes, hello, I need a connection from Cologne to Munich". The rectangles describe possible alignment templates. The solid boxes within the rectangles mark word-to-word alignments.

6.4.1 Training

In machine translation, source words that have no correspondence on the target side are mapped onto the empty word. In contrast to translations between natural languages, we do not allow empty words, that is, each word of the source sentence must be aligned to a concept of the target sentence since words that do not contribute to the meaning of a sentence are explicitly modeled by a filler concept. The same holds if we exchange source and target language. This results in the following training procedure:

1. The fertility for the empty word is explicitly set to 0.

2. We compute the Viterbi alignment for each sentence of the training corpus using $1^4H^43^44^45^4$ as sequence of alignment models[2], that is, we first apply 4 iterations of model 1 training and continue with 4 iterations of HMM alignment and so on. Finally, we stop after 4 iterations of model 5. This is done for both directions, source to target and target to source. For computing the alignments we use the publicly available Giza++ toolkit [Och & Ney 03].

3. We compute the alignment matrices using intersections with refinements as combination method.

Further details concerning the training of alignment templates can be found in [Och 02a].

6.4.2 Search

According to Equation 6.2, both probability distributions are combined during a search process, which leads to the architecture depicted in Figure 6.2. If we plug in the phrase-based translation model together with a standard left-to-right trigram language model into the source-channel approach, we obtain the following search criterion in maximum approximation, which is used in combination with beam search:

[2] This sequence of alignment models turned out to be optimal for Taba. TelDir uses a slightly different sequence (see Section 6.9 for details).

$$\hat{e}_1^I = \underset{e_1^I, I}{\operatorname{argmax}} \left\{ pr(e_1^I) \cdot pr(f_1^J | e_1^I) \right\} \tag{6.14}$$

$$\begin{aligned} &= \underset{e_1^I, I}{\operatorname{argmax}} \left\{ \prod_{i=1}^{I} p(e_i | e_{i-2}^{i-1}) \cdot \max_{\tilde{a}_1^K, \tilde{z}_1^K, K} \left\{ \prod_{k=1}^{K} \cdot \right. \right. \\ &\qquad \left. \left. \cdot\, p(\tilde{a}_k | \tilde{a}_{k-1}) \cdot\ p(z_k | \tilde{e}_{\tilde{a}_k}) \cdot\ p(\tilde{f}_k | z_k, \tilde{e}_{\tilde{a}_k}) \right\} \right\} . \end{aligned} \tag{6.15}$$

6.5 Natural Language Understanding Using Maximum Entropy

Alternatively to the source-channel approach, we can directly model the posterior probability $pr(c_1^M | w_1^N)$ in Equation 6.1. A well-founded framework for doing this is maximum entropy (ME). This framework allows for integrating information from many heterogeneous information sources for classification. The data for a classification problem is described as a number of problem-specific features $h_l, l = 1, \ldots, L$. Each feature h_l corresponds to a constraint of the model and is associated with a model parameter λ_l. Among all possible models that satisfy these constraints, the model with maximum entropy is computed during the training phase [Berger 97], that is, the resulting model is consistent with the features observed on the training data, but otherwise makes the fewest possible assumptions about the distributions. Within this framework, the posterior probability can be modeled as follows:

$$pr(c_1^M | w_1^N) = p_{\lambda_1^L}(c_1^M | w_1^N) = \frac{\exp\left[\sum\limits_{l=1}^{L} \lambda_l h_l(c_1^M, w_1^N) \right]}{\sum\limits_{c'^M_1} \exp\left[\sum\limits_{l=1}^{L} \lambda_l h_l(c'^M_1, w_1^N) \right]} . \tag{6.16}$$

The architecture of the ME approach is depicted in Figure 6.4. The feature functions in Equation 6.16 depend on full sentences. Similar to [Papineni & Roukos$^+$ 97], we could generate a set of candidate translations and apply the ME framework on top of this candidate set, so that the λ_l parameters can be estimated such that we obtain better translations. This step would require a separate decoder to generate the candidate translations, which we want to avoid. If we can formulate the NLU task directly within the ME framework we will not depend on an additional classifier generating lists of candidate translations, but would define feature functions on the basis of words and phrases and apply them directly on the source sentences. Furthermore, we do not just want to combine the knowledge sources from the previous section in a log linear way, but also want to overcome some shortcomings of the alignment templates approach. A

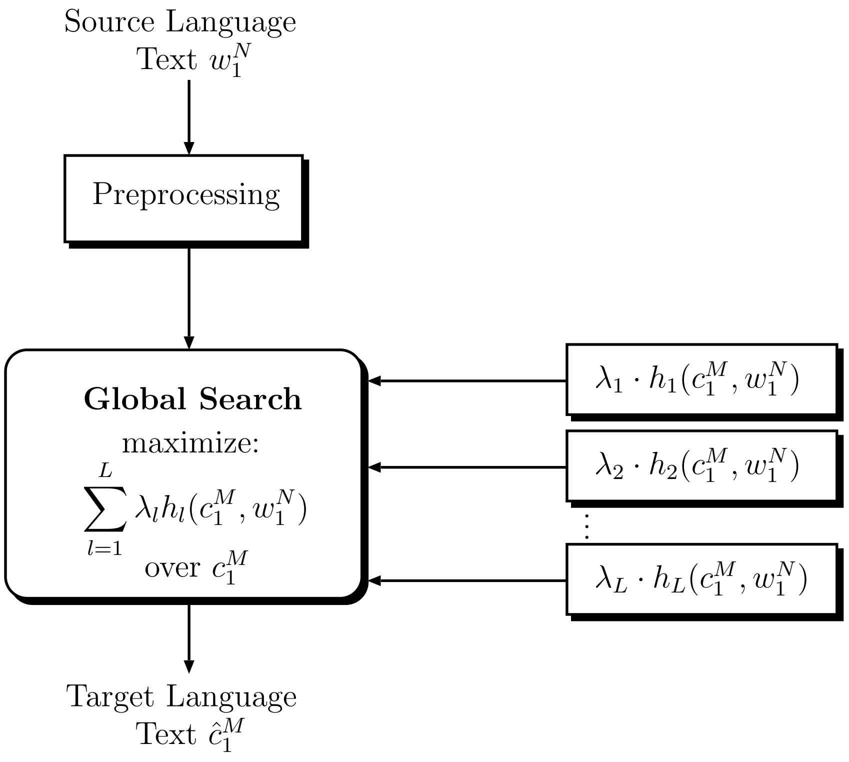

Source Language
Text w_1^N
Preprocessing
Global Search
maximize:
$\sum_{l=1}^{L} \lambda_l h_l(c_1^M, w_1^N)$
over c_1^M
$\lambda_1 \cdot h_1(c_1^M, w_1^N)$
$\lambda_2 \cdot h_2(c_1^M, w_1^N)$
$\lambda_L \cdot h_L(c_1^M, w_1^N)$
Target Language
Text $\hat{c}_1^M$

Figure 6.4: Architecture of the natural language understanding component using a statistical machine translation approach based on the maximum entropy framework.

shortcoming of the alignment templates approach is that each word can be used in at most one template. Overlapping templates are not allowed due to the segmentation into K word groups. For the ME approach, we still want to assign each word to one concept, but we also want to use the context of each word in a more flexible way such that each word can be used in more than one feature function. To solve this problem in the ME approach, we proceed as follows. We slightly modify the NLU task such that source and target sentence are of the same length ($N = M$), yielding a one-to-one correspondence between words and concepts.[3] As discussed in Section 6.3, mapping compounds to concepts is not a problem. A case that occurs more frequently is that several contiguous words are mapped onto a single concept. Therefore, we distinguish whether a word belongs to an initial or a non-initial concept. This modeling was explicitly required in shallow parsing tasks [Tjong Kim Sang & Buchholz 00]. In the context of NLU it was introduced in [Bender & Macherey$^+$ 03]. With these changes, our classification model becomes very similar to maximum entropy Markov models (MEMMs), which are

[3] Because a concept can cover multiple words, this will turn the notion of a concept into a fractional concept that we call a *concept tag*. Thus, a consecutive sequence of similar concept tags then forms a concept. For ease of terminology we will continue to speak of concepts rather than of tags.

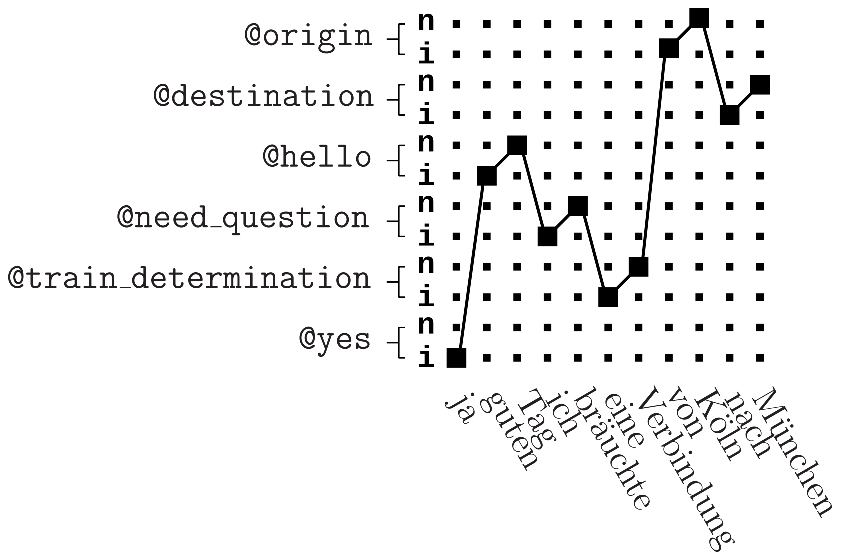


Figure 6.5: Example of a sentence-to-concept mapping for the utterance "yes, hello, I need a connection from Cologne to Munich" using maximum entropy. Each concept starts with an initial concept marked by 'i' and optionally continues with a non-initial concept denoted by 'n'.

often used in information extraction tasks [McCallum & Freitag$^+$ 00]. The difference to a standard information extraction problem is that for the NLU task described in this thesis, we do not know the correct alignment between words and concepts during training, that is, word and concept sequences have different lengths. Thus, MEMMs are not directly applicable here. We solve this problem by first computing the Viterbi alignment between words and concepts on our training data and then by keeping the alignment fixed for all subsequent training steps. Figure 6.5 depicts a one-to-one mapping applied to a sentence concept string pair from the German TABA corpus. We assume that the decisions only depend on a window $w_{n-2}^{n+2} = w_{n-2}, \ldots, w_{n+2}$ around word w_n and on the predecessor concept. Thus, we can factorize the posterior probability of Equation 6.1 and obtain the following first-order model:

$$pr(c_1^N|w_1^N) = \prod_{n=1}^{N} pr(c_n|c_1^{n-1}, w_1^N) \tag{6.17}$$

$$= \prod_{n=1}^{N} p_{\lambda_1^L}(c_n|c_{n-1}, w_{n-2}^{n+2}) \ . \tag{6.18}$$

Because of the distinction between initial and non-initial concepts, we must ensure that a non-initial concept only follows its corresponding initial one, which can be guaranteed by appropriate transition features.

6.5.1 Feature Functions

In the following, we describe a set of binary valued feature functions that are used for text-based NLU. Elements $d, d_k \in D = \{-2, \ldots, 2\}$ refer to positions in the source string. We shall introduce additional feature functions when we apply the ME framework on speech-based inputs.

Lexical Features

The lexical feature $h_{w,d,c}$ employs lexical information. The parameter d allows for taking context words into account. Formally, the feature is defined as follows:

$$h_{w,d,c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \delta(w_{n+d}, w) \cdot \delta(c_n, c) \qquad d \in \{-2, ..., 2\} \ . \tag{6.19}$$

It will only fire if the word w_{n+d} matches w and if the prediction for the current concept c_n is equal to c. Here, $\delta(\cdot, \cdot)$ denotes the Kronecker-function.

Prefix and Suffix Features

In case that a test corpus contains words that were not observed in the training corpus (unknown words), the necessity for vocabulary independent features arises. To achieve this, we generalize the words by looking at their prefixes or suffixes. Let $w_n = \alpha\beta$ be a decomposition of word w_n such that α is a prefix of w_n and β is its suffix. If the prefix (suffix) of w_n is equal to a given prefix (suffix), these features will fire:

$$h_{\overline{\alpha},c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \begin{cases} 1 & \text{if } \exists \alpha, \beta : w_n = \alpha\beta \wedge \alpha = \overline{\alpha} \wedge c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{6.20}$$

$$h_{\overline{\beta},c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \begin{cases} 1 & \text{if } \exists \alpha, \beta : w_n = \alpha\beta \wedge \beta = \overline{\beta} \wedge c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{6.21}$$

Capitalization Features

A capitalization feature will fire if w_n starts with a capitalized letter, has an internal capital letter, or is all capitalized:

$$h_{\text{cap},c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \begin{cases} 1 & \text{if } w_n \text{ starts with a capital letter } \wedge \ c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{6.22}$$

$$h_{\text{int},c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \begin{cases} 1 & \text{if } w_n \text{ has an internal capital letter } \wedge \ c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{6.23}$$

$$h_{\text{all},c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \begin{cases} 1 & \text{if } w_n \text{ is all capitalized } \wedge \ c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{6.24}$$

Capitalization features can be used to detect new proper names that were not observed during training.

Transition Features

Transition features model the dependence on the predecessor concept:

$$h_{c',0,c}(c_{n-1}^{n}, w_{n-2}^{n+2}) = \delta(c_{n-1}, c') \cdot \delta(c_n, c) \ . \tag{6.25}$$

This feature will fire if the prediction c_n of the current class is equal to c and the predecessor concept c_{n-1} is equal to c'.

Prior Features

The single concept priors are incorporated by prior features. The prior feature h_c describes the prior knowledge of the concept c on the training corpus and is the concept's unigram count:

$$h_c(c_{n-1}^{n}, w_{n-2}^{n+2}) = \delta(c_n, c) \ . \tag{6.26}$$

Compound Features

The feature functions defined so far produce only features that refer to single words or concepts. To enable word phrases and word-concept combinations, we introduce the following compound features:

$$\begin{aligned} h_{\{z_1^K, d_1^K\},c}\,(c_{n-1}^{n}, w_{n-2}^{n+2}) &= \prod_{k=1}^{K} h_{z_k,d_k,c}\,(c_{n-1}^{n}, w_{n-2}^{n+2}) \\ z_k \in \{w, c'\} \ , \ d_k &\in \{-2, ..., 2\} \ , \end{aligned} \tag{6.27}$$

where K defines the length of a compound feature. For $z_k = w$ the compound feature is constructed of lexical features. For $z_k = c'$ it corresponds to multiple transition features with the additional constraint that $d_k = 0$ so that it matches Equation 6.25.

6.5.2 Training

The objective function of maximum entropy is a convex function. The convexity of the objective function prohibits the incorporation of hidden variables. Thus, it is not possible to introduce the alignment as a hidden variable as we did in Section 6.4. Therefore, we will determine the alignment beforehand and keep it fixed during training. For training, we consider the set R of aligned training sentences to form a single long sentence. As training criterion, we use the maximum class posterior probability criterion:

$$\hat{\lambda}_1^L = \underset{\lambda_1^L}{\operatorname{argmax}} \left\{ \sum_{r=1}^{R} \sum_{n=1}^{N_r} \log p_{\lambda_1^L}(c_n | c_{n-1}, w_{n-2}^{n+2}) \right\} \ . \tag{6.28}$$

This corresponds to maximizing the likelihood of the ME model. The direct optimization of the posterior probability in Bayes' decision rule is referred to as discriminative training since we directly discriminate the numerator expression modeling the correct class against the denominator expression modeling all competitive classes including the correct one. Thus, we directly take into account the overlap between the probability distributions, which is the main cause for classification errors. Since the optimization criterion is convex, there is only a single optimum and no convergence problems occur. To train the model parameters λ_1^L we use the *generalized iterative scaling (GIS)* algorithm [Darroch & Ratcliff 72].

Models trained within the maximum entropy framework tend to overfit on training data. To avoid overfitting, [Chen & Rosenfeld 99] have suggested a smoothing method where a Gaussian prior on the parameters is applied. Instead of maximizing the probability of the training data, we now maximize the probability of the training data times the prior probability of the model parameters:

$$\hat{\lambda}_1^L = \underset{\lambda_1^L}{\operatorname{argmax}} \left\{ \log p(\lambda_1^L) + \sum_{r=1}^{R} \sum_{n=1}^{N_r} \log p_{\lambda_1^L}(c_n | c_{n-1}, w_{n-2}^{n+2}) \right\} , \tag{6.29}$$

where

$$p(\lambda_1^L) = \prod_{l=1}^{L} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{\lambda_l^2}{2\sigma^2} \right] \tag{6.30}$$

with the standard deviation σ as the smoothing parameter.

6.5.3 Search

For decoding a sequence of words w_1^N, we perform a breadth-first search where all hypotheses are expanded position-synchronously, that is, we start with expanding hypotheses for word w_1 at position 1, then continue with hypotheses for word w_2 at position 2, and so on. To solve the search problem in maximum approximation (*Viterbi search*) we define the auxiliary quantity $Q_{\max}(n, c)$ as the maximal probability of the best partial path covering positions $1, \ldots, n$ and ending in concept c [Bender 02]. This leads to the following recursive equations, which are solved using dynamic programming

$$Q_{\max}(0, \$) = 1 \tag{6.31}$$

$$Q_{\max}(n, c) = \max_{c'} \left\{ Q_{\max}(n-1, c') \cdot p_{\lambda_1^M}(c | c', w_{n-2}^{n+2}) \right\} . \tag{6.32}$$

Here, \$ denotes the sentence start symbol. If no pruning is applied, the Viterbi search is guaranteed to find the most likely sequence of concepts with respect to the underlying models [Borthwick & Sterling+ 98].

6.6 Search Variants and Implementation

In Section 6.5, we have defined NLU as a tagging problem, that is, we assign each word to a concept tag. With the Markov model assumption from Equation 6.18 we can look at the sequence of concept tags as a Markov chain (Markov model tagging). Here, the tags correspond to the states of the Markov model. Although, we have determined the training tag sequence in an unsupervised manner by using different alignment models, we keep the tag sequence fixed during training. Therefore, the states of the Markov model are visible during training. For decoding, we can keep this perspective and assign each word a tag. Thus, we have a Markov model in decoding where the sequence of states is not hidden. Since we a more interested in the sequence of concepts rather than a sequence of concept tags, we obtain the concepts by reducing the tag sequence to its initial start tags, which is done in a post-processing step, and by aligning each word that was tagged with the corresponding continue tag to the start tag.

Alternatively, we can organize the search such that the output of the decoder is directly a sequence of concepts. This can be achieved by using a hidden Markov model decoder where the tag sequence remains hidden. Each concept is then modeled by a Markov automaton consisting of two states (see Appendix B.3). For both decoders, the Viterbi algorithm is used in order to find the most probable sequence.

Because the concept language consists of only 22 concepts for the TelDir corpus and 27 concepts for the Taba corpus respectively we could basically perform a full search in order to find the global optimum.[4] For offline experiments this is acceptable. However, due to realtime constraints we have to constrain the search. Note that we do not perform natural language understanding for only the first best recognition result but for a full N-best list when we combine speech recognition and natural language understanding (see Section 6.9.6). Thus, the actual time complexity plays an important role. The search and pruning parameters can be adjusted such that we do not encounter any search errors on the test sets while avoiding to perform a full search, which leads to a significant reduction in decoding time.

From an implementation point of view, there is not much difference between a decoder for speech recognition and a decoder for natural language understanding. In fact, both decoders are $m-1$th order decoders, that is, they take into account the decision of $m-1$ predecessor words or concepts. The set of flat concepts used for natural language understanding defines a dictionary similar to a pronunciation dictionary used in speech recognition. Here, each concept is composed of initial and continue tags, which corresponds to a sequence of phonemes in a pronunciation dictionary. The language models used for both decoders are m-gram models. The scorer changes from a Gaussian mixture model to an exponential model. Because the above listed items are all knowledge sources provided to the decoder in some parameterized form, this means

[4] Due to initial and continue tags this means 43 tags for TelDir and 53 tags for Taba in the maximum entropy approach. Here, the filler concept is represented by only one tag for both tasks.

that the decoder for speech recognition can be reused for tagging, which significantly simplifies software development. For the implementations done for this thesis, the speech recognition module and the language understanding module inherit a templatized tree search class where knowledge sources like dictionaries or language models are provided as templates and, thus, can be easily replaced. As a side effect, the natural language understanding component inherits all methods, such as pruning techniques, extraction of graphs and N-best lists, computation of confidence measures, and so on.

6.7 Confidence Measures for Natural Language Understanding

In this section, we introduce confidence measures for natural language understanding that shall provide a score for the reliability of each concept being produced by the NLU module. Confidence measures for natural language understanding can be defined in a similar way as confidence measures for speech recognition. The concept posterior probability $p([c;\nu,n]|w_1^N)$ of a concept c with starting position ν and ending position n, given the sequence of input words w_1^N, can directly be interpreted as a confidence measure.[5] According to Equation 6.18, the maximum entropy framework assigns posterior probabilities to concept tags, which we could use as confidence measure. However, this would condition the posterior probability of a concept tag to only a small window of words. Furthermore, we are more interested in the confidence of concepts rather than individual concept tags. We could circumvent this by computing the average posterior probability of all contiguous concept tags that form a single concept and interpret this value as a confidence score, but for the cost of losing the normalization property.

A more suitable approach is to use a hidden Markov model decoder in order to produce concept graphs on which we can then apply the same techniques described in Section 5.3 for speech recognition tasks. Denote $[c;n]_1^M = [c_1; n_0+1, n_1], \ldots, [c_M; n_{M-1}+1, n_M]$ a sequence of concept hypotheses where $n_0 = 0$ and $n_M = M$. Further, define $\nu_m \coloneqq n_{m-1}+1$. Then we define the posterior probability of a concept c with starting and ending position (ν, n) as follows:

$$p([c;\nu,n]|w_1^N) = \sum_{[c;n]_1^M} \sum_{\substack{m=1: \\ [c;\nu_m,n_m]=[c;\nu,n]}}^{M} p([c;n]_1^M|w_1^N) \ . \tag{6.33}$$

Likewise to Section 5.3, this quantity can be efficiently computed using a forward-backward algorithm. By relaxing start and end position of a concept, we arrive at the final definition for the concept-based confidence measure:

$$\mathcal{C}_{\text{NLU}} \coloneqq \tilde{p}([c;\nu,n]|w_1^N) = \max_{\tilde{n}:\nu\leqslant\tilde{n}\leqslant n} \sum_{(\nu',n'):\nu'\leqslant\tilde{n}\leqslant n'} p([c;\nu',n']|w_1^N) \ . \tag{6.34}$$

[5] For the case of concept tags, ν is equal to n.

The reason for relaxing start and end positions is that a concept may occur in slightly different positions. Another way to account for this when calculating concept posterior probabilities is to determine the Levenshtein alignment between the sentence c_1^M under consideration and all other concept sentences [Macherey & Bender+ 03]. In the context of machine translation, this approach was investigated in [Ueffing 06]. The summation is then performed over all sentences containing concept c in position m or in a position Levenshtein-aligned to m. Denote $\mathcal{L}(c_1^M, \tilde{c}_1^{\tilde{M}})$ the Levenshtein-alignment between sentences c_1^M and $\tilde{c}_1^{\tilde{M}}$, and $\mathcal{L}_m(c_1^M, \tilde{c}_1^{\tilde{M}})$ that of concept c in position m in c_1^M. Then, the concept posterior probability of concept c occurring in a position Levenshtein-aligned to m is given by

$$p_{m,Lev}(c|w_1^N, c_1^M, \mathcal{L}) = \frac{p_{m,Lev}(c, w_1^N, c_1^M, \mathcal{L})}{\sum\limits_{c'} p_{m,Lev}(c', w_1^N, c_1^M, \mathcal{L})} , \tag{6.35}$$

where

$$p_{m,Lev}(c, w_1^N, c_1^M, \mathcal{L}) = \sum_{\tilde{M}, \tilde{c}_1^{\tilde{M}}} \delta(c, \mathcal{L}_m(c_1^M, \tilde{c}_1^{\tilde{M}})) \cdot p(w_1^N, \tilde{c}_1^{\tilde{M}}) . \tag{6.36}$$

In [Macherey 98, Wessel 02], it is shown that for speech recognition tasks, word posterior probability-based confidence measures estimated on word graphs clearly outperform confidence measures estimated on N-best lists. In [Ueffing 06], it is shown that for machine translation tasks, there is no significant difference between posterior-based confidence measures estimated on word graphs and N-best lists, given that the reference class labels are determined with respect to word error rate or position independent word error rate. Because we consider natural language understanding as a translation problem with a monotonicity constraint, it is not clear whether graph-based estimations will outperform the N-best list approach. Therefore, we will investigate both variants and compute confidence measures on graphs as well as on N-best lists.

For natural language understanding, we define the confidence error rate by means of the slot error rate, similar as we based the confidence error rate for speech recognition on the word error rate. Thus, we define the confidence error rate (CER) as the number of incorrectly assigned confidence class labels divided by the total number of generated concepts. Likewise, the baseline CER is given by the number of substitutions and insertions divided by the total number of generated concepts.

6.8 Combining Speech Recognition and Natural Language Understanding

As discussed in Section 2.2, the sequential application of speech recognition and language understanding can turn out to be suboptimal. Therefore, we need an approach that allows for a tighter coupling between both components. Furthermore, the approach used should allow us to easily integrate additional knowledge sources. The automatic speech recognition module is based on the source-channel paradigm. Since it is difficult to integrate additional knowledge sources into this framework, we will use N-best lists for representing the most likely word hypotheses that are promising candidates for the natural language understanding task, and perform all following investigations on this structure.

Using N-best lists rather than word graphs has the advantage that also sentence-based features can be employed. The features will be combined in a log-linear model and the corresponding feature weights are trained in a discriminative manner using the minimum error rate criterion. We will proceed as follows: we briefly review the minimum error rate criterion and describe the optimization of the feature weights λ_1^L using the Line Sweep algorithm. The optimization is an iterative procedure that is applied on an N-best repository. As error criterion to be minimized we will use both the slot error rate and the word error rate. Thus, we can investigate whether for a given set of feature functions the natural language understanding component can benefit from speech recognition features and, vice versa, whether the automtatic speech recognition component can benefit from features derived from the language understanding component.

6.8.1 Minimum Error Criterion

Minimum error criteria have a wide range of applications in pattern recognition tasks. In the field of automatic speech recognition, minimum error criteria were for example employed in [Juang & Chou+ 95, Evermann 99]. In [Beyerlein 00], the acoustic scores are combined with multiple language models in a log-linear approach yielding a significant reduction in terms of word error rate for a state-of-the-art speech recognition system. In the context of machine translation, it was first used in [Och 03]. To the best of our knowledge minimum error criteria have not been applied to SLU tasks. Since we regard NLU as a translation task, we can define the minimum error criterion similar to the approach proposed in [Och 03] with the extension that we aim at minimizing both the word error rate and the slot error rate. The challenging task is then to find appropriate feature functions that result in a reduction of both error rates.

Denote $\mathbf{x} = x_1^T$ the sequence of acoustic observation vectors, $\mathbf{w} = w_1^N$ the sequence of words, and $\mathbf{c} = c_1^M$ the sequence of concepts, then we want to find that sequence of parameters $\hat{\lambda}_1^L$ that minimizes the number of errors denoted by an error function $E(\cdot,\cdot)$

$$\hat{\lambda}_1^L = \operatorname*{argmin}_{\lambda_1^L} \left\{ \sum_{s=1}^{S} E\big((\mathbf{c}_s, \mathbf{w}_s), (\hat{\mathbf{c}}, \hat{\mathbf{w}})(\mathbf{x}_s; \lambda_1^L)\big) \right\} \tag{6.37}$$

where

$$(\hat{\mathbf{c}}, \hat{\mathbf{w}})(\mathbf{x}_s; \lambda_1^L) = \operatorname*{argmax}_{(\mathbf{c},\mathbf{w})\in\mathcal{R}_s} \left\{ \sum_{l=1}^{L} \lambda_l h_l(\mathbf{c}, \mathbf{w}, \mathbf{x}_s) \right\}. \tag{6.38}$$

Here, s denotes the sentence index and $(\hat{\mathbf{c}}, \hat{\mathbf{w}})$ denotes the maximizing concept and word sequence, respectively. $\mathcal{R}_s$ defines the repository of candidates, which for each utterance

Algorithm 2 Minimum Error Training

```
Input: initial weights $\lambda_1^L$
Output: optimized weights $\hat{\lambda}_1^L$
  repeat
    Generate $N$-best repositories with current $\lambda_1^L$
    for all dimensions $l$ do
      for all sentences $s$ do
        Compute upper envelope and error statistics
      end for
      Merge error statistics
      Search for optimal $\gamma$ and determine error reduction $\Delta e_l$
      $\lambda'_l \leftarrow \lambda_l + \gamma$
    end for
    $\lambda_l \leftarrow \lambda'_{\hat{l}}$    with $\hat{l} = \operatorname*{argmin}_l \{\Delta e_l\}$
  until convergence
```

s is the product of the N-best candidates from the ASR module times the $\mathcal{N}$-best list of concept strings for each recognized sentence candidate. For ease of terminology, we will refer to pairs $(\mathbf{c}, \mathbf{w})$ of translation and sentence hypotheses as candidate translations. Once produced by the two decoding steps they are kept fixed throughout the algorithm.

6.8.2 Line Sweep Algorithm

The sum of weighted feature functions can be expressed as a scalar product:

$$\sum_{l=1}^{L} \lambda_l h_l(\cdot, \cdot) = \left(\lambda_1^L\right)^\top \cdot \left(h_1^L\right). \tag{6.39}$$

In each iteration we try to change the weight vector λ_1^L such that the total number of errors is minimized. This is an L-dimensional optimization problem. For simplicity, we loop over all dimensions and optimize each dimension separately. Let γ denote the change of weight λ_l for a given dimension l. Then this change can be expressed as:

$$g_l(\gamma):\ \left(\lambda_1^L + \gamma \cdot d_1^L\right)^\top \cdot \left(h_1^L\right) = \underbrace{\left(d_1^L\right)^\top \left(h_1^L\right)}_{\text{slope } a} \cdot \gamma + \underbrace{\left(\lambda_1^L\right)^\top \left(h_1^L\right)}_{\text{offset } b} \tag{6.40}$$

where $\left(d_1^L\right)$ is the L-dimensional unit vector with component $l = 1$. For each utterance $\mathbf{x}_s$ there is a repository $\mathcal{R}_s$ of candidate translations. Each candidate translation defines a line g_l with slope a and offset b, and is associated with its error with respect to the reference translation. For the one-dimensional optimization problem, the goal is now to adjust parameter λ_l such that the error is minimal. From a geometric point of view the

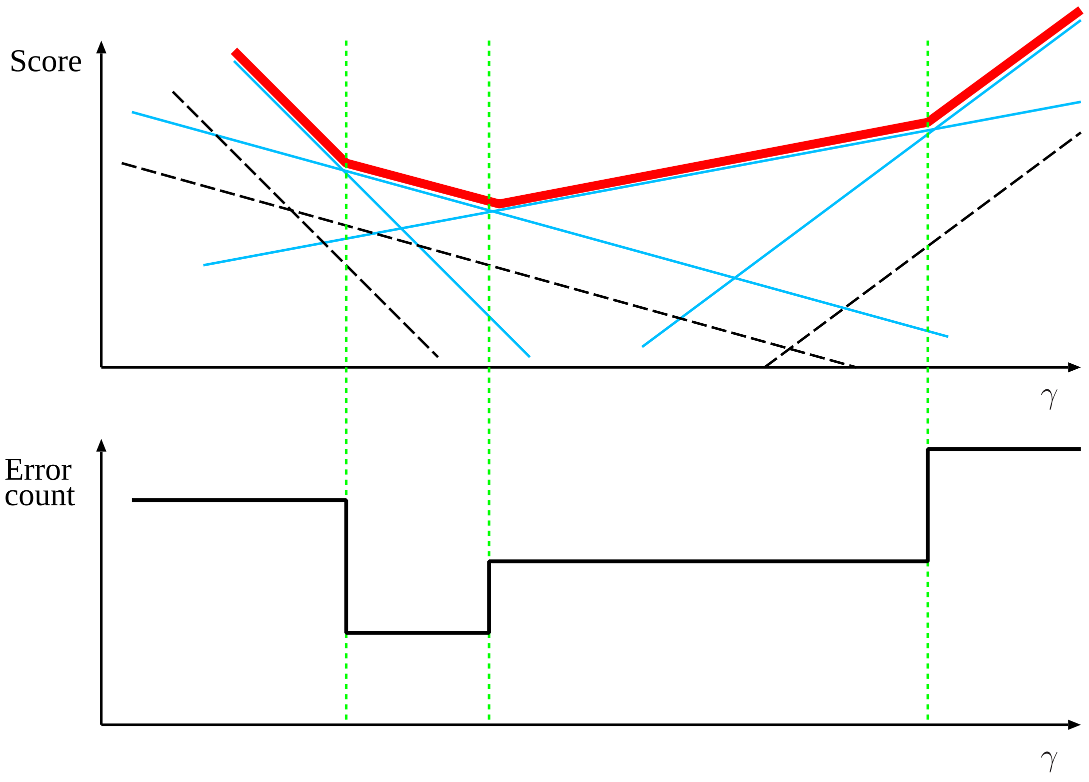


Figure 6.6: Upper envelope and error counting. The bold red line segments define the upper envelope of the set of solid blue lines. The dashed black colored lines are parallel lines that have a lower score than the blue lines and can be discarded for the computation of the upper envelope. Each segment of the upper envelope defines an area of constant errors. By projecting the upper envelope to the x-axis, we obtain a histogram of the total number of errors. The total number of errors can only change at phase transitions, that is, at the points of intersections of the line segments.

intersection points of the lines divide each line into at most $|\mathcal{R}_s|$ line segments. The line segments form polyhedrons partitioning the plane into regions of equal numbers of errors. However, due to the argmax decision rule we only need to consider those line segments with maximal scores, that is, which belong to the *upper envelope*. A line segment belongs to the upper envelope if a vertical upward ray starting at this line segment does not hit any other line. Since each line is associated with its error rate we can search for that segment along the upper envelope that minimizes the error. By projecting the points of intersections of the corresponding line segments together with their error counts onto the x-axis, we obtain a histogram of the total error counts. The phase transitions mark the positions where the total error may change. The basic principle is depicted in Figure 6.6.

The upper envelope can be efficiently computed using the *line sweep* algorithm, which was introduced in [Bentley & Ottmann 79]. A vertical line is moved from the far left to the far right of the plane, and the exact order of the intersections with the segments is maintained as the line sweeps along. All vertices in the arrangement of segments are detected in this process. From the point of view of the sweep line, the segments can be viewed as points that move up and down the line [Basch & Guibas$^+$ 97]. If we compute the error counts for all sentences and merge the projected points of intersections together with their error counts, we obtain a finite sequence of λ_l values together with their associated error change. A linear search in this sequence yields the optimal parameter λ_l. The pseudo code for this procedure is given in Algorithm 2, which is guaranteed to converge to a local optimum. Once we have determined the optimal parameter set λ_1^L, we can determine the optimal sequence of words and concepts in a rescoring step. Depending on whether we want to find the optimal word sequence, the optimal concept sequence, or both, we define the following decision rules:

$$\hat{\mathbf{w}}(\hat{\mathbf{c}}, \mathbf{x}; \lambda_1^L) = \operatorname*{argmax}_{\mathbf{w}} \left\{ \sum_{l=1}^{L} \lambda_l h_l(\mathbf{c}, \mathbf{w}, \mathbf{x}) \right\} \tag{6.41}$$

$$\hat{\mathbf{c}}(\hat{\mathbf{w}}, \mathbf{x}; \lambda_1^L) = \operatorname*{argmax}_{\mathbf{c}} \left\{ \sum_{l=1}^{L} \lambda_l h_l(\mathbf{c}, \mathbf{w}, \mathbf{x}) \right\} \tag{6.42}$$

$$(\hat{\mathbf{c}}, \hat{\mathbf{w}})(\mathbf{x}; \lambda_1^L) = \operatorname*{argmax}_{(\mathbf{c},\mathbf{w})} \left\{ \sum_{l=1}^{L} \lambda_l h_l(\mathbf{c}, \mathbf{w}, \mathbf{x}) \right\} \tag{6.43}$$

Note that for each case, the parameter set is optimized with respect to the corresponding decision rule.

6.8.3 Feature Functions

We introduce a new set of sentence-based feature functions in order to take ASR and NLU features into account. The features are computed on N-best lists and are log-linearly combined using the minimum error criterion. The N-best lists are annotated with concepts using the maximum entropy-based approach. When taking speech recognition effects into account, it is straightforward to directly employ the source channel-based knowledge sources of a speech recognition module as additional feature functions. Note that in general these feature functions are non-binary features. Furthermore, we omit the one-to-one correspondence between words and initial-continue concepts and use the (c_1^M, w_1^N) notation where w_1^N denotes a speech recognition sentence hypothesis stored in an N-best list, and c_1^M denotes a candidate translation from the NLU module stored in a corresponding $\mathcal{N}$-best list, that is, each recognition result stored in an N-best list produces a set of $\mathcal{N}$-best translations. Thus, a user utterance is represented by a total of $N \times \mathcal{N}$ sentence pair hypotheses. For ease of notation we will not distinguish between N and $\mathcal{N}$ in the following, but rather use the term N-best list. Using these conventions, we can define the following sentence-based feature functions:

Language Model features

The language model feature assigns each candidate sentence derived from an N-best list its language model probability. Here, we use trigram language models and define feature functions for both class-based and non class-based language models. Thus, we obtain the following two feature functions:

$$h_{\text{LM}}(c_1^M, w_1^N, x_1^T) = \prod_{n=1}^{N} p(w_n | w_{n-2}^{n-1}) \tag{6.44}$$

and

$$h_{\text{classLM}}(c_1^M, w_1^N, x_1^T; \mathcal{K}) = \prod_{n=1}^{N} p(w_n | k_n) \cdot p(k_n | k_{n-2}^{n-1}) \ , \tag{6.45}$$

where $\mathcal{K}$ denotes the set of word classes k for the class-based language model.

Acoustic Model Features

Similar to the language model features we use the acoustic score of a sentence hypothesis as additional feature. This yields the following feature function:

$$h_{\text{AM}}(c_1^M, w_1^N, x_1^T) = \prod_{n=1}^{N} p(x_\tau^t | w_n) \ , \tag{6.46}$$

where x_τ^t denotes the sequence of acoustic vectors assigned to word hypothesis w_n.

ASR Sentence Length

The length of an ASR sentence hypothesis is defined as the number of evaluation words contained in this sentence, disregarding silence and noise words. The weight for the sentence length is usually referred to as word penalty. Formally, the feature function is defined as follows:

$$h_{\text{Length}_{\text{ASR}}}(c_1^M, w_1^N, x_1^T) = N \ . \tag{6.47}$$

ASR Posterior Probabilities

The posterior probabilities defined in Section 5.3 can be used as additional features. By defining the posterior probability of a sentence hypothesis as the product of its word posterior probabilities, we arrive at the following definition:

$$h_{\text{Post}_{\text{ASR}}}(c_1^M, w_1^N, x_1^T) = \prod_{n=1}^{N} p([w_n; \tau, t] | x_1^T) \ . \tag{6.48}$$

ASR Confidence Measures

Word posterior probabilities introduced in Section 5.3 are also the basis for word-based confidence measures. Since we consider only sentence-based feature functions, we have to aggregate the word-based confidence measures to a sentence-based confidence measure. For a sentence-based feature function, we define the geometric mean of the word-based confidence values $\tilde{p}$ contained in a sentence hypothesis as the sentence confidence value. This yields the following feature function:

$$h_{\mathcal{C}_{\text{ASR}}}(c_1^M, w_1^N, x_1^T) = \left(\prod_{n=1}^{N} \tilde{p}([w_n; \tau, t] | x_1^T) \right)^{1/N} . \tag{6.49}$$

NLU Model Features

The NLU model feature is the factorized posterior probability obtained from the maximum entropy model defined in Equation 6.18 for a given input sentence w_1^N. This is the first feature function that takes NLU-related knowledge source into account. It is defined as follows:

$$h_{\text{NLU}}(c_1^M, w_1^N, x_1^T) = \prod_{m=1}^{M} p_{\lambda_1^L}(c_n | c_{n-1}, w_{n-2}^{n+2}) . \tag{6.50}$$

NLU Sentence Length

Similar to the ASR sentence length we use the NLU sentence length as an additional feature in order to penalize sentence hypotheses that are too short or too long. The NLU sentence length feature function can be interpreted as a concept penalty and is defined as follows:

$$h_{\text{Length}_{\text{NLU}}}(c_1^M, w_1^N, x_1^T) = M . \tag{6.51}$$

Concept Language Model

A trigram language model trained on concepts is used as additional feature function. The concept language model uses absolute discounting with backing off as smoothing variant. The feature function is defined as follows:

$$h_{\text{LM}_{\text{NLU}}}(c_1^M, w_1^N, x_1^T) = \prod_{m=1}^{M} p(c_m | c_{m-2}^{m-1}) . \tag{6.52}$$

NLU Confidence Measures

Similar to the sentence-based speech recognition confidence measures, we define sentence-based confidence measures for natural language understanding using again the geometric mean of the concept-based confidence measures. This yields the following feature function:

$$h_{\mathcal{C}_{\mathrm{NLU}}}(c_1^M, w_1^N, x_1^T) = \left(\prod_{m=1}^{M} \tilde{p}([c_m; \nu, n] | w_1^N) \right)^{1/M} . \tag{6.53}$$

6.9 Results

In this section, we investigate the proposed approaches on the TelDir and Taba task. In Section 6.9.4, we report experiments using reference transcriptions, that is, the input sentences do not contain recognition errors. In Section 6.9.5, we use transcriptions produced by an automatic speech recognition system as input. Finally, we report results for combining speech recognition and natural language understanding.

6.9.1 Evaluation Metrics

To evaluate the quality of the different approaches, we use the following error criteria:

- **Sentence Error Rate**
  The *sentence error rate (SER)* for natural language understanding tasks is defined as the number of wrongly translated sentences that contain concept errors, normalized by the total number of sentences. Thus, it is similarly defined as the sentence error rate criterion used in speech recognition except that we use concepts (slots) instead of words.

- **Slot Error Rate (a.k.a. Concept Error Rate)**
  The *slot error rate (Slot-ER)* is similar defined to the well known word error rate and is the ratio of the sum of deleted, inserted, and substituted concepts (slots), normalized by the total number of reference concepts.

- **Attribute Error Rate**
  The *attribute error rate (AER)* measures the number of falsely assigned or missed attributes normalized by the total number of attributes. Besides the slot error rate, this measure is the most important criterion because it measures to what extent a certain approach is able to extract the correct values from the input sentences.

- **Word-based Slot Error Rate**
  The *word-based slot error rate (word-based Slot-ER)* is defined as the number of words that have been aligned to wrong concepts normalized by the number of all words. In statistical machine translation, alignments of words between natural language sentence pairs are often ambiguous due to idiomatic expressions or missing function words [Melamed 98]. Therefore, [Och 02b] introduced the notion of sure and possible alignments in order to measure the quality of automatically determined alignments. Within the scope of natural language understanding, we assume that there is only one correct alignment, which is provided by the corpora used. Since we disallow empty words, each word must be aligned to a concept. Thus, we can measure the quality of alignments as the number of words that have been mapped onto the correct concepts (slots). The word-based Slot-ER is a useful measure because quantifying the number of words that have been mapped onto the wrong

Table 6.1: Excerpt of word categories. In a preprocessing step all source word occurrences are mapped onto the corresponding word categories.

Category	Examples	Category	Examples
$CITY	• Aachen	$MONTH	• Januar
	• Köln		• Februar
$DAYTIME	• Morgen	$CARDINAL	• erster
	• Vormittag		• zweiter
$WEEKDAY	• Montag	$NUMBER	• null
	• Dienstag		• eins
$SURNAME	• Schlegel	$FUZZY	• circa
	• Wagner		• gegen

concepts is beyond the scope of the AER. Since the attribute values are extracted for each concept separately from its aligned words, a correct alignment is essential for this step. The AER only measures the proportion of erroneous attributes and gives no hint whether this is caused by an insufficiently designed extraction algorithm or simply a result of wrongly aligned words.

6.9.2 Preprocessing

Each source sentence is preprocessed before the sequence of concepts is determined. Some preprocessing steps are only necessary if speech is used as input modality. However, since these steps leave the text input unaffected, they are included in the following list as well. For both approaches the preprocessing consists of the following steps:

1. **Removal of no-word entities**
   This step mainly affects transcriptions derived from the automatic speech recognition module. In this step, all no-word entities, such as silence, hesitations, and other noise occurrences, are removed.

2. **Normalization of pronunciation variants**
   Similar to the first step, this step only affects transcriptions obtained from the automatic speech recognition module. Pronunciation variants are mapped onto the first pronunciation variant, which is determined by the order the words occur in the pronunciation lexicon.

3. **Categorization**
   To reduce the number of unseen events, proper names, numbers, and date expressions are mapped onto categories. Categories for proper names can easily be obtained from the entries of the database the dialogue system is connected with.

6.9.3 Postprocessing

No postprocessing steps are applied for the ME approach. However, an additional filler-sequence penalty is used for decoding that avoids chains of filler concepts. This filler-sequence penalty is similar to the word penalty in ASR. For the source-channel approach, special postprocessing steps are applied for unknown words. Unknown words are words that do not have an entry in the phrase table and are translated by their identity. The steps are applied in the following order:

1. **Unknown words**
   Unknown words that occur at the beginning of a translated sentence are mapped onto the filler concept. Unknown words that occur after the first position in a translated sentence are assigned to the same concept as their predecessor words.

2. **Sequence of equal concepts**
   A Sequence of equal concepts is reduced to a single concept.

6.9.4 Comparison of Alignment Templates and Maximum Entropy Approach Using Text Input

We first determine the alignments between source and target sentences for the TelDir training corpus using $1^3H^33^44^35^1$ as sequence of models for the source-channel approach (i.e., we apply 3 iterations of model 1, continue with 3 iterations of the HMM model, etc.) and $1^4H^43^44^45^4$ for the ME approach. For the Taba corpus, the sequence of model iterations used is $1^4H^43^44^45^4$ for both the source-channel and the ME approach. The optimal training sequences were determined with respect to the error rates obtained on the development sets.

Feature Selection, Count Cutoffs, Model Order, and Smoothing

Feature selection for the feature functions described in the previous section is done on the development corpus of each task. The selection is always done for complete feature types rather than for individual features. For the lexical features, the count cutoff is set to 1, that is, we keep all triples (w, d, c) observed during training. For prefix and suffix features, we additionally require that a word must consist of at least 10 characters in order to generate a feature. The smoothing parameter σ is set to 3 for all corpora. The following experiments were carried out using a first-order model. Switching from a first-order model to a second-order model is investigated in Section 6.9.4.

Preprocessing using Categorizations

A problem that often occurs in NLU tasks is that many semantically relevant words are not covered by the training corpus. For example, it is unlikely that all proper names are observed during training. To improve the results we use categorizations for both approaches and map words, such as proper names and numbers, onto categories. Such categories can easily be obtained from the entries of the database a dialogue system is connected with. Table 6.1 shows an excerpt of the categories used in the TELDIR and TABA corpus. Because the database does not contain all named entities, a large proportion of proper names occurring in the test data remains unobserved.

Effect of Alignment Models

Tables 6.2 and 6.4 summarize results for both the alignment templates approach and the maximum entropy framework using different alignment models. Both tables present results with and without categorizations. Unlike translations between natural language pairs, word reorderings do not occur in natural language understanding tasks investigated in this thesis. Therefore, we disallow word reorderings for the alignment templates approach by penalizing possible reorderings with additional costs set to infinity. Because the maximum entropy-based approach basically works similar to a tagger the monotonicity of the word-concept alignment is guaranteed for this method.

The maximum entropy results presented in Tables 6.2 and 6.4 are based on the 6 feature types listed in Tables 6.3 and 6.5. Starting with only lexical features, we successively extend the model by including additional feature functions. The results show that the maximum entropy models clearly outperform the alignment templates approach. The quality of the alignment templates approach is achieved within the maximum entropy framework by just including lexical and transition features, and is significantly improved by adding further features.

Another interesting effect when looking at the results obtained without categorizations is that the maximum entropy framework suffers less from unknown words compared to the alignment templates approach. The problem can be mitigated but not eliminated if more training data is available as is the case for the TABA corpus. The reason is that the ME model can use variable-length contexts flexibly. Whereas the alignment template approach either has an appropriate template or has not, and each word can occur in at most one template, the ME model can reuse contexts flexibly for various adjacent words, that is, the same word in position n of the input sentence can be used in multiple feature functions. Furthermore, the ME framework directly models the posterior probability and allows for integrating structural information by using appropriate feature functions in a very simple way as opposed to a source channel-based approach where integrating additional knowledge sources is often very complicated.

Comparing the performance on both the TELDIR and the TABA task, we see that the error rates are much lower for the TABA task than for the TELDIR task, which is caused by the very small number training data available for the TELDIR task. However, we still achieve good results using the maximum entropy framework, even if the training corpus is very small. For both approaches, the length of the local word context plays an important role that shall be investigated in the following.

Table 6.2: Effect of different alignment models on the slot error rate for the TelDir corpus. Error rates are presented with and without categorizations for the alignment templates (AT) approach and the maximum entropy (ME) framework. The ME results were obtained using a first-order model.

corpus TelDir development	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	41.4	26.2	23.4	21.6
HMM	21.4	10.9	9.9	4.5
Model 3	20.4	7.4	7.1	3.4
Model 4	19.9	6.4	7.0	2.6
Model 5	19.6	5.8	6.9	**2.5**

corpus TelDir evaluation	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	32.9	21.9	24.3	21.2
HMM	15.7	10.7	9.8	4.4
Model 3	14.4	8.1	7.3	4.3
Model 4	13.2	6.5	6.7	3.1
Model 5	13.2	6.3	6.5	**3.1**

Table 6.3: Dependence of different error rates on the number of included feature types for the TelDir corpus. The source sentences are preprocessed using categorizations. The results were obtained using a first-order model.

corpus TelDir development	total # features	SER [%]	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	60,287	10.7	5.5	5.7	2.7
+ prior	60,330	10.2	5.0	5.6	2.6
+ transition	62,179	7.2	3.2	5.6	2.3
+ pre- & suffixes	91,290	6.0	2.7	5.3	2.3
+ compound	130,893	5.5	2.5	5.3	2.1
+ capitalization	130,936	5.7	**2.5**	5.3	1.8

corpus TelDir evaluation	total # features	SER [%]	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	60,287	10.1	5.5	6.6	3.4
+ prior	60,330	9.7	5.3	6.4	3.3
+ transition	62,179	7.8	3.5	6.6	2.2
+ pre- & suffixes	91,290	7.1	3.3	6.4	1.6
+ compound	130,893	6.5	3.1	6.3	1.7
+ capitalization	130,936	6.8	**3.1**	6.4	1.6

Table 6.4: Effect of different alignment models on the slot error rate for the Taba corpus. Error rates are presented with and without categorizations for the alignment templates (AT) approach and the maximum entropy (ME) framework. The ME results were obtained using a first-order model.

corpus Taba development	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	13.6	8.8	12.0	12.4
HMM	13.3	8.7	6.6	4.0
Model 3	11.3	7.8	6.4	2.9
Model 4	10.8	6.0	6.2	2.6
Model 5	10.8	5.2	6.2	**2.5**

corpus Taba evaluation	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	13.0	8.3	11.6	11.5
HMM	12.3	7.4	5.7	3.4
Model 3	11.3	6.9	5.3	3.1
Model 4	11.0	5.5	4.9	2.7
Model 5	11.1	5.2	4.8	**2.6**

Table 6.5: Dependence of different error rates on the number of included feature types for the Taba corpus. The source sentences are preprocessed using categorizations. The results were obtained using a first-order model.

corpus Taba development	total # features	SER [%]	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	226,152	10.1	7.5	5.1	6.7
+ prior	226,205	8.5	5.8	4.0	6.1
+ transition	229,014	4.5	3.0	3.4	3.6
+ pre- & suffixes	229,226	4.5	3.0	3.4	3.6
+ compound	827,861	3.6	2.5	2.6	3.3
+ capitalization	827,914	3.6	**2.5**	2.6	3.3

corpus Taba evaluation	total # features	SER [%]	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	226,152	8.6	6.8	4.8	6.4
+ prior	226,205	6.5	4.8	3.6	5.8
+ transition	229,014	4.0	2.8	3.1	4.0
+ pre- & suffixes	229,226	4.0	2.8	3.1	4.0
+ compound	827,861	3.8	2.7	2.7	3.9
+ capitalization	827,914	3.8	**2.6**	2.7	3.9

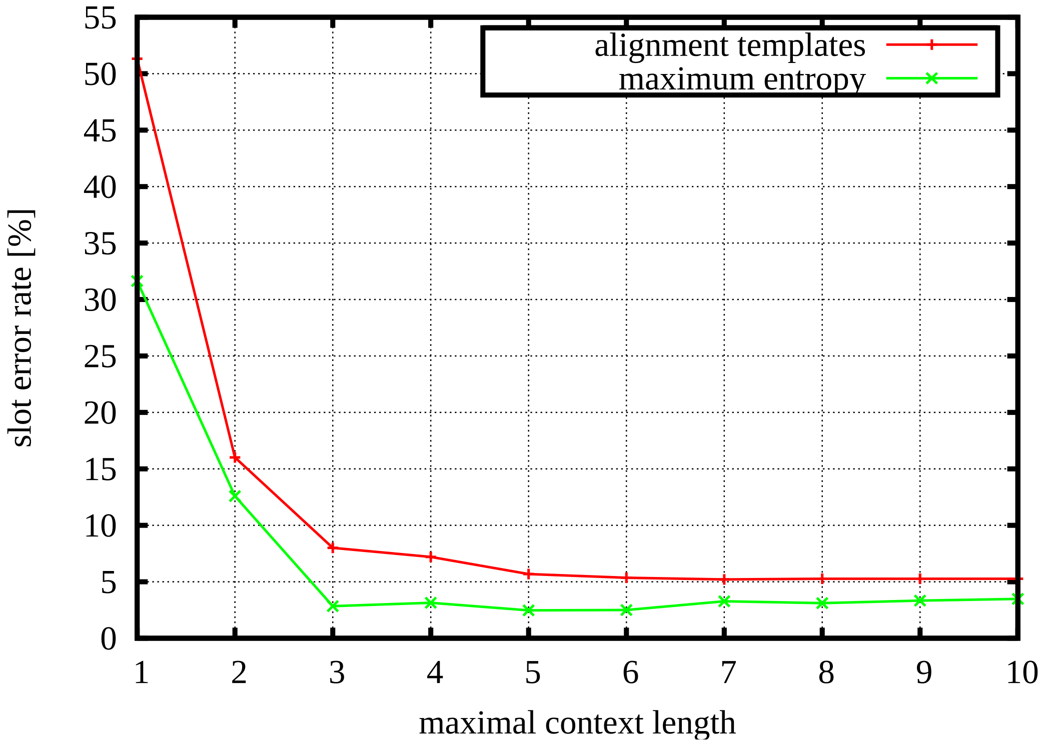


Figure 6.7: Effect of the maximal allowed context length on the slot error rate for both the alignment templates approach and the maximum entropy-based framework. The plot shows error rates for the TABA development corpus using categorizations.

Effect of Context Lengths

Both the alignment templates approach as well as the feature functions defined for the maximum entropy framework make use of local word contexts, that is, predecessor and successor words are explicitly taken into account. For the alignment templates approach, we have chosen a maximum template length of 7 for both corpora. This value was selected empirically from the TABA corpus in Figure 6.7. As can be derived from the figure, a larger value does not improve the slot error rate. The feature functions for the maximum entropy model use a shorter context of 5 consecutive words. Augmenting the context length slightly deteriorates the performance and results in an overfitted model.

First-Order versus Second-Order Model

The ME results reported in Tables 6.2 - 6.5 were obtained using a first-order model. Switching from a first-order model to a second-order model is done by using additional transition features that take the appropriate concept histories into account. This yields the results reported in Table 6.6. The results obtained with a second-order model slightly

Table 6.6: Error rates obtained from a first order model and a second order model using the maximum entropy approach with categorizations. All features reported in Table 6.3 and 6.5 were used for this experiment plus additional second order transition features.

corpus TELDIR development	Slot-ER [%]	AER [%]
1st order model	2.5	1.8
2nd order model	3.1	2.3

corpus TELDIR evaluation	Slot-ER [%]	AER [%]
1st order model	3.1	1.6
2nd order model	3.3	2.0

corpus TABA development	Slot-ER [%]	AER [%]
1st order model	2.5	3.3
2nd order model	2.6	3.2

corpus TABA evaluation	Slot-ER [%]	AER [%]
1st order model	2.6	3.9
2nd order model	2.8	4.0

deteriorate due to overfitting on the training data. Therefore, we will restrict all further ME results on a first-order model.

Effect of Number of Training Data on Error Rate

The number of training data used for the TELDIR and the TABA corpus differs roughly by a factor of 40. Since the source sentences need to be annotated manually by sequences of formal concepts in order to apply supervised learning techniques, the question is what degradation in error rate we have to expect if we reduce the number of training data used. This is important if a natural language component is to be trained for a new domain where only little data is available. Because additional training data should be collected from dialogue transactions of real users interacting with a fully functional and running dialogue system, a curve plotting the natural language understanding performance against the number of training data gives some hints on how much training data should be used for setting up a new system. Figure 6.8 shows the performance of the natural language understanding component in the course of the number of training data used. Both plots

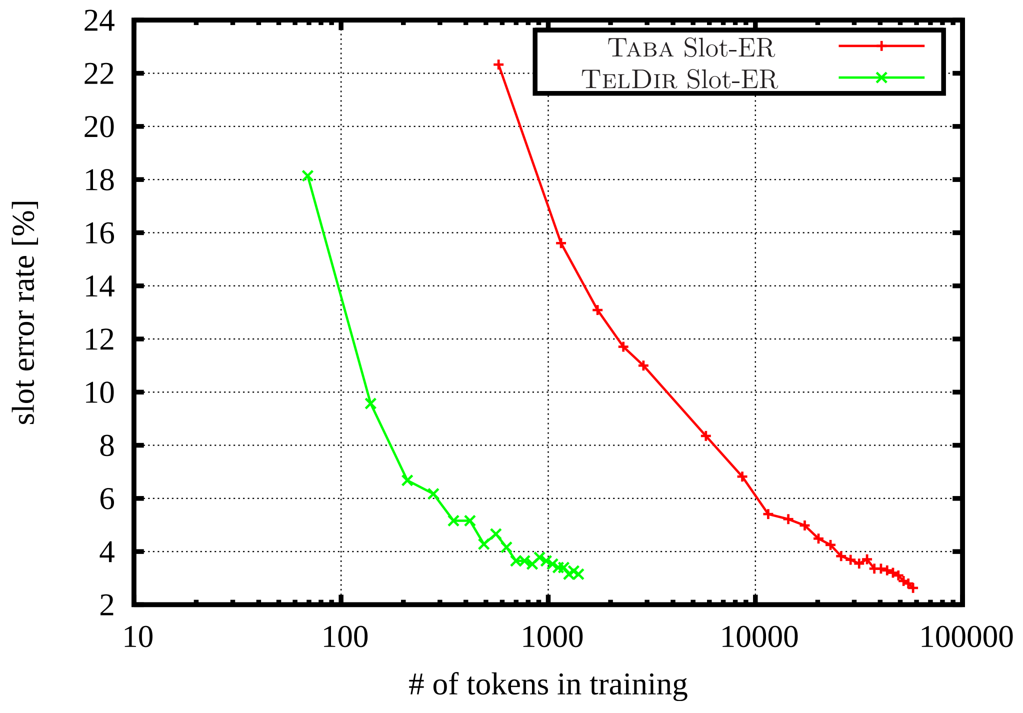


Figure 6.8: Effect of number of training data on the slot error rate for the TELDIR and TABA evaluation corpus using the maximum entropy-based approach.

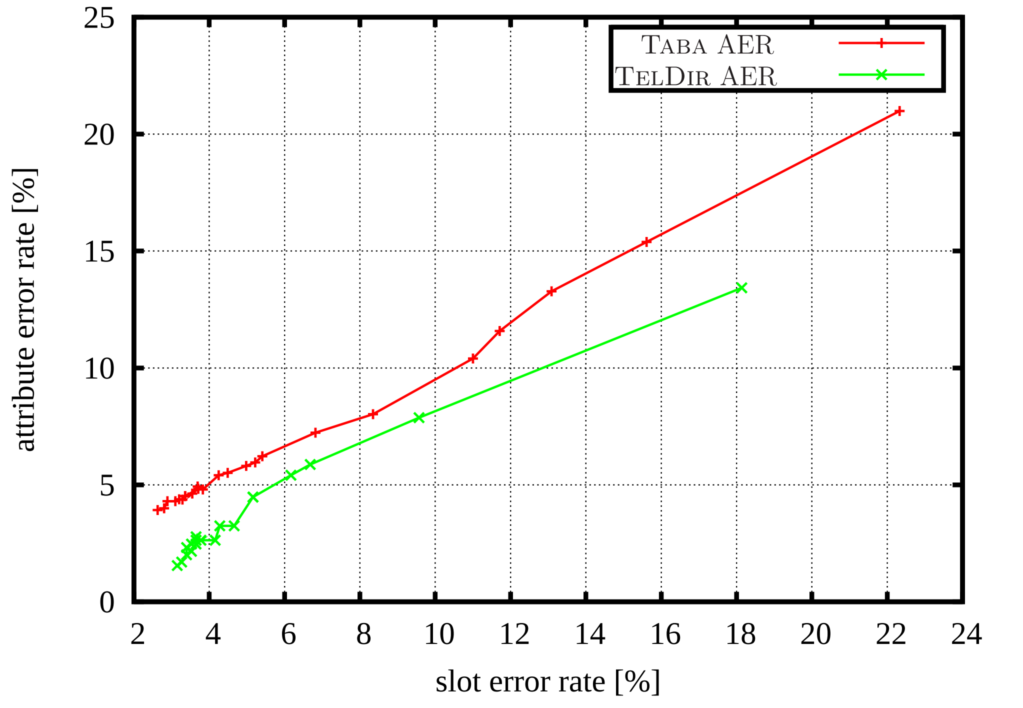


Figure 6.9: Attribute error rate in course of slot error rate for the TELDIR and TABA evaluation corpus using the maximum entropy-based approach.

use the full set of feature functions described in Section 6.5.1. In order to get a reasonable slot error rate, it is sufficient to have around 1000 training tokens for the TelDir task. The Taba task is more complex because it contains time and date expressions. Here, a slot error rate of around 6% can be achieved with 10,000 training samples. Although both tasks differ in their complexity, they almost have the same slope.

Effect of Slot Error Rate on Attribute Error Rate

Each concept is associated with a (possibly empty) set of attributes.[6] Because a wrongly assigned concept also affects the attribute error rate, we want to analyze this effect more thoroughly. Figure 6.9 plots the attribute error rate in course of the slot error rate using the maximum entropy approach for the TelDir and Taba evaluation corpus. Increasing slot error rates were obtained by successively reducing the number of training data (see Figure 6.8). For both tasks, the attribute error rate is proportional to the slot error rate. This effect results in part from the categorizations used, which yield a good generalization even if only a few training samples have been used.

6.9.5 Comparison of Alignment Templates and Maximum Entropy Approach Using Speech Input

So far, we have presented results for text-based inputs. If the input modality is speech, inputs to the natural language understanding component can contain speech recognition errors. Furthermore, spontaneous speech effects, such as disfluencies (e.g., repeated words, false starts, or repairs), hesitations, and stressed speech, can deteriorate the performance of the speech recognition module and, thus, the quality of the natural language understanding component.[7] Therefore, we want to investigate how both approaches perform if the input modality is speech. For this purpose, both models were trained on sentence pairs using reference transcriptions. Table 6.7 summarizes slot error rates and attribute error rates for both tasks. For comparison with Section 6.9.4, the first row contains error rates for text-based results.

If we compare the source-channel approach with the direct model, we see that the maximum entropy approach clearly outperforms the alignment templates approach, which is in accordance to the results obtained from text data input. However, compared to the Taba corpus the attribute error rate for the TelDir corpus deteriorates much more from erroneous input data. Apart from fewer training data, this effect results from erroneous recognitions of spelling sequences. The TelDir domain allows users to spell names, which is modeled by a spelling concept. The attribute of a spelling concept is the sequence of letters uttered by a user. If this sequence contains only one wrong letter, the concept remains most likely a spelling concept, but the attribute value is considered to be wrong,

[6] For a complete list of concepts and attributes see Appendix B.

[7] Because the text-based results presented in Section 6.9.4 use transcriptions of spontaneous speech recordings, some of these effects may also be observed on text input.

Table 6.7: Speech understanding results for the TelDir and the Taba corpus.

corpus TelDir-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.8	4.1	2.5	1.8
baseline	13.7	11.3	22.1	7.8	20.4
with incr. F-MLLR	12.4	10.6	20.6	6.8	19.1

corpus TelDir-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	6.3	4.6	3.1	1.6
baseline	14.9	15.2	26.0	11.8	23.0
with incr. F-MLLR	13.6	14.1	22.7	10.6	20.6

corpus Taba-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.2	7.9	2.5	3.3
baseline	13.0	14.5	18.9	11.8	15.2
with incr. F-MLLR	12.3	14.0	18.5	11.4	14.4

corpus Taba-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.2	6.9	2.6	3.9
baseline	12.8	13.6	16.1	11.6	13.6
with incr. F-MLLR	12.5	13.4	16.0	11.6	13.4

independent of the total number of letters that may were recognized correctly. To mitigate this effect, we can use the proper names contained in the application database as an additional knowledge source. The idea is to extract all proper names from the database and decompose them into characters sequences. Doing this we can match a recognized spelling sequence with the character sequence obtained from proper names contained in the database. If a sufficiently high number of characters matches a decomposed database entry, we assume that the remaining unmatched characters resulted from an erroneous recognition and replace the sequence by the best matching database entry. The number of sufficiently matching characters must be chosen such that on the one hand we benefit from using the database as additional knowledge source while on the other hand we allow for accepting new spelling sequences that are not contained in the database. The threshold

Table 6.8: Speech understanding results for the TelDir corpus using automatic spelling correction. The threshold for the mutual error rate between a hypothesized spelling sequence and a database entry was set to $\frac{1}{3}$. This value was optimized on the development set.

corpus TelDir-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
baseline	12.4	10.6	20.6	6.8	19.1
with spelling correction	12.4	10.6	19.1	6.8	17.3

corpus TelDir-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
baseline	13.6	14.1	22.7	10.6	20.6
with spelling correction	13.6	14.1	20.7	10.6	18.4

for the mutual error rate between a hypothesized spelling sequence and a database entry was optimized on the development set and is set to $\frac{1}{3}$. Results are listed in Table 6.8.

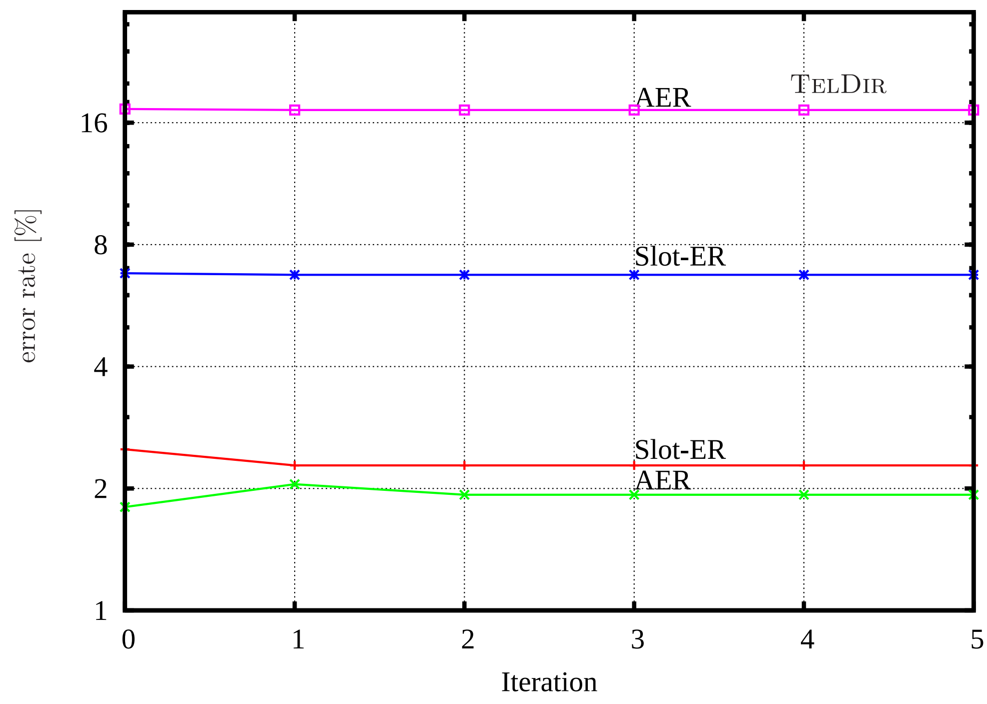
Effect of Using Maximum Entropy-based Segmentations

As described in Section 6.4.1, the alignments between words and concepts are trained using a sequence of alignment models with increasing complexity. Once this alignment has been determined it is kept fixed during training of the maximum entropy model. To obtain a refined alignment we can proceed as follows: since decoding a source sentence into a sequence of concepts with a trained maximum entropy model also provides a segmentation of the source sentence, we can use this information to extract a new alignment and train a new maximum entropy model based on this alignment. That is, by decoding our training data with a maximum entropy model for natural language understanding, we obtain a new alignment that can be used to train a new maximum entropy model.

Results are listed in Table 6.9. Except for a small gain in terms of slot error rate for the TelDir development set we mostly observe a similar performance or small degradations in terms of concept and attribute error rates. Figure 6.10 plots the discourse of the concept and attribute error rates over the number of iterations. Since these refined alignments did not yield better results, we did not further investigate this direction.

Concept Graphs and N-best Lists

Similar to the word graph generation described in Section 5.2 we can generate concept graphs as the final result of the decoding phase for the natural language understanding



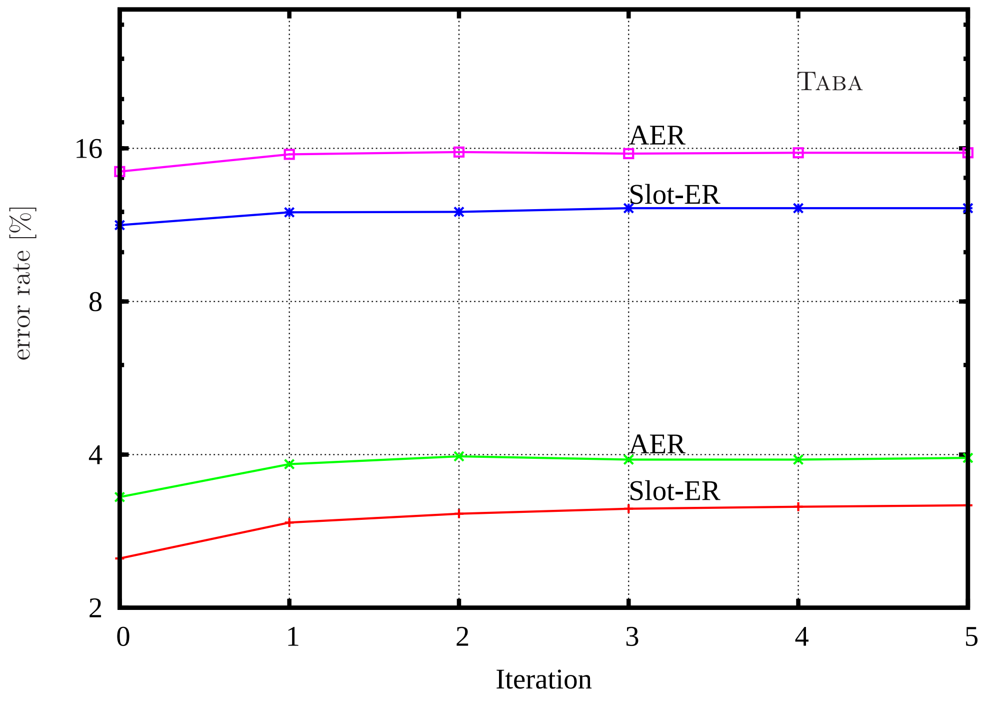


Figure 6.10: Attribute error rate and slot error rate in course of maximum entropy-based alignments for the development corpora. The points at iteration 0 denote the baseline error rates. Note that the error axis is logarithmically scaled for display purposes. The upper two curves in each graph represent results obtained from speech data, the lower two curves describe results obtained on text data.

Table 6.9: Effect of using maximum entropy-based segmentations on both the slot error rate (Slot-ER) and the attribute error rate (AER) for the TelDir and the Taba corpus. All table entries use the full set of feature functions described in Section 6.5.1 together with categorizations.

corpus	TelDir	development		evaluation	
		Slot-ER [%]	AER [%]	Slot-ER [%]	AER [%]
text	baseline	2.5	1.8	3.1	1.6
	ME-based alignment	2.3	1.9	3.4	2.0
speech	baseline	6.8	17.3	10.6	18.4
	ME-based alignment	6.7	17.2	11.0	18.4

corpus	Taba	development		evaluation	
		Slot-ER [%]	AER [%]	Slot-ER [%]	AER [%]
text	baseline	2.5	3.3	2.6	3.9
	ME-based alignment	3.2	3.9	2.7	4.1
speech	baseline	11.4	14.4	11.6	13.4
	ME-based alignment	12.2	15.7	11.3	13.5

Table 6.10: Oracle error rates for the spoken language understanding component. Concept graphs are produced by applying the ME approach on the first best results from the ASR component. The 100-best lists are directly extracted from the concept graphs whereas the 100-best* lists are obtained by merging 10-best concept lists derived from 10 concept decodings. Each decoding uses one candidate from a 10-best ASR list as input.

corpus TelDir-SLU	baseline	graph	100-best	100-best*
	Slot-ER [%]	Slot-ER [%]	Slot-ER [%]	Slot-ER [%]
development	6.8	1.9	2.4	1.7
evaluation	10.6	2.6	2.8	2.3

corpus Taba-SLU	baseline	graph	100-best	100-best*
	Slot-ER [%]	Slot-ER [%]	Slot-ER [%]	Slot-ER [%]
development	11.4	2.5	3.9	3.1
evaluation	11.6	2.7	4.0	3.5

component. Again, each node of the graph represents the language model history of its outgoing edges. For the maximum entropy model approach, an edge of the graph represents a concept. With an A^* search we can extract N-best lists from concept graphs, which are then used for computing confidence measures and for combining speech recognition with natural language understanding. Because the first-best speech recognition sentence hypothesis may contain recognition errors, we extract a 10-best list from the word graphs of the speech recognition decoding and compute a 10-best concept list for each hypothesis contained in the ASR list, resulting in a 100-best concept list for each utterance. Table 6.10 shows graph and N-best list error rates for the natural language understanding component using both text and speech as input modality.

Confidence Measures for Spoken Language Understanding

According to Section 6.7, we compute confidence measures on N-best lists. In contrast to the work presented in [Ueffing 06] our machine translation task is always monotone. Furthermore, we do not allow empty words. Therefore, one might expect that the concept graph-based approach will always outperform the N-best list approach. Table 6.11 lists confidence error rates computed on a 100-best list for the spoken language understanding component. The optimal threshold for the evaluation set was adjusted on the development set beforehand. The performance of the NLU confidence measures does not reach the performance of their speech counterparts. However, the results clearly show that the N-best list approach together with a Levenshtein alignment clearly outperforms the graph-based approach. A possible explanation for this is that a concept has in general much fewer word positions then a word hypothesis has time frames. Because of that, the notion of overlapping concept edges within a concept graph is less distinctive than the overlapping word hypothesis edges that we observe in a word graph. As a result the Levenshtein-aligned concept hypotheses derived from N-best lists perform much better.

6.9.6 Combining Speech Recognition and Language Understanding

To combine speech recognition with language understanding we proceed as follows: we first generate ASR word graphs for all utterances and compute confidence values for each hypothesis in a word graph. We then extract N-best lists from the word graphs and process them each N-best entry with the ME approach, thus enriching the N-best lists with the additional features described in Section 6.8.3. The enriched N-best lists are then inserted into a repository on which the minimum error rate training algorithm is applied.

A problem with this approach is that the ASR decoder cannot produce better results in subsequent iterations because the new features are not integrated into the ASR decoding process, but are added after the N-best lists have been extracted. Since the decoder cannot use the new knowledge sources directly, it cannot produce new entries for the repository. Therefore, we have to make sure that word graphs and N-best lists contain a sufficiently large number of hypotheses after the first iteration. According to Figure 5.4 and Table 6.10, a 10×10-best list provides enough sentence alternatives from which we can select promising candidates in order to lower the error rate. Furthermore, selecting a candidate out of a small N-best list might be more realistic than striving to find the oracle-best candidate in a huge N-best list. The motivation behind this is that the additional features should rather aim at reranking candidates whose scores are very close

Table 6.11: Comparison of concept posterior probability-based confidence measures determined on concept graphs and N-best lists. Here, CER denotes the confidence error rate. The 100-best list is directly extracted from the concept graph whereas the 100-best* list accumulates concept 10-best lists derived from concept decodings computed from the 10 candidates of the ASR list.

corpus TelDir-SLU	baseline Slot-ER [%]	baseline CER [%]	graph CER [%]	100-best CER [%]	100-best* CER [%]
development	6.8	4.8	4.6	4.1	4.7
evaluation	10.6	8.4	7.5	7.1	7.5

corpus Taba-SLU	baseline Slot-ER [%]	baseline CER [%]	graph CER [%]	100-best CER [%]	100-best* CER [%]
development	11.4	8.7	8.7	8.4	8.7
evaluation	11.6	8.6	8.6	8.4	8.6

to each other, than trying to move a very low-ranked candidate to the first position. The final repository contains a 10 × 10 N-best list for each utterance where each N-best entry consists of three streams together with the associated error counts: the word stream obtained from ASR decoding, the concept stream obtained from NLU decoding, and the attribute stream obtained from the attribute extraction. The N-best lists do not contain duplicates. The hypothesized word-concept segmentation is kept for the concept stream in order to simplify the attribute extraction.

Minimizing the Word Error Rate

We first investigate whether NLU features can help reducing ASR errors. Table 6.12 presents results for combining ASR with NLU. The feature weights were optimized on the development set according to the algorithm described in Sec. 6.8.2. The objective function used is minimizing the WER. The first line in each table is the baseline WER obtained under realtime recognition constraints, which is equal to the ASR performance reported in Tables 5.6 and 5.8. To measure the effect of newly added features, we first optimize the weights for the ASR baseline features on the development N-best repository and apply those weights on the evaluation N-best repository. The ASR baseline features comprise the acoustic model, the language model, and the word penalty. For the TelDir task this gives a small improvement of 0.2% on the development set, which does not generalize to the evaluation set. For the Taba task, the baseline optimization reduces the WER much more on both the development and the evaluation corpus. Due to realtime constraints we cannot use the optimized feature weights directly during speech decoding since this would require larger beam settings. Therefore, applying the optimized baseline feature weights on the N-best repository is similar to a second-pass decoding, where we rescore

Table 6.12: Combination of speech recognition and natural language understanding for the TelDir and Taba task. The feature weights were optimized on the development set. The objective function used is minimizing the word error rate.

corpus TelDir-Speech Features	development WER [%]	evaluation WER [%]
Baseline (acu + lm + word-penalty)	12.4	13.6
Baseline (optimized on Repository)	12.2	13.8
+ concept score	11.7	12.9
+ spellingCorrection	10.9	12.3
+ concept penalty	**10.7**	**12.4**
+ asr posterior	10.9	12.3
+ asr confidence	10.9	12.3
+ concept LM	11.0	13.1

corpus Taba-Speech Features	development WER [%]	evaluation WER [%]
Baseline (acu + lm + word penalty)	12.3	12.5
Baseline (optimized on Repository)	11.8	11.7
+ concept score	11.7	11.4
+ concept penalty	11.7	11.4
+ concept LM	**11.6**	**11.5**

the candidates from the first pass with optimized weights.

The first added feature is the NLU concept score. With this feature, the WER is reduced by 0.6% on the TelDir development set and by 0.5% on the evaluation set. The improvements on the Taba task are smaller but consistent.

A special feature that we can use for the TelDir corpus is the spelling correction feature. We already showed in Sec. 6.9.5 that this feature reduces the AER by comparing the actual attribute values with the values stored in the application database. We can use the same knowledge source here and use the Levenshtein distance of a hypothesized spelling sequence with respect to the closest database entry as additional feature. Because only a fraction of the spelled proper names is covered by the database, we leave it to the minimum error rate training framework to find a tradeoff between covered and uncovered proper names.

Minimizing the Slot Error Rate

In the previous section, the objective was to minimize the word error rate by adding NLU-related knowledge sources. Now, we investigate if we can reduce the Slot-ER by adding ASR-related features to the NLU features. Table 6.13 presents results for combining ASR

Table 6.13: Combination of speech recognition and natural language understanding for the TelDir and Taba task. The feature weights were optimized on the development set. The objective function used is minimizing the slot error rate.

corpus TelDir-Speech Features	development Slot-ER [%]	evaluation Slot-ER [%]
Baseline (conceptScore)	6.8	10.6
Baseline (1st best on Repository)	9.9	13.9
+ acoustic model	8.3	11.7
+ language model	6.3	10.8
+ word penalty	**6.3**	**10.1**

corpus Taba-Speech Features	development Slot-ER [%]	evaluation Slot-ER [%]
Baseline (conceptScore)	11.4	11.6
Baseline (1st best on Repository)	17.1	17.7
+ acoustic model	14.9	15.6
+ language model	11.0	11.3
+ word penalty	**10.6**	**10.7**

with NLU when minimizing the Slot-ER. The baseline Slot-ER is equal to the Slot-ER reported in Table 6.10. Note that this is a single feature that employs all the feature functions described in Sec. 6.5.1. Therefore, no baseline optimization on the N-best repository is necessary. However, the first-best result determined on the repository might actually be worse compared to the baseline result given that we use only the concept score as a single feature. The reason is that the baseline result is computed from the first best ASR sentence hypothesis, for which the ASR features, such as the acoustic model weight and the language model weight, are already fixed. If we take only the concept score into account and rescore the repository, we decouple the concept feature function from the ASR knowledge sources and allow choosing much shorter sentence hypotheses that might have more deletion errors but produce lower concept scores. As soon as we add more ASR feature functions, we observe a reduction in terms of Slot-ERs. Optimizing the weights for all standard ASR feature functions, that is, adding acoustic scores, language model scores, and word penalties, significantly reduces the Slot-ER.

Minimizing Word and Slot Error Rate simultaneously

By combining the error counts for ASR and NLU and by using all features defined previously, we can minimize both WER and Slot-ER simultaneously. The unweighted error counts are combined by summation. Table 6.14 lists results obtained when minimizing the combined error counts. For comparison reasons, the table also contains the results

Table 6.14: Combination of speech recognition and natural language understanding. The objective function used is minimizing the slot and the word error rate simultaneously. The approach uses all features defined previously. For comparison reasons, the results obtained with optimizing the word error rate or slot error rate independently are included as well.

corpus TelDir-Speech	development		evaluation	
Features	WER [%]	Slot-ER [%]	WER [%]	Slot-ER [%]
Baseline	12.4	6.8	13.6	10.6
optimized for WER + Slot-ER	10.9	6.7	12.3	9.6
optimized for WER	10.7	–	12.4	–
optimized for Slot-ER	–	6.3	–	10.1

corpus Taba-Speech	development		evaluation	
Features	WER [%]	Slot-ER [%]	WER [%]	Slot-ER [%]
Baseline	12.3	11.4	12.5	11.6
optimized for WER + Slot-ER	11.6	10.3	11.4	10.2
optimized for WER	11.6	–	11.5	–
optimized for SlotER	–	10.6	–	10.7

from the previous tables. An interesting effect is that the slot error rate can be further reduced for both tasks if we minimize the joint error counts. Although we would expect that optimizing for the slot error rate explicitly should give lower slot error rates, this indicates that the error function to be optimized is not convex. Furthermore, the minimum error training algorithm used contains a geady step at the end of the outermost `for` loop where we select the feature weight with the biggest gain that shall be updated. Overall, combining the error counts helps to further improve the system and generalizes well on unseen test data.

6.10 Summary

In this chapter, we have proposed two different approaches to natural language understanding based on statistical machine translation. The main difference between both approaches is that the ME framework directly models the posterior probability in a discriminative manner whereas the source channel-based approach applies Bayes' theorem resulting in two distributions: the translation probability and the language model probability. We have evaluated both approaches on two language understanding tasks, which are derived from different domains, and have shown that the ME approach clearly outperforms the source channel-based approach within these settings.

One of the advantages of the ME approach results from the property that the ME framework directly models the posterior probability and allows for integrating structural information by using appropriate feature functions. Furthermore, the ME approach is consistent with the features observed on the training data, but otherwise makes the fewest possible assumptions about the distribution. Since the optimization criterion is convex, there is only a single optimum and no convergence problems occur.

We also analyzed the performance of both approaches when speech is used as input modality and investigated several effects, for example, the effect of categorizations, different context lengths, and the relation between slot error rates and attribute error rates. Furthermore, we defined confidence measures for natural language understanding that are inspired by posterior probability-based approaches used in speech recognition.

Finally, we showed how ASR and NLU-based knowledge sources can be log-linearly combined such that the overall error criterion is minimized. This allows for a tighter coupling between speech recognition and natural language understanding. We have employed the minimum error rate training framework in order to determine an optimal set of feature weights and showed that the WER can be reduced using NLU-based feature functions and that, vice-versa, the Slot-ER can be reduced using ASR-based feature functions.

Chapter 7

Dialogue Management

Ideally, dialogue managers should be application-independent in order to avoid a redesign of the dialogue system when changing the application domain. In this chapter, we propose a new dialogue management system that employs trees as fundamental data structure. Based on several feature functions, all operations of the dialogue manager, such as collecting information and deciding the next dialogue action are formulated as cost functions that operate directly on nodes, paths, and trees.

7.1 Construction of a Domain-Independent Dialogue Manager

If several tasks and domains are to be treated by a single dialogue system without replacing or rewriting parts of the system, the need for an application-independent dialogue manager arises. In order to separate the dialogue manager from a concrete application, we have to provide the task knowledge in the form of parameterizable data structures and distill those steps that are common for many domains. These steps include:

- information request,
- information collection and evaluation,
- error detection and error handling,
- ambiguity resolution, and
- information retrieval.

Parameterizable data structures must be derived from the knowledge of each domain, and all other operations, such as integrating concept-attribute pairs derived from user utterances, ambiguity detection and resolution as well as determining the subsequent dialogue action should be based on this structure. Here, the fundamental data structures used are trees. Their hierarchical structure allows for a natural distinction between different dialogue goals. In the next sections, we will describe this data structure and its employment in a dialogue management task.

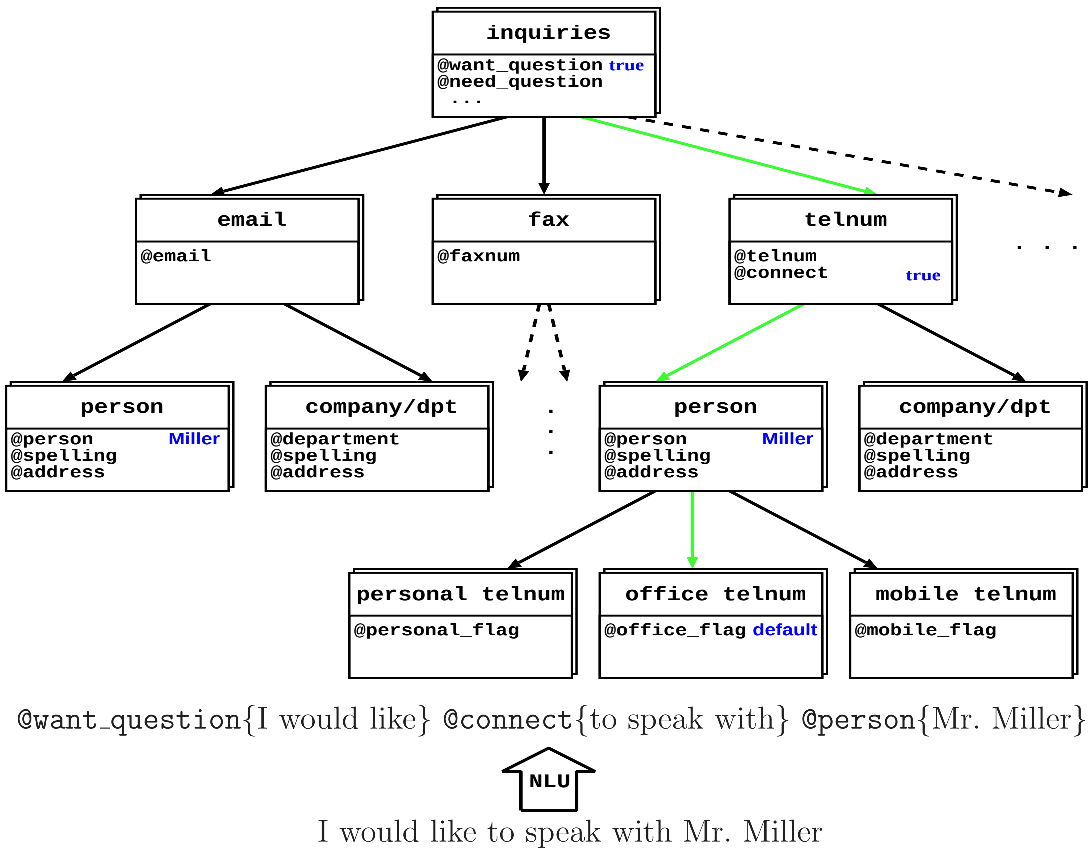


Figure 7.1: Tree-based knowledge representation for the telephone directory assistance task TelDir. The sentence "I would like to speak with Mr. Miller" is transformed into a concept representation using statistical machine translation. After that, each concept together with its associated attributes is incorporated into the corresponding tree nodes.

7.1.1 Tree-based Representations

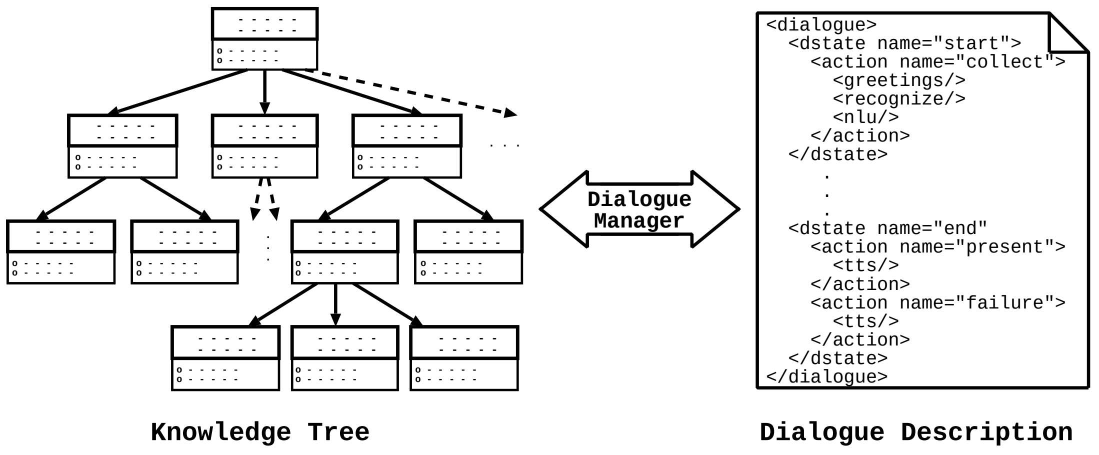
In order to obtain domain-independent representations, we use trees as the fundamental data structure. An example is depicted in Figure 7.1. The tree is a knowledge representation for the telephone directory assistance task TelDir. Users can ask for information about telephone numbers, email addresses, and fax numbers of persons as well as companies. The tree is an overspecification of the given domain. The upper part of each tree node describes the part of the dialogue that is processed by the corresponding subtree. The lower part of each node consists of a list of concepts that are associated with that specific node.[1] Each path of the tree describes a separate dialogue goal. As depicted in the figure, the root node's name is "inquiries", and the associated list consists of concepts that are related to different kinds of request verbalizations. The successor nodes are

[1] For presentation reasons, only the concept names and attribute values are included in the figure.

specifications of the corresponding parent node. For the given example, the specifications are requests for email addresses, fax numbers, and phone numbers, respectively. For a given utterance, the sequence of concepts is produced using the methods described in Chapter 6. The machine translation approach translates the input sentence into a sequence of concepts and provides an alignment between words and concepts. In the example, the word sequence “I would like” is aligned to the concept `@want_question`, the word sequence “to speak with” is aligned to the concept `@connect`, and so on. These alignments are used in order to extract the attributes for each concept out of the aligned words. Each concept together with its attributes is then incorporated into all nodes of the tree in which the concept’s name occur. In Figure 7.1, the concept `@person`{Mr. Miller} is incorporated into two nodes. A tree being instantiated with the attribute values derived from user input is called an *instance tree.*

7.1.2 Basic Dialogue Framework

The basic framework of the dialogue system is depicted in Figure 7.2. The XML-based dialogue description consists of different dialogue states, subdividing a dialogue into smaller sections. Each dialogue state may again be subdivided into several action states. During a dialogue transaction, the dialogue manager incorporates the knowledge from user input into the knowledge tree. If the subsequent dialogue state is not explicitly given by the dialogue description, the dialogue manager will determine the next state-action pair by analyzing the tree’s information content. Depending on the chosen state-action



```
<dialogue>
  <dstate name="start">
    <action name="collect">
      <greetings/>
      <recognize/>
      <nlu/>
    </action>
  </dstate>
      .
      .
      .
  <dstate name="end"
    <action name="present">
      <tts/>
    </action>
    <action name="failure">
      <tts/>
    </action>
  </dstate>
</dialogue>
```

Figure 7.2: Basic structure of the dialogue system: The dialogue manager reads in an XML-based dialogue description and a task specific knowledge tree. During a dialogue transaction, the dialogue manager incorporates information gained from user inputs into the knowledge tree and determines the next dialogue state-action by analyzing the tree’s information content.

pair, the dialogue manager will execute only those dialogue commands that are specified in the dialogue description for the chosen pair.

7.1.3 Dialogue Course Management

During a dialogue transaction, instance trees are built from the original knowledge tree. Concept-attribute pairs that have been retrieved from user input are incorporated into these instance trees. If there is only one path from the root to a leaf satisfying the property that all necessary concept-attribute pairs from nodes along that path are filled (illustrated by the green path in Figure 7.1), the user's request can be answered by the dialogue system. If more than one completely filled path exists, the data retrieved from the user is ambiguous and the user is asked to provide more information so that the dialogue system can further reduce the set of candidates.[2] If there is no path from the root to a leaf such that all concepts along that path have attribute values, some of the nodes are still empty. In this case the system must ask for additional information in order to fill the remaining nodes. In general, there are several possibilities to continue a dialogue. Therefore, we introduce a cost function that computes a score for all nodes, paths, and trees. Starting from the root node, the dialogue manager chooses that node, whose corresponding subtree has minimal cost. Besides choosing the subsequent dialogue state, the dialogue manager also chooses the next dialogue action, which can be verifying information stored in a node in case of low confidence of the recognized word sequence, resolving ambiguities, asking for additional information, or answering the user's request. The decision rule for choosing the next dialogue state-action pair is given in the following equation:

$$(\hat{s}_{t+1}, \hat{a}_t) = \underset{s_{t+1}, a_t}{\operatorname{argmax}} \left\{ pr(s_{t+1}, a_t | s_t, x_1^T) \right\} \tag{7.1}$$

$$\begin{aligned} pr(s_{t+1}, a_t | s_t, x_1^T) &= \sum_{c_1^M} pr(s_{t+1}, a_t, c_1^M | s_t, x_1^T) \\ &\cong \max_{c_1^M} \left\{ \underbrace{p(a_t|s_t)}_{\text{DM}} \cdot \underbrace{p(s_{t+1}|s_t, a_t, c_1^M)}_{\text{Task Model}} \cdot p(c_1^M | x_1^T, s_t) \right\} . \end{aligned} \tag{7.2}$$

Similar to the problem of natural language understanding described in Section 6.1, Equations 7.1 and 7.2 induce two different approaches: we could either try to model the posterior probability of the joint event (s_{t+1}, a_t) directly or decompose the joint event into two probability distributions, a task model that describes the state transitions and a dialogue management model that describes the allowed dialogue actions. The first approach requires an $S \times A \times S$ lookup table. If a dialogue state explicitly encodes all

[2] Of course, this simple strategy is not able to detect other kinds of ambiguities that may occur during a dialogue session. For example, ambiguities caused by homophones are not covered by this strategy. Additional methods for handling ambiguities are described in Section 7.2.

the information a dialogue system has collected so far, the corresponding lookup table will become huge, depending on the number of different states and actions. In the second approach, we have to model and estimate two different probability distributions: a task model that might use a reduced $S \times A \times S$ lookup table and the much smaller $S \times A$ table for the dialogue management model. At this point it is not entirely clear which approach would lead to a more robust dialogue system. Another problem is how to collect a reasonable number of training data in order to estimate the models' parameters. Especially when building a dialogue system for a new domain, getting samples of dialogue transactions is often hard if not impossible. Training samples should ideally be collected during actual human-machine interactions, which already requires the implementation of a reasonable strategy. Therefore, we discard the probability function in Equation 7.1 and use a decision tree for determining the next state-action pair. In order to reduce the number of states, we introduce meta-states that describe the tree's information state on a more coarse level. For example, if the dialogue system must decide whether it can answer the user's request it is sufficient to know whether a fully informed path from the root to a leave exists, which is a binary information. The actual values of the nodes can be ignored.

7.2 Feature Functions

For the cost function, different features and knowledge sources can be taken into account. We will use the following node-specific features.

7.2.1 Word Confidence Measures

The confidence measure described in Section 5.3 is probabilistic and exploits only information that is contained in the output word graph of the speech recognition system. After computing the confidence, each recognized word is tagged as either correct or wrong, depending on whether its confidence exceeds a given threshold τ. Denote $\mathcal{W}(c)$ the set of words that are assigned to a concept c. Then, the first feature for node n is defined as follows:

$$v_1(n) := \min_{c \in \mathcal{C}(n)} \left\{ \left(\prod_{w \in \mathcal{W}(c)} \tilde{p}(w) \right)^{\frac{1}{|\mathcal{W}(c)|}} \right\} , \tag{7.3}$$

where $\mathcal{C}(n)$ is the set of concepts for node n and $\mathcal{W}(c)$ is the multiset of words that are aligned to concept c. The feature function $v_1(n)$ computes the minimal geometric mean of the word confidence values for concepts c in node n.

7.2.2 Importance and Degree of Concept and Attribute Instances

The importance of a concept c and of an attribute a depends on the given domain. For the telephone directory example, the last name of a person is more important than its first name. Therefore, we introduce a ranking r describing the relevance of concepts and attributes. Consequently, a person's last name is *mandatory* ($r(a) = 1$) whereas the first name is *supplementary* ($r(a) = 0$). The ranking of concepts and attributes is taken into account by summing over all concepts and attributes, respectively, for which the associated attribute value is required but has not yet been instantiated by user input. For an attribute a of a concept c, we compute:

$$\begin{aligned} f(a) &\longmapsto \begin{cases} 1 & \text{if attribute } a \text{ is assigned a value} \\ 0 & \text{otherwise.} \end{cases} \\ v_2(n) &\coloneqq \sum_{c \in \mathcal{C}(n)} \sum_{a \in \mathcal{A}(c)} \delta(r(a), 1) \cdot \delta(f(a), 0) \ , \end{aligned} \tag{7.4}$$

where $\mathcal{C}(n)$ is the set of concepts of a node n and $\mathcal{A}(c)$ is the set of attributes belonging to a concept c. $v_2(n)$ counts the number of mandatory attributes for node n that still have no value. Here, $\delta(\cdot, \cdot)$ denotes the Kronecker function. The importance $r(c)$ of a concept c can be derived from the maximum ranking of its related attributes. However, for some nodes, it is more convenient to fix a concept's rating independently of the related attributes. Therefore, we compute the third feature for node n as follows:

$$\begin{aligned} g(c) &\longmapsto \begin{cases} 1 & \text{if concept } c \text{ is sufficiently instantiated} \\ 0 & \text{otherwise.} \end{cases} \\ v_3(n) &\coloneqq \sum_{c \in \mathcal{C}(n)} \delta(r(c), 1) \cdot \delta(g(c), 0) \ . \end{aligned} \tag{7.5}$$

Again, $\delta(\cdot, \cdot)$ denotes the Kronecker function. A concept c is sufficiently instantiated if all its related attributes with ranking $r(a) = 1$ have already been assigned values, that is, $|\{a \in \mathcal{A}(c) | r(a) = 1, f(a) = 1\}| = \sum_{a \in \mathcal{A}(c)} r(a)$. The feature function $v_3(n)$ counts the number of concepts for node n that are insufficiently instantiated.

7.2.3 Degree of Ambiguity

Ambiguities within a spoken dialogue system can result from several sources: misrecognized utterances, errors during the natural language understanding step, or ambiguous user language [Ammicht & Potamianos$^+$ 01]. In the context of a telephone directory assistance, ambiguities may also occur from proper names that can be used as both first names and last names. Another source of ambiguities are homophones, that is, proper names that have the same pronunciation but different spellings can result in additional ambiguities that must be detected and resolved by the dialogue system. In the latter case, the speech recognition system simply constructs a list of all possible sentences with different homophone names and leaves it to the dialogue manager to resolve this kind of ambiguity. The dialogue manager constructs an instance tree for every sentence hypothesis that is delivered by the recognition system. If an ambiguity has been detected in node n, this is annotated in the cost vector:

$$\begin{aligned}\text{amb}(n) &\longmapsto \begin{cases} 1 & \text{if node } n \text{ has ambiguous information} \\ 0 & \text{otherwise.} \end{cases} \\ v_4(n) &\coloneqq \text{amb}(n)\ .\end{aligned} \tag{7.6}$$

The feature function $v_4(n)$ keeps a record of detected ambiguities for node n. If the ambiguity is resolved by the dialogue system, the amb(n) flag is reset to 0.

7.2.4 Contradictory Information

A node n contains contradictory information if an already instantiated attribute is overwritten by a new value that is inconsistent with the old one. In this case, the following feature is set to 1:

$$\begin{aligned}\text{cnt}(n) &\longmapsto \begin{cases} 1 & \text{if node } n \text{ has contradictory information} \\ 0 & \text{otherwise.} \end{cases} \\ v_5(n) &\coloneqq \text{cnt}(n)\ .\end{aligned} \tag{7.7}$$

The feature function $v_5(n)$ keeps a record of nodes that contain contradictory information. Note that the number of SQL results for a database query of a node containing contradictory information is always 0.

7.2.5 Number of SQL Results

If the number of database entries returned from a database query is too large, the user should refine his request. If no database entry has been returned, the answer to the user's request is not covered by the database or the request should be less restrictive. The number of SQL results is taken as an additional feature for the cost vector. Let $t(q)$ be a table returned by a database query q. Then, feature $v_6(n)$ is defined as follows:

$$v_6(n) \coloneqq |t(q_n)|\ , \tag{7.8}$$

where $|\cdot|$ describes the number of table entries. Thus, the feature function $v_6(n)$ simply counts the number of returned table entries.

7.2.6 Verification of Information

Since automatic speech recognition is error-prone, it seems reasonable to allow for verification questions in order to verify the attribute values of some concepts, particularly, if the confidence of the aligned words is low. Verification of node information is taken as an additional feature for the cost vector:

$$\begin{aligned}\text{verf}(n) &\longmapsto \begin{cases} 1 & \text{if information in } n \text{ has been verified} \\ 0 & \text{otherwise.} \end{cases} \\ v_7(n) &\coloneqq \text{verf}(n)\ .\end{aligned} \tag{7.9}$$

The feature function $v_7(n)$ keeps a record of whether the information stored in node n has been verified by the system.

7.3 Computing Dialogue Costs

For each input sentence, a semantic analysis is performed. The concept-attribute pairs are extracted and inserted into temporary arrays for all tree nodes that are associated with these pairs. Temporary arrays are used in order to detect contradictory information. Dialogue costs are then computed on different levels: local node costs, path costs, and tree costs.

7.3.1 Node Costs

For a node n, the computation of node costs is done by applying the feature functions described in Section 7.2. For a tree t, this yields node-specific cost vectors $v_t(n)$ consisting of feature values computed from the user input and the available knowledge sources:

$$v_t(n) = \begin{pmatrix} v_1(n) \\ v_2(n) \\ \vdots \\ v_7(n) \end{pmatrix}, \quad n \in \text{nodes}(t) \ . \tag{7.10}$$

Here, nodes(t) is the set of nodes for an instance tree t. The node costs are local costs and are computed independently from each other. Although not expressed in the formulae, some cost function values depend on the input modality used. For example, when using text input via keyboard in contrast to speech input, the confidence is always set to 1.0, and the verification function is set to 1, accordingly.

7.3.2 Path Costs

For many applications, a knowledge tree has only a moderate number of leaves. Since a tree has as many paths as leaves, there is no need to combine the costs of different paths within all parent nodes. Instead, all paths of a tree are treated separately. For computing the costs of a path π for an instance tree t, we simply add the node costs for all nodes along this path π.

$$v_t(\pi) = \bigoplus_{n \in \pi(t)} v_t(n) \ . \tag{7.11}$$

The combination function $\oplus$ for combining node-specific cost vectors is defined as follows: most of the feature values are added component-wise, except the confidence feature and the feature that computes the number of SQL results. Here, the confidence of a path is defined as the minimum of the confidence values of all its nodes. For the SQL feature, we internally expand the SQL query by additional “where” constraints that are given by the information stored in the nodes along the path. At the end of the computation, each path is assigned a cost vector corresponding to the costs that arise for continuing the dialogue along that path.

7.3.3 Tree Costs

At the end of the path costs computation, each path is assigned a cost vector corresponding to the costs that arise for continuing the dialogue along that path. If there are different possible paths that may conclude the dialogue, the dialogue manager will choose the optimal scored path in order to proceed the dialogue. This requires a comparison function, for which we use a decision tree. The decision tree determines the optimal scored path of an instance tree. By equating a tree’s costs with its optimal scored path, we can use the decision tree also for determining the optimal scored instance tree.

7.4 Selection of Dialogue State-Action Pairs

There are different dialogue actions that can be chosen by the dialogue manager in order to continue a dialogue. Typical dialogue actions are collecting information and presenting database query results to the user. The choice of the subsequent dialogue action depends on the costs that have been computed for each path of a tree. Since the best-scored path as well as the subsequent dialogue action are determined by decision trees, the structure of the decision trees has an immediate influence on the dialogue strategy. A partial decision tree for choosing the subsequent dialogue action is shown in Figure 7.3. If the confidence of some information is lower than a given threshold, the information stored in the node with the lowest confidence is explicitly verified by additional system requests. If the best-

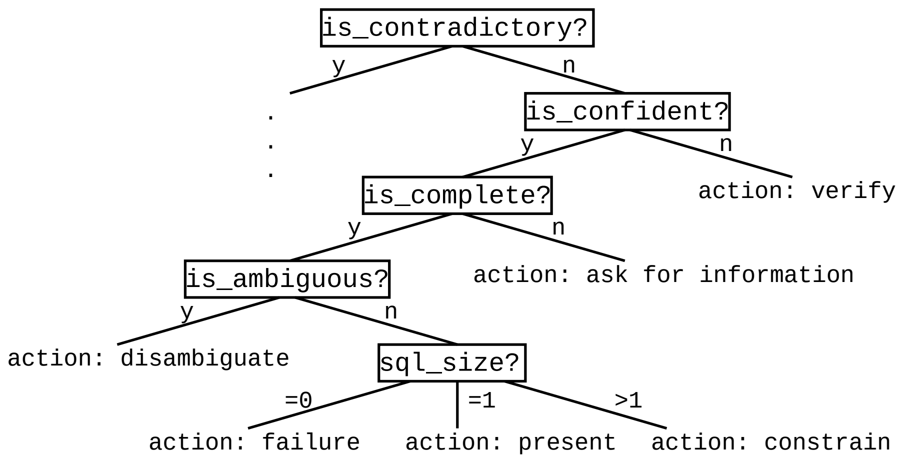


Figure 7.3: Partial decision tree for determining the subsequent dialogue action.

Table 7.1: Recognition results of the speech recognition module integrated into a former version of the spoken dialogue system. This version was used during the field tests where humans interacted with a fully functional spoken dialogue system.

corpus TelDir-Speech	WER [%]	CER [%] baseline	CER [%]
development	15.7	14.3	10.5
evaluation	16.4	14.1	9.4

scored path includes ambiguous information (which is marked by the ambiguity function, cf. Equation 7.6) that cannot be resolved by the system, the user is asked by the system to solve this ambiguity. If the best-scored path is incomplete because at least one node is empty, the system asks for additional information in order to complete this path. If there is a complete path with a moderate number of SQL answers, the system replies to the user's request.

7.5 Results

Experiments were performed using a spoken dialogue system developed for the telephone directory assistance task TelDir. The feature functions described in Section 7.2 were implemented and integrated into a former version of the dialogue system. The evaluation was done in a field test where humans interacted with a fully functional system. The speech recognition module was trained on a subset of the TelDir corpus excluding spelling units. Furthermore, no speaker-adaptive techniques or normalization methods were applied. Table 7.1 lists recognition results for the development and evaluation test set of the speech recognition module used for this field test. Word posterior probabilities were employed as confidence measures and were computed on bigram-decoded word graphs. All confidence measure-related free parameters, that is, the acoustic scaling factor, the language model scaling factor, and the tagging threshold, were optimized on the development test set beforehand. Table 7.1 summarizes the corresponding confidence error rates. The detection error tradeoff curve is depicted in Figure 7.4.

The natural language understanding component was based on statistical machine translation using the alignment templates approach [Macherey & Och$^+$ 01]. The underlying database contained approximately 500 German and foreign proper names as well as personal-related data, including office phone numbers, home phone numbers, the position of a person in a company, and so on. For evaluating the performance of the dialog management module, a set of 40 dialogue transactions of humans interacting with the dialogue system was recorded and analyzed. These dialog transactions were judged by a human who decided whether the dialog manager chose the correct or wrong action depending on the user's query. In 88% of all cases, the dialog manager was able to choose the correct subsequent action and finished the dialog successfully. For text input, the

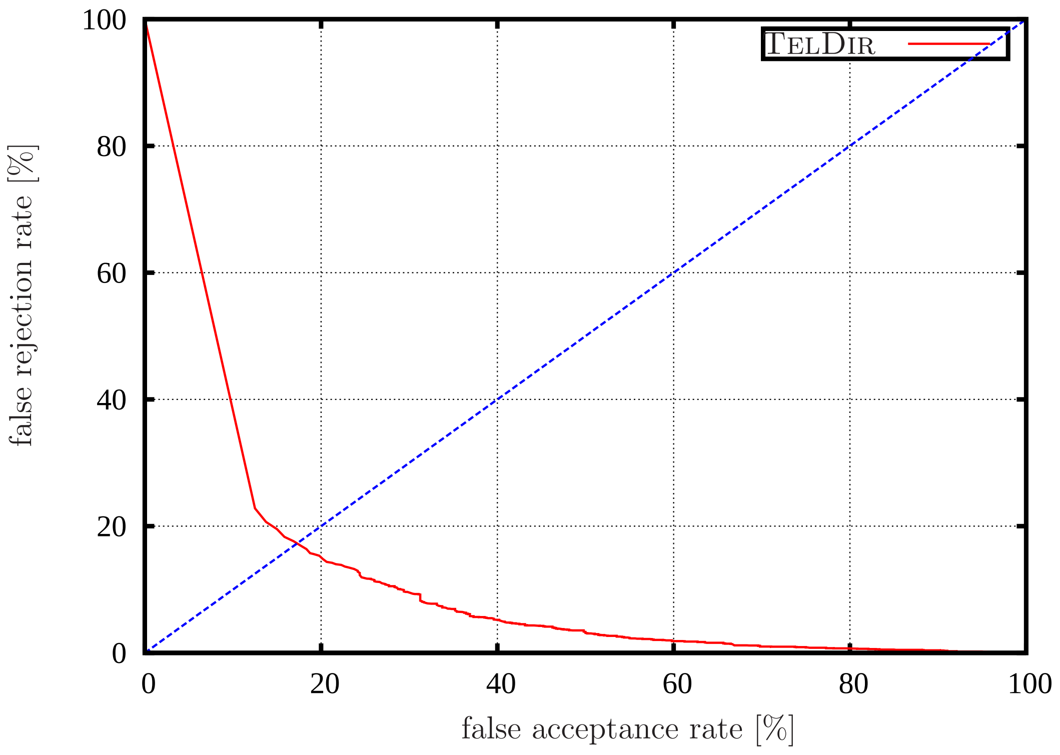


Figure 7.4: Detection error tradeoff curve for the TelDir corpus with the settings used for the field test. The point of intersection between the diagonal and the detection error tradeoff curve corresponds to the equal error rate.

attribute error rate was lower than 5%. Therefore, the more interesting input modality is speech. Despite the high word error rate, the concept translation proved to be robust against recognition errors. Table 7.2 summarizes the results for the 40 speech-based dialog sessions. In case of poor recognition performance, the low confidence values often caused the dialog manager to verify erroneous data explicitly.

The following two examples describe typical dialog sessions as they occurred during the logging of the 40 dialogs that were evaluated. Recognized words that were rejected due to low confidence values are underlined. The output of the natural language understanding component is given as a sequence of concepts for every input sentence. The tree's state and the chosen dialog action are listed explicitly in the examples, but were kept hidden from the users interacting with the dialogue system. The first example is a dialog where no recognition errors occurred. The second dialog covers the case, where the confidence measure caused the dialog manager to verify some information explicitly. For understanding purposes, the dialog sessions are translated into English.

System: Hello. This is the RWTH telephone directory assistance. You can ask questions about telephone numbers, fax numbers, and email addresses of persons as well as companies. How may I help you?

User: Uhm, what shall I do now?

recog: [hesitation] what shall I do now

nlu: `@FILLER`{what shall I do now}

tree: all tree nodes empty → select action: `ask_for_information`

System: Do you need a telephone number, a fax number, or an email address?

User: Can you give me the number of Mr. Macherey?

rec: can you give me the number of [pause] Mr. Macherey

nlu: `@can_question`{can you give me} `@telnum`{the number of} `@person`{Mr. Macherey}

tree: found path with completely filled nodes, SQL-size > 1
→ select action: `disambiguation`

System: I have more than one person entries with the name Macherey in my database. Please refine your request.

User: Mr. Wolfgang Macherey

rec: Mr. Wolfgang Macherey

nlu: `@person`{Mr. Wolfgang Macherey}

tree: found path with completely filled nodes, SQL-size = 1 → select action: `present`

System: The telephone number of Mr. Wolfgang Macherey is 80 21613. Shall I repeat the number?
. . .

Figure 7.5: First example of a real human-machine dialogue transaction that was logged during the field test of the TelDir dialogue system.

The two example dialogues from Figures 7.5 and 7.6 show some interesting properties of the tree-based dialogue management.

- **Mixed initiative dialogue system**
  The dialogue system is per construction a mixed-initiative system. The first example shows an inexperienced user who does not know how to use the dialogue system. As a result his first response leaves the knowledge tree in its initial state and the

System: Hello. This is the RWTH telephone directory assistance. You can ask questions about telephone numbers, fax numbers, and email addresses of persons as well as companies. How may I help you?

User: I need the number of Mrs. Hummel

recog: [noise] need the number of Mrs. Mail

nlu: `@need_question`{need the} `@telnum`{number of} `@person`{Mrs.} `@FILLER`{Mail}

tree: last name required, confidence low → select action: `verification`

System: Excuse me but I did not understand the last name. Please repeat.

User: Hummel

rec: Hummel

nlu: `@person`{Hummel}

tree: found path with completely filled nodes, SQL-size = 1 → select action: `present`

System: The telephone number of Mrs. Hummel is ...

Figure 7.6: Second example of a real human-machine dialogue transaction that was logged during the field test of the TelDir dialogue system.

dialogue system continues with asking whether the user wants a phone number, a fax number, or an email address. In contrast to this, the second example shows an experienced user who puts more information into his first utterance (phone number + person name). Although the last name was not understood, the dialogue system extracts the information and continues the transaction with a request for the missing information.

- **Ambiguity resolution**
  The first example shows a simple case of ambiguity resolution. The SQL database of the application contains two entries of people having the same last name. Therefore, the number of retrieved SQL results is larger than 1 and the path in the tree is considered to be ambiguous. As a result the user is asked to refine his request.

- **Error handling and recovery strategy**
  The second example shows a situation where a speech recognition error occurs. The person's last name is misrecognized, which results in a low confidence value for the last name. Independent of the speech recognition confidence value the language understanding module maps the misrecognized word onto the filler concept, which leaves the attribute value for the last name of the concept `@person` empty. Thus, the

Table 7.2: Dialogue evaluation using speech as input modality. As evaluation criteria, the attribute error rate (AER), the percentage of correct chosen successor states, and the percentage of successfully finished dialog sessions are used.

domain TELDIR # dialogues	AER [%]	choice of best successor state [%]	successful sessions [%]
40	18.4	88.4	90.0

dialogue manager receives two signals: (1) a low confidence and (2) an incomplete path caused by the missing attribute value. According to the decision tree depicted in Figure 7.3 the dialogue manager first checks the confidence of the tree's nodes before testing for the completeness of the paths. As a result, the verification action is chosen.

7.6 Summary

In this chapter, we described the framework for an application-independent dialogue management system. We proposed several feature functions for spoken dialog course management and investigated whether the proposed cost functions are able to select those dialog states during a dialog session that lead as quickly as possible to a final state that is likely to meet the user's request. In 88% of all cases, the dialog manager was able to choose the best successor state during a dialog session. Despite a relatively high word error rate, the statistical natural language understanding component proved to be robust against recognition errors.

One of the advantages of this approach is that the features do not depend on the given domain. For a concrete task, the domain-specific knowledge is provided by XML-based dialogue descriptions. The strict separation between application-dependent knowledge sources and the general feature-based dialogue management system makes it easy to change the application domain. Thus, the dialog management module can be used for different domains without the necessity to change the core implementation.

Another advantage is that the proposed tree-based dialogue system is a mixed-initiative system by construction because a user can always provide additional information without being explicitly asked for. In each dialogue turn, all extracted information is incorporated into the tree's knowledge base, which then affects the selection of the next dialogue action.

Chapter 8

Error Handling in a Tree-based Spoken Dialogue System

In spoken dialogue systems, errors can occur on different levels of the system's architecture. Because errors cannot be avoided in general, each component should have means of detecting errors. In this chapter, we take a closer look at how errors can be detected within a tree-based spoken dialogue system and what strategies can be implemented to correct errors.

8.1 Detecting Errors in a Spoken Dialogue System

One of the principal causes for errors during a dialogue transaction are erroneous recognitions, which often lead to incorrect semantic interpretations. Even if the speech input signal has been correctly recognized, the natural language understanding component can produce error-prone sentence meanings due to the limitations of its underlying model. Because errors cannot be avoided in general, the dialogue manager should at least be "aware" of such errors when they occur. To cope with this problem, we aim at detecting errors in those stages where they occur and propagate them through the next stage up to the dialogue manager. To measure the certainty of the components' outputs, we need confidence measures for both speech recognition and natural language understanding. The confidence measures are passed to the dialogue manager who then determines whether the collected information provided by the user must be confirmed or if another dialogue action should be chosen.

The distinction whether the speech recognition output or the language understanding output is likely to be wrong is of less importance for the dialogue manager. Therefore, the values from both confidence types are assigned to one out of two classes, *correct* or *wrong*, depending on the confidence thresholds, and then are combined using the logical *and* operator. However, in response to the modification of the dialogue strategy introduced in Section 8.3 we keep the actual confidence values and use the above described combination only when computing tree-costs and storing information in meta-states.

A shortcoming of passing confidence values directly to the dialogue manager is that intermediate components, such as the language understanding module, cannot benefit from the error assessment of the preceding components. Therefore, we want to investigate in the following section how speech recognition confidence values can be integrated into the language understanding module. For this, we refine the language understanding module by incorporating speech recognition confidence values as an additional feature function, thus, making the language understanding module dependent on the certainty of the recognition output. Note that this is a different approach compared to the combination proposed in Section 6.8. While in Section 6.8, the combination was performed as a rescoring step using optimal weights, we directly integrate the speech recognition confidence values as an additional feature into the language understanding model and train the model parameters using the GIS algorithm.

8.2 Confidence-Dependent Natural Language Understanding

During a dialogue transaction, the dialogue manager collects information provided by the automatic speech recognition and natural language understanding components and decides on the subsequent dialogue action. In case of recognition errors, the question is, whether the language understanding component can benefit from the confidence scores provided by the speech recognition module. Instead of just forwarding the speech recognition confidence scores to the dialogue manager, we want to incorporate the confidence values into the language understanding module. Because we have defined the language understanding approach within the maximum entropy framework, we can easily integrate the confidence scores of the speech recognition module by defining an appropriate feature function. For this purpose, we use the speech recognition confidence measures introduced in Section 5.3. Although the maximum entropy training is in general not restricted to binary valued feature functions, we use the confidence threshold in order to preclassify a word-confidence pair. The feature functions defined in Section 6.5.1 are defined for text-based input. However, the speech recognition confidence value of a hypothesized word depends on the acoustic observations and the hypothesized time boundaries for this word. Therefore, we extend the notation of a feature function by adding acoustic feature vectors and time boundaries to the argument list. Note that the original framework for text-based feature functions is a special case of this more general framework that we obtain if we replace the time boundaries by position indices and if use an empty observation vector. Using this extension, we can define the following binary feature function:

$$h_{\mathcal{C}_{\mathrm{ASR}},\tau,c}(c_{n-1}, c_n, [w;t]_{n-2}^{n+2}, x_1^T) = \begin{cases} 1 & \text{if } \tilde{p}([w_n; t_{n-1}+1, t_n] | x_1^T) \geqslant \tau \wedge c_n = c \\ 0 & \text{otherwise.} \end{cases} \tag{8.1}$$

Here, $[w;t]_{n-2}^{n+2} = [w_{n-2}; t_{n-3}+1, t_{n-2}], \ldots, [w_{n+2}; t_{n+1}+1, t_{n+2}]$ denotes the sequence of words together with the time boundaries, and τ denotes the speech recognition confidence threshold, which has been determined on the development test set beforehand. In contrast to the combination proposed in Section 6.8, where we combined speech recognition with language understanding knowledge sources on a sentence level by rescoring an N-best list repository with an optimal set of feature weights, we now directly incorporate a speech recognition knowledge source into the underlying model of the language understanding component. Because the new feature function requires a speech recognition confidence measure, we have to decode the training corpus in order to derive speech recognition confidence values. Details and experimental results can be found in Section 8.4.

8.3 Refined Confirmation Strategy

We shall refine the decision tree-based dialogue strategy introduced in Section 7.4 by allowing explicit and implicit verification questions. To accomplish this, the `is_confident?` node is modified such that the speech recognition confidence value is assigned to one out of three classes: correct, unsure, and wrong. The classification is based on two different thresholds τ_1 and τ_2, with $0 \leqslant \tau_1 < \tau_2 \leqslant 1$, separating the interval $[0, 1]$ into three disjoint sets $[0, \tau_1)$, $[\tau_1, \tau_2)$, and $[\tau_2, 1]$. If the confidence of the recognized utterance falls into the first interval $[0, \tau_1)$, an explicit confirmation dialogue is started. If the confidence value is an element of the second interval $[\tau_1, \tau_2)$, the dialogue manager will continue the dialogue with an implicit confirmation question. In the third case, no confirmation strategy is used. To also handle language understanding errors, we always choose an explicit confirmation strategy if the language understanding-based confidence value exceeds a given threshold. For each user input, a semantic analysis is performed. The concept-attribute pairs are

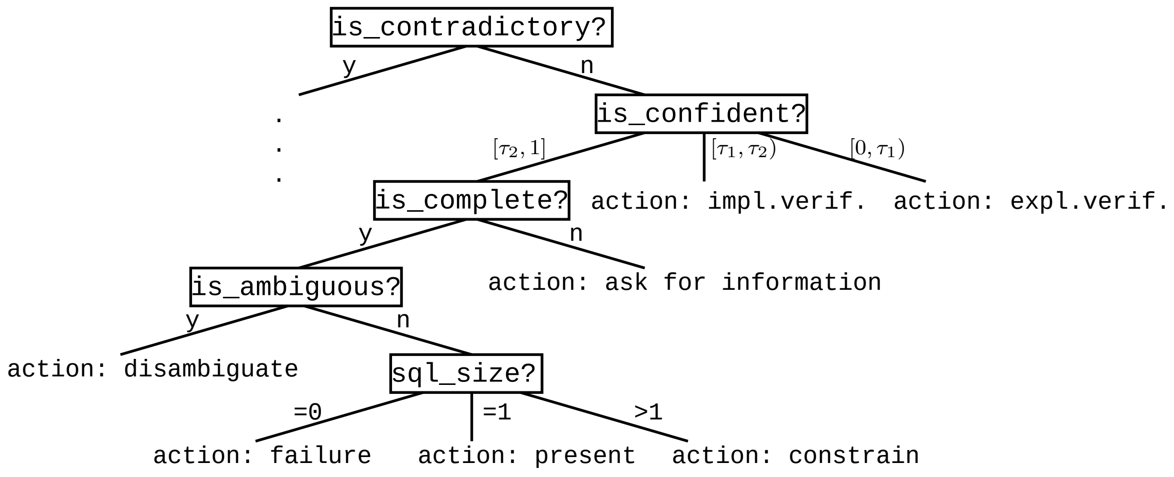


Figure 8.1: Refined decision tree for determining the subsequent dialogue action. The `is_confident?` node uses two confidence thresholds τ_1 and τ_2, thus, allowing for implicit and explicit verification questions.

extracted and inserted into temporary arrays for all tree nodes that are associated with these pairs. Temporary arrays are used in order to detect contradictory information. Cost vectors are computed for all nodes and paths of an instance tree according to Section 7.3. If there are several paths that may continue the dialogue, the dialogue manager will choose the path with the best score in order to proceed the dialogue. Note that the refined strategy of choosing an implicit confirmation question can affect the user's next utterance. At this point it is not clear whether allowing implicit confirmation questions is actually useful.

8.4 Results

Experiments for confidence-based natural language understanding were performed for both tasks TelDir and Taba. The refined dialogue strategy was evaluated in a second field test using a former version of the TelDir spoken dialogue system.

8.4.1 Confidence-Dependent Natural Language Understanding

Training a feature weight for the confidence-dependent feature function defined in Equation 8.1 requires a decoding step during training in order to obtain the confidence measures. However, decoding the training data with the same prototype vectors that we obtained from the acoustic model training would yield error rates that are too optimistic. Therefore, we split the training corpus into five partitions and estimate new prototype vectors in a round-robin fashion, that is, we train on four partitions and use the fifth partition as held-out set. By shifting the partitions five times, we can decode the training corpus and obtain error rates that are closer to the error rates we expect from the test corpora.

Because we want to preserve the correct sequence of concept tags, we keep only those speech-decoded source sentences that produce the same number of tokens as the reference source sentences. Thus, the new training corpus consists of two parts. The first part consist of the reference transcriptions and is equal to the corpus used in Chapter 6, that is, it contains no errors. The second part consists of the now speech-decoded training sentences. The so augmented training corpus is used to train a new model using the maximum entropy framework. The number of GIS iterations for the augmented training set was set to 1000.

Table 8.1 lists results for both language understanding tasks. For comparison reasons the table includes both the baseline results that were obtained using the training reference sentences as well as the baseline results that were obtained when using the augmented training corpus. The table presents results with and without categorization. The results show small improvements for the Taba task. However, for the TelDir task, the performance slightly deteriorates, which might be caused by the fact that the overall number of training data for TelDir is much smaller compared with Taba. Thus, the results show that on top of the baseline system, adding speech recognition-based confidence features

Table 8.1: Confidence-dependent natural language understanding. The "augmented training corpus" row lists slot error rates for a model trained on the augmented training corpus. The last row in each table shows results for the augmented natural language understanding (NLU) model that contains the automatic speech recognition (ASR) confidence value-based feature function.

corpus TelDir-Speech	Slot-ER [%]		Slot-ER [%] + Categ	
Features	development	evaluation	development	evaluation
Baseline	10.3	12.7	6.8	10.6
augmented training corpus	10.3	12.7	6.9	10.9
+ ASR confidence values	10.4	12.6	6.8	11.1

corpus Taba-Speech	Slot-ER [%]		Slot-ER [%] + Categ	
Features	development	evaluation	development	evaluation
Baseline	13.7	12.6	11.4	11.6
augmented training corpus	13.5	12.7	11.1	11.2
+ ASR confidence values	13.5	12.6	11.2	11.2

do not generalize well for the TelDir task.

Table 8.2: Dialogue evaluation using a refined confirmation strategy. As evaluation criteria, the slot error rate, the attribute error rate, and the percentage of successfully finished dialogue sessions are used.

domain TelDir # dialogues	Slot-ER [%]	AER [%]	successful sessions [%]
35	17.0	15.0	88.6

8.4.2 Refined Dialogue Strategy

In a second field test, we have recorded 35 dialogue sessions of users interacting with a former version of the TelDir spoken dialogue system that did not include spelling units. The recorded dialogues were evaluated manually by comparing the system's decisions with human judged decisions. The results are listed in Table 8.2. The percentage of successful sessions is with 88.6% smaller than the success rate of 90.0% that was achieved in the first field test. Although the number of evaluated dialogue sessions is too small in order to draw conclusions, it seems that implicit confirmation strategies may more often lead to confusion because in case of an error the user gets the impression that the dialogue system tries to continue the dialogue with a wrong knowledge base, which negatively affects subsequent user responds compared to the case where information is explicitly confirmed. Therefore, explicit confirmation strategies may be more suitable for error-tolerant spoken dialogue systems.

8.5 Summary

In this chapter, we used confidence measures based on posterior probabilities for both modules the automatic speech recognition component and the natural language understanding component. The confidence scores of both components are passed as features to the dialogue manager who then determines the subsequent dialogue action. We refined the confirmation strategy by using implicit and explicit verification questions. The refined dialogue strategy was tested within a field test where we logged and analyzed 35 dialogue sessions.

We also extended the language understanding framework by directly taking the speech recognition-based confidence measure into account. For this, the speech recognition-based confidence values were preclassified into one out of two classes and integrated as an additional feature function into the maximum entropy-based language understanding component.

Chapter 9

Summary and Outlook

This chapter summarizes the main contributions of this thesis and presents an outlook on interesting and promising extensions of the work presented in the previous chapters.

9.1 Summary

In this thesis, we proposed and investigated statistical methods for natural language understanding and spoken dialogue systems. The main contributions are summarized in the following itemization.

- **Natural language understanding is a machine translation problem**
  We defined the problem of natural language understanding as a special instance of a machine translation problem where we translate from a natural language sentence into a formal concept language. We investigated two approaches that result from rewriting the decision rule for natural language understanding: the first approach is based on the source-channel paradigm and factorizes the posterior probability into two separate distributions, the translation probability and the lexicon probability; the second approach is based on maximum entropy and directly models the posterior probability.

- **Effect of alignments, categorizations, and contexts on slot errors**
  We investigated the effect of different alignment models and categorizations on the slot error rate and proposed several feature functions that are suitable for natural language understanding. We showed the importance of local context dependencies and analyzed the effect of maximum entropy-based segmentations. Furthermore, we investigated how both approaches perform if speech input is used as opposed to text input.

- **Confidence measures for natural language understanding**
  We defined posterior-based confidence measures for natural language understanding that are computed on concept graphs and alternatively on N-best lists. Together with the speech recognition-based confidence measures, the language understanding

confidence measures are passed to the dialogue manager who takes these values into account when determining the next dialogue action.

- **Combining speech recognition and natural language understanding**
  We provided an efficient framework for combining speech recognition with natural language understanding. Using the minimum error training framework we defined several feature functions derived from speech recognition and natural language understanding and showed significant improvements when combining all knowledge sources in a log linear manner. We showed that a fixed repository of N-best lists is sufficient in order to determine the feature weights. Furthermore, we showed that the word error rate can be reduced using language understanding-based feature functions, and that, vice-versa, the slot error rate can be reduced using speech recognition-based feature functions. We also showed that both error types can be minimized simultaneously if the error counts are combined. To the best of our knowledge this is the first time that the minimum error training framework is applied to the problem of natural language understanding using speech as input modality.

- **Domain-independent dialogue management**
  We proposed and implemented a domain-independent dialogue manager that uses trees as fundamental data structure. Based on several feature functions, the dialogue manager analyses the tree's information content and decides about the next dialogue action depending on a cost function. The proposed dialogue architecture is per construction a mixed initiative system. We showed that it is easy to integrate additional knowledge sources into the system and evaluated the overall performance of the dialogue system in two field tests.

- **Error-tolerant dialogue modeling**
  We extended the set of feature functions used for the maximum entropy-based natural language understanding component by confidence-based features derived from automatic speech recognition. For both tasks except for the TelDir evaluation corpus this lead to a slight deterioration in terms of slot-error rate. Furthermore, we refined the confirmation strategy of the dialogue system by introducing implicit and explicit verification questions depending on the current confidence level. The refined dialogue strategy was evaluated within a field test for the TelDir task.

- **Efficient computation of confidence measures for speech recognition**
  We have described an efficient algorithm for computing confidence measures based on word-conditioned word graphs. The algorithm described is suitable for online recognitions under realtime constraints. In contrast to methods described in the literature that are based on time-conditioned word graphs we showed that a computation based on word-conditioned graphs significantly reduces the computational effort because no additional language model lookups are necessary. We also described and analyzed various strategies for applying an F-MLLR online adaptation

that is suitable for realtime recognition systems. We described a modified forward-backward pruning algorithm for word-conditioned word graphs that has many useful properties suitable for realtime recognition systems.

- **Implementation and overall system architecture**
  For the investigations reported in this thesis, the author has implemented a full dialogue system including the online realtime speech recognition system, the natural language understanding component, and the dialogue system. The code base consists of around 65K lines of mainly C++ code, excluding third party software or tools provided by others. The speech recognition training environment as well as the decoding environment for offline tests were fully parallelized and distributed over an SGE cluster. The speech recognition and the language understanding module were designed such that they share the same decoder core as well as the implementation of graphs, N-best lists, confidence measures, and so on. In addition to the dialogue management system, a dialogue description language based on XML was developed that is described in part in the appendix of this thesis.

9.2 Outlook

There are many promising extensions and interesting alternatives to the work presented in this thesis from which some are listed in the following.

- **Natural language understanding as an information retrieval task**
  In this thesis, the task of natural language understanding was defined as a special translation problem where we translate from a natural language input to a formal target language that consists of concepts. This approach requires a bilingual training corpus. Although we used flat concepts and avoided to provide explicit alignments between words and concepts, which makes the annotation cheap, an approach where no intermediate meaning representation is necessary would be even more appealing. A first step towards this goal could be to define natural language understanding as a special information retrieval task. Instead of transforming sequences of concepts into SQL queries, the application database could be defined as a large set of documents where each valid dialogue goal together with the answer defines one document. The goal would then be to retrieve the document that satisfies all the constraints the user has given. If too many documents remain the user is asked for more information in order to further narrow down the number of retrieved documents. This could define a first approach where the input sentence itself is used as meaning representation.

- **Episode features for spoken language understanding**
  The combination of speech recognition and natural language understanding could be further refined. In the context of spoken dialogues systems one could also introduce episode and discourse features that take previous user utterance and

derived concepts into account. For example, if during a dialogue transaction, the value of a concept was explicitly confirmed, this information could be directly integrated into the language understanding model as an additional feature function.

- **Maximum entropy-based dialogue strategies**
  The decision tree-based dialogue strategy described in this thesis employs several feature functions that are derived from the current information state of the dialogue system. This could be easily replaced by a maximum entropy-based classifier where the dialogue actions define the classes. In order to derive a sufficiently large number of training examples, dialogue simulations would be necessary in order to train the maximum entropy-based model. This would also allow a comparison of different dialogue strategies based on decision trees, maximum entropy models, and reinforcement learning approaches.

Appendix A

Symbols and Acronyms

A.1 Mathematical Symbols

A^*	adjoint matrix of A
$A^\top$	transposed matrix A
c_1^M	sequence of concepts
f_1^J	source sentence
e_1^I	target sentence
f_j	source word in position j of the sentence
e_i	target word in position i of the sentence
a_1^J	word alignment mapping from source to target positions
ℓ_i	N-best list entry
η	hidden Markov model
$h(\dots)$	feature function
λ_1^L	feature weights
$\mathcal{C}_{\text{ASR}}$	confidence measure for automatic speech recognition
$\mathcal{C}_{\text{NLU}}$	confidence measure for natural language understanding
$\mathcal{L}(c_1^M, \tilde{c}_1^{\tilde{M}})$	Levenshtein alignment between sentences c_1^M and $\tilde{c}_1^{\tilde{M}}$
$\mathcal{V}$	vocabulary
$\mathcal{F}^d$	feature space
$\mathcal{K}$	set of word classes
Q	auxiliary function
$[c; \nu, n]$	concept hypothesis with start- and end position
$[w; \tau, t]$	word hypothesis with start- and end time
w_1^N	sequence of words
x_1^T	sequence of acoustic vectors
τ	start time or threshold
$\Phi(\dots)$	forward probability
$\Psi(\dots)$	backward probability
pr	general distribution
p	model distribution
$\delta(\cdot, \cdot)$	Kronecker delta
π	path or policy
$V_\pi(s)$	state value function
$Q_\pi(s, a)$	state-action value function
r_t	reward at time t

A.2 Acronyms

AER	**a**ttribute **e**rror **r**ate
ASR	**a**utomatic **s**peech **r**ecognition
AT	**a**lignment **t**emplate
ATIS	**a**ir **t**ravel **i**nformation **s**ervices
BGD	**b**oundary **g**raph **d**ensity
CAS	**c**ommon **a**nswer **s**pecification
CER	**c**onfidence error **rate**
CRIM	**C**entre de **R**echerche **I**nformatique de **M**ontréal
DET	**de**tection **e**rror **t**radeoff
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
DET	**de**tection **e**rror **t**radeoff
F-MLLR	**f**eature space **m**aximum **l**ikelihood **l**inear **r**egression
EM	**e**xpectation **m**aximization
GER	**g**raph **e**rror **r**ate
GIS	**g**eneralized **i**terative **s**caling
HMM	**h**idden **M**arkov **m**odel
LDA	**l**inear **d**iscriminant **a**nalysis
LDC	**l**inguistic **d**ata **c**onsortium
MDP	**m**arkov **d**ecision **p**rocess
ME	**m**aximum **e**ntropy
MFCC	**m**el **f**requency **c**epstral **c**oefficients
MLLR	**m**aximum **l**ikelihood **l**inear **r**egression
MT	**m**achine **t**ranslation
NGD	**n**ode **g**raph **d**ensity
NLG	**n**atural **l**anguage **g**eneration
NLU	**n**atural **l**anguage **u**nderstanding
PCFG	**p**robabilistic **c**ontext **f**ree **g**rammar
POMDP	**p**artially **o**bservable **m**arkov **d**ecision **p**rocess
SCT	**s**emantic **c**lassification **t**ree
SDS	**s**poken **d**ialogue **s**ystem
SER	**s**entence **e**rror **r**ate
SGE	**s**un **g**rid **e**ngine
Slot-ER	**slot e**rror **r**ate
SLU	**s**poken **l**anguage **u**nderstanding
SQL	**s**tructured **q**uery **l**anguage
SLU	**s**poken **l**anguage **u**nderstanding
TTS	**t**ext **t**o **s**peech
WER	**w**ord **e**rror **r**ate
WGD	**w**ord **g**raph **d**ensity

Appendix B

Concept Inventory

B.1 Concepts and Attributes for the TELDIR Corpus

concept	attribute		concept	attribute	
	name	type		name	type
requests			**others**		
• `@auskunft`	—		• `@please`	—	
• `@brauchen_Frage`	—		• `@filler`	—	
• `@koennen_Frage`	—		• `@spelling`	sequence	string
• `@wollen_Frage`	—		**objects**		
greetings			• `@department`	name	string
• `@hello`	—		• `@organization`	name	string
• `@introduction`	—		• `@person`	surname	string
conjunction				forename	string
• `@and`	and	bool		gender	string
• `@or`	or	bool		occupation	string
yes/no			**addresses**		
• `@yes`	yes	bool	• `@address`	flag	bool
• `@no`	no	bool		result	string
phone, fax, email			• `@location`	location	string
• `@connect`	flag	bool			
	result	string			
• `@email`	flag	bool			
	result	string			
• `@faxnum`	flag	bool			
	result	string			
• `@phone`	numberFlag	bool			
	officeFlag	bool			
	homeFlag	bool			
	mobileFlag	bool			
	officeResult	string			
	homeResult	string			
	mobileResult	string			

B.2 Concepts and Attributes for the TABA Corpus

concept	attribute	
	name	type
requests		
• @brauchen_Frage	—	
• @koennen_Frage	—	
• @kommen_Frage	—	
• @wann_Frage	—	
• @wollen_Frage	—	
• @wollen_Aussage	—	
• @Zugbestimmung	—	
greetings		
• @guten_Tag	—	
origin and destination		
• @origin	origin	string
• @origin_and_destin	origin	string
	destination	string
• @destination	destination	string
• @destin_and_arrival	arrivalTime	time
	destination	string
• @lp_origin_and_destin	origin	string
	destination	string
date and time expressions		
• @complete_time	exactTime	bool
	startTime	time
	endTime	time
• @date	date	date
• @day_time	startTime	time
	endTime	time
• @time_and_date	date	date
	startTime	time
	endTime	time
• @time_and_date_ao	date	date
	exactTime	bool
	startTime	time
	trainTime	string
• @train_time	train_time	string
• @arrival	arrival	time
yes/no		
• @yes	yes	bool
• @no	no	bool
others		
• @kommen	—	
• @umsteigen	—	
• @bitte	—	
• @filler	—	
• @meaningful_filler	—	

B.3 Concept Automata

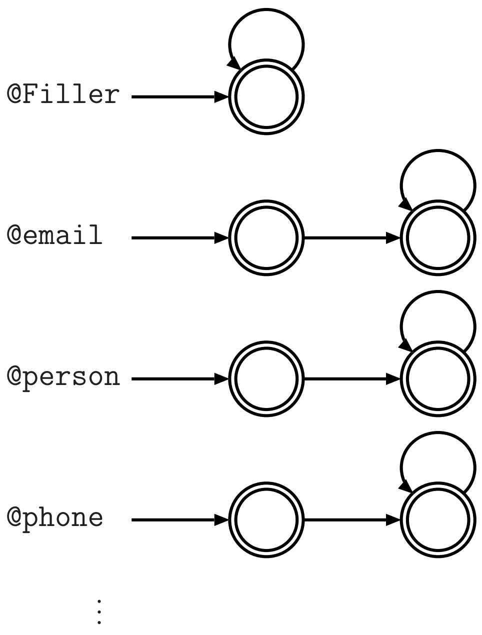


Figure B.1: Concept automata for the TelDir corpus.

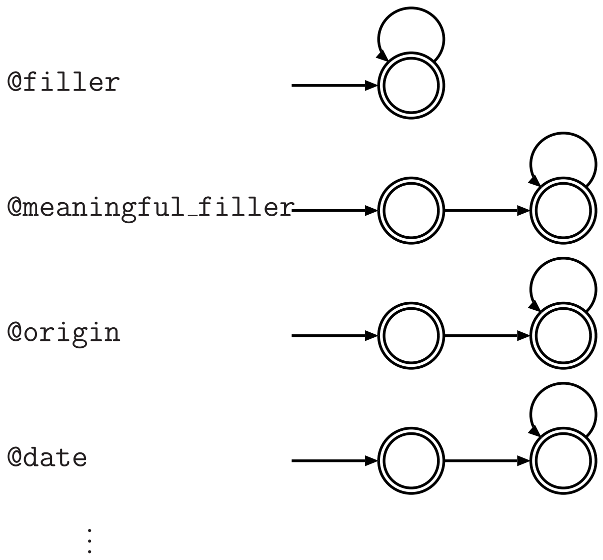


Figure B.2: Concept automata for the Taba corpus.

Appendix C

An XML-based Dialogue Description Language

The feature functions and selection criterion described in chapter 7 do not use any task-specific knowledge and can therefore be used for different applications of spoken dialogue systems. For a concrete task, however, some domain-specific knowledge must be defined that is provided by three XML files. These files are the concept-attribute table defining the concepts and attributes for the natural language understanding part, the knowledge tree collecting and storing information provided by the natural language understanding module, and the dialogue description specifying concrete dialogue actions.

C.1 Concept-Attribute Table

To represent the meaning of an utterance, a set of domain-specific concepts is used. A concept is defined as the smallest unit of meaning that is relevant to a specific task. Each concept may have several attributes. In order to derive the values from the concepts, we assume that the attributes only depend on their associated concept and its aligned words. Each attribute can have two rules: one for deriving the attribute values and one for converting the values into a more formal SQL constraint. Figure C.1 shows a part of the XML-based concept-attribute table for the telephone directory assistance task.

C.2 Knowledge Tree

This XML file specifies the structure of the knowledge tree. A knowledge tree stores information extracted from the user utterances during a dialogue. Each node of the knowledge tree contains a list of concepts that is associated with this specific node. Additionally, each tree node has a link to a dialogue state as specified in the dialogue description file (see Section C.3). For the dialogue manager, this link acts as a mediator between the knowledge representation and the dialogue description. While the determination of the subsequent dialogue action only depends on the analysis of the tree's information content, the concrete realization and execution of this action is not part of the tree specification but of the dialogue description. If, for example, the dialogue manager has determined

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE concept_attribute_table SYSTEM "ca_table.dtd">

<concept_attribute_table>
  <concept name="@person">
    <attrib name="surname"    type="str" value="" nlu=".." sql=".." rating="man"/>
    <attrib name="forename"   type="str" value="" nlu=".." sql=".." rating="sup"/>
    <attrib name="gender"     type="str" value="" nlu=".." sql=".." rating="sup"/>
    <attrib name="occupation" type="str" value="" nlu=".." sql=".." rating="sup"/>
  </concept>
      ⋮
</concept_attribute_table>
```

Figure C.1: Part of an XML-based concept-attribute table. Each concept may consist of several attributes. An attribute is a tuple consisting of the attribute's name, the type, and its value. The `rating` specifies the importance of the attribute and can be either mandatory or supplementary.

the best scored path and decides for the subsequent dialogue action, he will switch to the next dialogue state as given by the node's dialogue state name and executes the dialogue commands of this dialogue state-action pair. A part of the knowledge tree specification for the telephone directory assistance task is listed in Figure C.2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE knowledge_tree SYSTEM "knowledge_tree.dtd">

<knowledge_tree name="telephone directory assistance" root="0">
  <!-- node definitions -->
  <node name="0" alias="Telephone Inquiries" dstate="0">
    <concept name="@can_question"/>
    <concept name="@want_question"/>
    <concept name="@need_question"/>
  </node>

  <node name="0.1"     alias="email" dstate="0.1">
    <concept name="@email"/>
  </node>

  <node name="0.2"     alias="fax" dstate="0.2">
    <concept name="@faxnum"/>
  </node>

  <node name="0.3"     alias="telephone number" dstate="0.3">
```

```
    <concept name="@number"/>
  </node>
    ⋮
  <node name="0.1.1"   alias="person" dstate="0.1.1">
    <concept name="@person"/>
    <concept name="@spelling"/>
    <concept name="@location"/>
  </node>

  <node name="0.1.2"   alias="company/institute" dstate="0.1.2">
    <concept name="@organization"/>
    <concept name="@spelling"/>
    <concept name="@location"/>
  </node>

  <!-- edge definitions -->
  <edge> 0,0.1 </edge>
  <edge> 0,0.2 </edge>
  <edge> 0,0.3 </edge>
  <edge> 0,0.4 </edge>
    ⋮
</knowledge_tree>
```

Figure C.2: Part of an XML-based knowledge tree representation. Each node of the tree has a unique name, an alias, and a dstate referring to a dialogue state in the dialogue description. A node's child tags describe the list of concepts that are associated with the node. The edges between the nodes are defined within the `edge definitions` section. The attribute `root` of the tag `knowledge_tree` defines the name of the tree's root node.

C.3 Dialogue Description

The XML-based dialogue description consists of different dialogue states, subdividing a dialogue into smaller sections. In addition to dialogue states for ordinary tasks, such as greeting the user or presenting some introductory information, there are more specialized dialogue states corresponding to the nodes of the knowledge tree. These dialogue states will be chosen by the dialogue manager if the subsequent dialogue state is not explicitly determined by the dialogue description itself. In the dialogue state `START` of Figure C.3, for example, the subsequent dialogue state is explicitly determined by the `next_state` command, whereas for dialogue state `0` the following dialogue state must be determined by the dialogue manager.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dialog_description SYSTEM "dialog_description.dtd">

<dialogue>
  <dialogue_state name="START">
    <action_state name="arbitrary">
      <set variable="TTS_DIR" type="str" value="..." />
      <next_state name="GREETINGS"/>
    </action_state>
  </dialogue_state>

  <dialogue_state name="GREETINGS">
    <action_state name="arbitrary">
      <greetings time="0-17"  text="Good morning."  barge_in="0"/>
      <greetings time="18-23" text="Good evening."  barge_in="0"/>
      <tts barge_in="1"> How may I help you? <tts/>
      <nlu/>
    </action_state>
  </dialogue_state>

  <dialogue_state name="0">
    <action_state name="guidance">
      <tts> Do you need a phone number, fax number, or email address? <tts/>
      <nlu/>
    </action_state>
  </dialogue_state>
      ⋮
</dialogue>
```

Figure C.3: Part of an XML-based dialogue description. A dialogue description consists of several dialogue states where each dialogue state is subdivided into different action states. Only those dialogue commands occurring in an action state that corresponds to the dialogue action as chosen by the dialogue manager are executed.

Since the actions that are performed during a dialogue turn depend on the tree's information content, the dialogue states are further subdivided into different action states. Beside others, these action states include the collection and presentation of information as well as disambiguation and verification of information. Only those dialogue commands that occur within the action state chosen by the dialogue manager are executed. A special action-state is the `arbitrary` state whose actions are always executed, regardless of the current dialogue action chosen by the dialogue manager. Figure C.3 shows a part of the dialogue description for the telephone directory assistance task.

Bibliography

[Abella & Gorin 99] A. Abella, A.L. Gorin: Construct Algebra: Analytical Dialog Management. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 191–199, College Park, Maryland, June 1999.

[Allauzen & Mohri^{+} 04] C. Allauzen, M. Mohri, M. Riley: Statistical Modeling for Unit Selection in Speech Synthesis. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 55–62, Barcelona, Spain, July 2004.

[Allen & Miller^{+} 96] J.F. Allen, B.W. Miller, E.K. Ringger, T. Sikorski: Robust Understanding in a Dialogue System. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 62–70, Santa Cruz, CA, June 1996.

[Allen 95] J. Allen: *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc., 1995.

[Ammicht & Potamianos^{+} 01] E. Ammicht, A. Potamianos, E. Fosler-Lussier: Ambiguity Representation and Resolution in Spoken Dialogue Systems. In *Proc. of the 7th European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 3, pp. 2217–2220, Aalborg, Denmark, Sept. 2001.

[Austin 62] J.L. Austin: *How to do Things with Words.* Clarendon Press, 1962.

[Baker 75] J.K. Baker. Stochastic Modeling for Automatic Speech Understanding. In D.R. Reddy, editor, *Speech Recognition*, pp. 512–542. Academic Press, New York, NY, USA, 1975.

[Bakis 76] R. Bakis. Continuous Speech Recognition via Centisecond Acoustic States. Technical Report RC 5971, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, pp. 1–8, April 1976.

[Basch & Guibas^{+} 97] J. Basch, L.J. Guibas, G.D. Ramkumar: Sweeping Lines and Line Segments with a Heap. In *Proc. of the 13th Annual ACM Sympos. on Comput. Geom.*, pp. 469–471, Nice, France, June 1997.

[Bates & Bobrow^{+} 94] M. Bates, R. Bobrow, R. Ingria, S. Peters, D. Stallard: Advances in BBN's Spoken Language System. In *Proc. of the Spoken Language Technology Workshop*, pp. 43–47, March 1994.

[Baum 72] L.E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. In O. Shisha, editor, *Inequalities*, Vol. 3, pp. 1–8. Academic Press, New York, NY, 1972.

[Bayes 63] T. Bayes: An Essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Society of London*, Vol. 53, pp. 370–418, 1763. Reprinted in *Biometrika*, vol. 45, no. 3/4, pp. 293–315, December 1958.

[Bellman 57] R.E. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, USA, 1957.

[Bender & Macherey$^+$ 03] O. Bender, K. Macherey, F.J. Och, H. Ney: Comparison of Alignment Templates and Maximum Entropy Models for Natural Language Understanding. In *Proc. of the 10th Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 11–18, Budapest, Hungary, April 2003.

[Bender 02] O. Bender. Untersuchung zur Tagging-Aufgabenstellung in der Sprachverarbeitung. Diploma thesis, RWTH-Aachen, University of Technology, Aachen, Germany, Oct. 2002.

[Bennacef & Bonnea-Maynard$^+$ 94] S.K. Bennacef, H. Bonnea-Maynard, J.L. Gauvain, L.F. Lamel, W. Minker: A Spoken Language System for Information Retrieval. In *Proc. of the 3rd Int. Conf. on Spoken Language Processing (ICSLP)*, pp. 1271–1274, Sept. 1994.

[Bentley & Ottmann 79] J.L. Bentley, T.A. Ottmann: Algorithms for Reporting and Counting Geometric Intersections. *IEEE Trans. on Computers*, Vol. C-28, No. 9, pp. 643–647, Sept. 1979.

[Berger 97] A. Berger. Improved Iterative Scaling: A gentle introduction, 1997. http://www.cs.cmu.edu/afs/cs/user/aberger/www/ps/scaling.ps.

[Beyerlein 00] P. Beyerlein. *Diskriminative Modellkombination in Spracherkennung mit großem Wortschatz*. Ph.D. Thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, 2000.

[Borthwick & Sterling$^+$ 98] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman: Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In *Proc. of the 6th Workshop on Very Large Corpora*, pp. 152–160, Montreal, Canada, Aug. 1998.

[Brown & Burton 75] J.S. Brown, R.R. Burton. Multiple representations of Knowledge for Tutorial Reasoning. In D.G. Bobrow, A. Collins, editors, *Representation and Understanding*, pp. 311–350. Academic Press, New York, 1975.

[Brown & Della Pietra$^+$ 93] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer: The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[Casacuberta & Llorens$^+$ 01] F. Casacuberta, D. Llorens, C. Martínez, S. Molau, F. Nevado, H. Ney, M. Pastor, D. Picó, A. Sanchis, E. Vidal, J.M. Vilar: Speech-to-Speech Translation based on Finite-State Transducers. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 613–616, Salt Lake City, Utah, May 2001.

[Chen & Goodman 98] S.F. Chen, J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Center for Research in Computing Technology Harvard University, Cambridge, Massachusetts, 64 pages, 1998.

[Chen & Rosenfeld 99] S.F. Chen, R. Rosenfeld. A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMU-CS-99-108, School of Computer Science Carnegie Mellon University, Pittsburgh, PA, 25 pages, Feb. 1999.

[Chow & Roukos 89] Y.L. Chow, S. Roukos: Speech Understanding Using a Unification Grammar. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 727–730, Glasgow, Scotland, May 1989.

[Chow & Schwartz 89] Y.L. Chow, R. Schwartz: The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses. In *DARPA Speech and Natural Language Workshop*, pp. 199–202, Cape Cod, Oct. 1989.

[Darroch & Ratcliff 72] J.N. Darroch, D. Ratcliff: Generalized Iterative Scaling for Log-Linear Models. *Annals of Mathematical Statistics*, Vol. 43, pp. 1470–1480, 1972.

[Davis & Mermelstein 80] S.B. Davis, P. Mermelstein: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. on Speech and Audio Processing*, Vol. 28, pp. 357–366, Aug. 1980.

[Della Pietra & Epstein$^+$ 97] S. Della Pietra, M. Epstein, S. Roukos, T. Ward: Fertility Models for Statistical Natural Language Understanding. In *Proc. of the 8th Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 168–173, Madrid, Spain, July 1997.

[den Os & Boves$^+$ 99] E. den Os, L. Boves, L. Lamel, P. Baggia: Overview of the ARISE Project. In *Proc. of the 6th European Conf. on Speech Communication and Technology (EUROSPEECH)*, pp. 1527–1530, Budapest, Hungary, Sept. 1999.

[Eckert & Levin$^+$ 97] W. Eckert, E. Levin, R. Pieraccini: User Modeling for Spoken Dialogue System Evaluation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 80–87, Santa Barbara, California, Dec. 1997.

[Epstein & Papineni$^+$ 96] M. Epstein, K. Papineni, S. Roukos, T. Ward, S. Della Pietra: Statistical Natural Language Understanding Using Hidden Clumpings. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 176–179, Atlanta, Georgia, May 1996.

[Evermann 99] G. Evermann. Minimum Word Error Rate Decoding. Master thesis, Churchill College, University of Cambridge, Cambridge, England, Aug. 1999.

[Ferguson & Allen$^+$ 96] G.M. Ferguson, J.F. Allen, B.W. Miller, E.K. Ringger. The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant. Technical Report 96-5, University of Rochester, 164 pages, Oct. 1996.

[Fisher 36] R.A. Fisher: The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, Vol. 7, pp. 179–188, 1936. reprinted in: *Contributions to Mathematical Statistics, 1950.*

[Gauvain & Bennacef$^+$ 97] J.L. Gauvain, S. Bennacef, L. Devillers, L. Lamel, S. Rosset. Spoken Language Component of the MASK Kiosk. In K. Varghese, S. Pfleger, editors, *Human Comfort & Security of Information Systems*, pp. 93–103. Springer Verlag, 1997.

[Goddeau & Pineau 00] D. Goddeau, J. Pineau: Fast Reinforcement Learning of Dialog Strategies. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 2, pp. 1233–1236, Istanbul, Turkey, June 2000.

[Hazen & Burianek^{+} 00] T.J. Hazen, T. Burianek, J. Polifroni, S. Seneff: Integrating Recognition Confidence Scoring with Language Understanding and Dialogue Modeling. In *Proc. of the 6th Int. Conf. on Spoken Language Processing (ICSLP)*, Vol. 2, pp. 1042–1045, Beijing, China, Oct. 2000.

[He & Young 03a] Y. He, S. Young: A Data-Driven Spoken Language Understanding System. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 583–588, St. Thomas, U.S. Virgin Islands, Dec. 2003.

[He & Young 03b] Y. He, S. Young: Hidden Vector State Model for Hierarchical Semantic Parsing. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 268–271, Hong Kong, China, April 2003.

[Issar & Ward 93] S. Issar, W. Ward: CMU's Robust Spoken Language Understanding System. In *Proc. of the 3rd European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 3, pp. 2147–2149, Berlin, Germany, Sept. 1993.

[Jelinek 69] F. Jelinek: A Fast Sequential Decoding Algorithm Using a Stack. *IBM Journal of Research and Development*, Vol. 13, pp. 675–685, Nov. 1969.

[Juang & Chou^{+} 95] B.H. Juang, W. Chou, C.H. Lee. Statistical and Discriminative Methods for Speech Recognition. In A.J. Rubio Ayuso, J.M. López Soler, editors, *Speech Recognition and Coding - New Advances and Trends*, Vol. 147 of *Series F: Computer and Systems Sciences*, pp. 41–55. Springer Verlag, Berlin Heidelberg, 1995.

[Kellner & Rueber^{+} 97] A. Kellner, B. Rueber, F. Seide, B.H. Tran: PADIS - An automatic telephone switchboard and directory information system. *Speech Communication*, Vol. 23, No. 1–2, pp. 95–111, Oct. 1997.

[Kemp & Schaaf 97] T. Kemp, T. Schaaf: Estimating Confidence Using Word Lattices. In *Proc. of the 5th European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 2, pp. 827–830, Sept. 1997.

[Komatani & Kawahara 00] K. Komatani, T. Kawahara: Generating Effective Confirmation and Guidance Using Two-Level Confidence Measures for Dialogue Systems. In *Proc. of the 6th Int. Conf. on Spoken Language Processing (ICSLP)*, Vol. 2, pp. 648–651, Beijing, China, Oct. 2000.

[Kuhn & de Mori 95] R. Kuhn, R. de Mori: The Application of Semantic Classification Trees to Natural Language Understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 5, pp. 449–460, 1995.

[Lamel & Rosset^{+} 98] L. Lamel, S. Rosset, J. Gauvain, S. Bennacef, M. Garnier-Rizet, B. Prouts: The LIMSI ARISE System. In *Interactive Voice Technology for Telecommunications Applications (IVTTA)*, pp. 209–214, Turin, Italy, Sept. 1998.

[Leggetter & Woodland 95] C.J. Leggetter, P.C. Woodland: Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, Vol. 9, No. 2, pp. 171–185, April 1995.

[Levin & Pieraccini 95] E. Levin, R. Pieraccini: Concept-Based Spontaneous Speech Understanding System. In *Proc. of the 4th European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 2, pp. 555–558, Madrid, Spain, Sept. 1995.

[Levin & Pieraccini 97] E. Levin, R. Pieraccini: A stochastic model of computer-human interaction for learning dialogue strategies. In *Proc. of the 5th European Conf. on Speech Communication and Technology (EUROSPEECH)*, pp. 1883–1886, Rhodes, Greece, Sept. 1997.

[Levin & Pieraccini$^+$ 98] E. Levin, R. Pieraccini, W. Eckert: Using Markov Decision Process for Learning Dialogue Strategies. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 201–204, Seattle, WA, USA, May 1998.

[Levin & Pieraccini$^+$ 00] E. Levin, R. Pieraccini, W. Eckert: A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans. on Speech and Audio Processing*, Vol. 8, No. 1, pp. 11–23, Jan. 2000.

[Litman & Walker$^+$ 99] D.J. Litman, M.A. Walker, M.S. Kearns: Automatic Detection of Poor Speech Recognition at the Dialogue Level. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 309–316, College Park, Maryland, June 1999.

[Lowerre 76] B. Lowerre. *A Comparative Performance Analysis of Speech Understanding Systems.* Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA, USA, 1976.

[Macherey & Bender$^+$ 03] K. Macherey, O. Bender, H. Ney: Multi-Level Error Handling for Tree-Based Dialogue Course Management. In *Proc. of the ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, pp. 123–128, Chateau-d'Oex-Vaud, Switzerland, Aug. 2003.

[Macherey & Och$^+$ 01] K. Macherey, F.J. Och, H. Ney: Natural Language Understanding Using Statistical Machine Translation. In *Proc. of the 7th European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 3, pp. 2205–2208, Aalborg, Denmark, Sept. 2001.

[Macherey 98] K. Macherey. Entwicklung und Implementierung von Konfidenzmaßen für kontinuierlich gesprochene Sprache auf der Basis von Wortgraphen. Diploma thesis, RWTH-Aachen, University of Technology, Aachen, Germany, Nov. 1998.

[Matusov & Kanthak$^+$ 05] E. Matusov, S. Kanthak, H. Ney: On the Integration of Speech Recognition and Statistical Machine Translation. In *Proc. of the 9th European Conf. on Speech Communication and Technology (INTERSPEECH)*, pp. 3177–3180, Lisboa, Portugal, Sept. 2005.

[McCallum & Freitag$^+$ 00] A. McCallum, D. Freitag, F. Pereira: Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proc. of the 17th Int. Conf. on Machine Learning (ICML)*, pp. 591–598, Stanford, CA, June 2000.

[Melamed 98] I.D. Melamed. Manual Annotation of Translational Equivalence: The Blinker Project. Technical Report 98-07, Department of Computer and Information Science University of Pennsylvania, Philadelphia, PA, 13 pages, 1998.

[Menzerath & de Lacerda 33] P. Menzerath, A. de Lacerda: *Koartikulation, Steuerung und Lautabgrenzung.* Berlin Bonn (Dümmler), 1933.

[Miller & Bates$^+$ 95] S. Miller, M. Bates, R. Ingria, J. Makhoul, R. Schwartz: Recent Progress in Hidden Understanding Models. In *ARPA Spoken Language Technology Workshop*, pp. 276–280, Austin, Texas, Jan. 1995.

[Miller & Bobrow$^+$ 94a] S. Miller, R. Bobrow, R. Ingria, R. Schwartz: Hidden Understanding Models of Natural Language. In *Proc. of the 32th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 25–32, Las Cruces, New Mexico, USA, June 1994.

[Miller & Bobrow$^+$ 94b] S. Miller, R. Bobrow, R. Schwartz, R. Ingria: Statistical Language Processing using Hidden Understanding Models. In *Proc. of the Human Language Technology Workshop sponsored by ARPA*, pp. 278–282, Plainsboro, NJ, USA, March 1994.

[Minker & Waibel$^+$ 99] W. Minker, A. Waibel, J. Mariani: *Stochastically-Based Semantic Analysis*. Kluwer Academic Publishers, Dordrecht, 1999.

[Minker 98] W. Minker: Stochastic versus rule-based speech understanding for information retrieval. *Speech Communication*, Vol. 25, No. 4, pp. 223–247, 1998.

[Moore & Appelt$^+$ 95] R. Moore, D. Appelt, J. Dowding, J.M. Gawron, D. Moran: Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS. In *ARPA Spoken Language Technology Workshop*, pp. 261–264, Austin, Texas, Jan. 1995.

[Ney & Mergel$^+$ 87] H. Ney, D. Mergel, A. Noll, A. Paeseler: A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 833–836, Dallas, TX, USA, April 1987.

[Ney & Welling$^+$ 98] H. Ney, L. Welling, S. Ortmanns, K. Beulen, F. Wessel: The RWTH Large Vocabulary Continuous Speech Recognition System and Spoken Document Retrieval. In *Proc. of the 24th Annual Conf. of the IEEE Industrial Electronics Society (IECON)*, pp. 2022–2027, Aachen, Germany, Sept. 1998.

[Ney 84] H. Ney: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. *IEEE Trans. on Speech and Audio Processing*, Vol. 32, No. 2, pp. 263–271, April 1984.

[Ney 90] H. Ney. Acoustic Modeling of Phoneme Units for Continuous Speech Recognition. In L. Torres, E. Masgrau, M.A. Lagunas, editors, *Signal Processing V: Theories and Applications, Fifth European Signal Processing Conference*, pp. 65–72. Elsevier Science Publishers B. V., Barcelona, Spain, 1990.

[Och & Ney 03] F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, 2003.

[Och & Tillmann$^+$ 99] F.J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. In *Proc. of the Joint Sigdat Conf. on Empirical Methods in Natural Language Processing and very Large Corpora (EMNLP/VLC)*, pp. 20–28, College Park, Maryland, June 1999.

[Och 02a] F.J. Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.d. thesis, RWTH-Aachen, University of Technology, Aachen, Germany, 2002.

[Och 02b] F.J. Och. Yasmet, 2002. http://www-i6.Informatik.RWTH-Aachen.de/web/Software/index.html.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 160–167, Sapporo, Japan, July 2003.

[Och 07] F.J. Och. personal communication, 2007.

[Oerder & Ney 93] Oerder, H. Ney: Word Graphs: An efficient Interface between continuous-speech Recognition and Language Understanding. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 2, pp. 119–122, Minneapolis, MN, USA, April 1993.

[Ortmanns & Ney 95] S. Ortmanns, H. Ney: An Experimental Study of the Search Space for 20000-Word Speech Recognition. In *Proc. of the 4th European Conf. on Speech Communication and Technology (EUROSPEECH)*, Vol. 2, pp. 901–904, Madrid, Spain, Sept. 1995.

[Papineni & Roukos$^+$ 97] K.A. Papineni, S. Roukos, R.T. Ward: Feature-based Language Understanding. In *Proc. of the 5th European Conf. on Speech Communication and Technology (EUROSPEECH)*, pp. 1435–1438, Rhodes, Greece, Sept. 1997.

[Paul 91] D.B. Paul: Algorithms for an Optimal A^* Search and Linearizing the Search in the Stack Decoder. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 693–696, Toronto, Canada, May 1991.

[Peckham 91] J. Peckham: Speech Understanding and Dialogue over the telephone: an overview of the ESPRIT SUNDIAL project. In *DARPA Speech and Natural Language Workshop*, pp. 14 – 27, Pacific Grove, CA, Feb. 1991.

[Potamianos & Ammicht$^+$ 00] A. Potamianos, E. Ammicht, H.K.J. Kuo: Dialogue Management in the Bell Labs Communicator System. In *Proc. of the 6th Int. Conf. on Spoken Language Processing (ICSLP)*, Vol. 2, pp. 603–606, Beijing, China, Oct. 2000.

[Price 90] P.J. Price: Evaluation of Spoken Language Systems: The ATIS Domain. In *DARPA Speech and Natural Language Workshop*, pp. 91 – 95, Hidden Valley, PA, June 1990.

[Rabiner & Juang 86] L. Rabiner, B.H. Juang: An Introduction to Hidden Markov Models. *IEEE Trans. on Speech and Audio Processing*, Vol. 3, No. 1, pp. 4–16, 1986.

[Rabiner & Schafer 78] L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals.* Prentice-Hall, Englewood Cliffs, NJ, 1978.

[Ratnaparkhi 00] A. Ratnaparkhi: Trainable Methods for Surface Natural Language Generation. In *Proc. of the 1st Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Vol. 1, pp. 194–201, Seattle, WA, May 2000.

[Ruske 82] G. Ruske. Auditory Perception and its Application to Computer Analysis of Speech. In C. Suen, R.D. Mori, editors, *Computer Analysis and Perception*, pp. 2–42. CRC Press, Boca Raton, FL, 1982.

[Sakoe 79] H. Sakoe: Two-Level DP-Matching - A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition. *IEEE Trans. on Speech and Audio Processing*, Vol. 27, pp. 588–595, Dec. 1979.

[Sarikaya & Gao$^+$ 03] R. Sarikaya, Y. Gao, M. Picheny: Word Level Confidence Measurement using Semantic Features. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 604–607, Hong Kong, China, May 2003.

[Schatzmann & Georgila$^+$ 05] J. Schatzmann, K. Georgila, S. Young: Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems. In *Proc. of the 6th SIGdial Workshop on Discourse and Dialogue (SIGDIAL)*, pp. 45–54, Lisbon, Portugal, Sept. 2005.

[Scheffler & Young 02] K. Scheffler, S. Young: Automatic Learning of Dialogue Strategy using Dialogue Simulation and Reinforcement Learning. In *Proc. of the Conf. on Human Language Technology (HLT)*, Vol. 1, pp. 12–18, San Diego, USA, March 2002.

[Scheffler 02] K. Scheffler. *Automatic Design of Spoken Dialogue Systems.* Ph.D. Thesis, St. John's College and Cambridge University Engineering Department, Cambridge, England, 2002.

[Schukat-Talamazzini 95] E.G. Schukat-Talamazzini. *Automatische Spracherkennung – Grundlagen, statistische Modelle und effiziente Algorithmen*, chapter 8, pp. 231–269. Vieweg, 1995.

[Schwartz & Austin 91] R. Schwartz, S. Austin: A Comparison of several Approximate Algorithms for Finding multiple N-best sentence hypotheses. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 701–704, Toronto, Canada, May 1991.

[Seneff 92] S. Seneff: TINA: A Natural Language System for Spoken Language Applications. *Computational Linguistics*, Vol. 18, No. 1, pp. 61–86, 1992.

[Sixtus & Ortmanns 99] A. Sixtus, S. Ortmanns: High Quality Word Graphs Using Forward-Backward Pruning. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 593–596, Phoenix, Arizona, USA, March 1999.

[Strange 83] W. Strange: Dynamic Specifications of Coarticulated Vowels. *Journal Acoust. Soc. Amer.*, Vol. 74, No. 3, 1983.

[Sudoh & Tsukada 05] K. Sudoh, H. Tsukada: Tightly Integrated Spoken Language Understanding using Word-to-Concept Translation. In *Proc. of the 9th European Conf. on Speech Communication and Technology (INTERSPEECH)*, Vol. 1, pp. 429–432, Lisboa, Portugal, Sept. 2005.

[Tjong Kim Sang & Buchholz 00] E. Tjong Kim Sang, S. Buchholz: Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of the 4th Conf. on Computational Language Learning and Workshop on the 2nd Learning Language in Logic (CoNLL-LLL)*, pp. 127–132, Lisbon, Portugal, Sept. 2000.

[Tur & Wright$^+$ 02] G. Tur, J. Wright, A. Gorin, G. Riccardi, D. Hakkani-Tur: Improving Spoken Language Understanding using Word Confusion Networks. In *Proc. of the 7th Int. Conf. on Spoken Language Processing (ICSLP)*, pp. 1137–1140, Denver, Colorado, Sept. 2002.

[Ueffing 06] N. Ueffing. *Word Confidence Measures for Machine Translation.* Ph.d. thesis, RWTH-Aachen, University of Technology, Aachen, Germany, 2006.

[Valtchev & Odell$^+$ 97] V. Valtchev, J.J. Odell, P.C. Woodland, S.J. Young: MMIE Training of Large Vocabulary Recognition Systems. *Speech Communication*, Vol. 22, No. 4, pp. 303–314, Sept. 1997.

[Vidal 97] E. Vidal: Finite-State Speech-to-Speech Translation. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 111–114, Munich, Germany, April 1997.

[Vintsyuk 71] T.K. Vintsyuk: Elementwise Recognition of Continuous Speech Composed of Words from a Specified Dictionary. *Kibernetika*, Vol. 7, pp. 133–143, March 1971.

[Viterbi 67] A. Viterbi: Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. *IEEE Transactions on Information Theory*, Vol. 13, pp. 260–269, 1967.

[Vogel & Ney$^+$ 96] S. Vogel, H. Ney, C. Tillmann: HMM-based Word Alignment in Statistical Translation. In *Proc. of the 16th Int. Conf. on Computational Linguistics (COLING)*, Vol. 2, pp. 836–841, Copenhagen, Denmark, Aug. 1996.

[Wakita 73] H. Wakita: Direct Estimation of the Vocal Tract Shape by Inverse Filtering of Acoustic Speech Waveform. *IEEE Trans. on Audio and Electroacoustics*, Vol. 21, No. 5, pp. 417–427, Oct. 1973.

[Ward & Issar 95] W. Ward, S. Issar: The CMU ATIS System. In *ARPA Spoken Language Technology Workshop*, pp. 249–251, Austin, Texas, Jan. 1995.

[Wessel & Macherey$^+$ 98] F. Wessel, K. Macherey, R. Schlüter: Using Word Probabilities as Confidence Measures. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 225–228, Seattle, Washington, USA, May 1998.

[Wessel 02] F. Wessel. *Word Posterior Probabilities for Large Vocabulary Continuous Speech Recognition.* Ph.D. Thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, 2002.

[Williams & Poupart$^+$ 05] J.D. Williams, P. Poupart, S. Young: Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management. In *Proc. of the 6th SIGdial Workshop on Discourse and Dialogue (SIGDIAL)*, pp. 25–34, Lisbon, Portugal, Sept. 2005.

[Young 93] S.J. Young. *HTK: Hidden Markov Model Toolkit V1.4. User Manual.* Cambridge, UK, Feb. 1993.

[Young 94] S.R. Young: Detecting Misrecognitions and Out-Of-Vocabulary Words. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 2, pp. 21–24, Adelaide, Australia, April 1994.

[Young 00] S. Young: Probabilistic Methods in Spoken Dialogue Systems. *Philosophical Trans.: Mathematical, Physical and Engineering Sciences*, Vol. 358, No. 1769, pp. 1389–1402, April 2000.

[Young 02] S. Young. The Statistical Approach to the Design of Spoken Dialogue Systems. Technical Report 433, Cambridge University Engineering Department, 25 pages, Sept. 2002.

[Zhou & Gao$^+$ 02] B. Zhou, Y. Gao, J. Sorensen, Z. Diao, M. Picheny: Statistical Natural Language Generation for Speech-to-Speech Machine Translation Systems. In *Proc. of the 7th Int. Conf. on Spoken Language Processing (ICSLP)*, Vol. 3, pp. 1897–1900, Denver, Colorado, Sept. 2002.

[Zue & Glass$^+$ 91] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, S. Seneff: Integration of Speech Recognition and Natural Language Processing in the MIT VOYAGER System. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 713–716, May 1991.

Resume

Dipl.-Inform. (M.S.) Klaus Macherey

Personal Data

date of birth:	March 29th, 1973
place of birth:	Düren-Birkesdorf
nationality:	German

Education

2001 - 2006	Research Assistant and Ph.D. student at Computer Science Department of RWTH Aachen University (Prof. Dr.-Ing. Hermann Ney)
2000	Basic military service (as regulated by law)
since 1999	Research Assistant and Ph.D. student at Computer Science Department of RWTH Aachen University (Prof. Dr.-Ing. Hermann Ney)
1999	Diploma's degree (M.S.) in Computer Science.
1992 – 1999	Studies of computer science at RWTH Aachen University. Minor field of study: physical chemistry, thermodynamics. Diploma's thesis title: "Development and Implementation of Confidence Measures for Automatic Speech Recognition based on Word Graphs."
06/1992	Final secondary-school examinations (Abitur). Awarded by the Chemical Industry.
1989 – 1992	Secondary school Burgau Gymnasium Düren (Abitur).
1991	Qualified for the 3rd of four rounds of the International Chemistry Olympiad (IChO).
1979 – 1982	Primary school.

Experience

since 03/2006	Research scientist at Google Inc. Mountain View, California
01/2005 – 09/2005	Internship at Google Mountain View, CA, in the *Rosetta* machine translation group. Participated with the *Rosetta* team in the NIST-2005 machine translation evaluation.
07/2003 – 08/2002	Participant in the Johns Hopkins CLSP 2003 workshop (Semantic Analysis over Sparse Data).
07/2002 – 09/2002	Summer student at T.J. Watson Research Center in Yorktown Heights, NY.
04/2001 – 12/2002	Scientific consultant of Aixplain Corporation. Development and implementation of ASR systems for industrial partners. Development of spoken dialogue systems.
05/1999 – 03/2006	Research Assistant and Ph.D. student at RWTH Aachen University. Responsibilities comprise research in statistical pattern recognition with the focus on natural language processing, especially natural language understanding, spoken dialogue systems, confidence measures, project management, and teaching activities (supervision of diploma theses, organization and conducting of seminars and laboratory courses for undergraduate students).
11/1996 – 04/1999	Undergraduate scientist in the speech recognition group of Prof. Dr.-Ing. H. Ney; domain: *Confidence Measures*. Tutor for several lectures in computer science.

Awards

1992	Award from the German Chamber of Chemical Industry.