

Bachelor Thesis

Modular Grammars for Speech Recognition in Ontology-Based Dialogue Systems

Hanne Marie Kosinowski

Matriculation Number 2519286

August 29, 2010

Saarland University
Faculty of Computational Linguistics

Thesis supervisor: Prof. Dr. Dr. h. c. mult. Wolfgang Wahlster

Advisor: Sebastian Germesin, M. Sc.

Reviewer: Prof. Dr. Dr. h. c. mult. Wolfgang Wahlster
Dr. Tilman Becker

Declaration of Academic Honesty

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Declaration of Academic Honesty

I hereby declare that the work submitted in this thesis is my own, and any work that is not my own has been quoted and acknowledged in the References section.

Saarbrücken, August 29, 2010

Hanne Marie Kosinowski

Abstract

In the present work we show an approach for writing modular grammars for an ontology-based dialogue system. They are written for the ambient intelligence bathroom environment of the IKS project. After giving a general introduction to the field of speech recognition, language models and formal grammars, we introduce the grammar format and its syntactic requirements. In a series of examples we present the structure of our grammars and how they were implemented. We present results from a user evaluation which was used for determining how many of our utterances would be used by others. The results from the evaluation led to a modification of the grammars to cover a larger percentage of the participants' answers. Furthermore, we introduce a basic framework for integrating a natural language understanding component into the grammars.

Contents

1	Introduction	1
1.1	Related Work	3
1.2	Thesis Overview	4
2	Speech Recognition	5
2.1	History	5
2.2	Typical Structure of Speech Recognition Systems	7
2.3	Language Models	8
2.3.1	Statistical Approach	8
2.3.2	Grammars	11
2.4	Current Systems	13
2.5	Chapter Summary	15
3	Speech Recognition Grammars	17
3.1	Chomsky Hierarchy	17
3.2	Developing Speech Recognition Grammars	20
3.2.1	Grammar Format	21
3.2.2	File Structure	22
3.3	Chapter Summary	24
4	Implementation	27
4.1	Modular Grammars	27
4.2	Situations for the AmI Bathroom	29
4.2.1	Situation 1	29
4.2.2	Situation 6	30
4.2.3	Situation 11	30
4.3	Grammar Creation	30
4.4	Testing	33
4.5	Requirements	35
4.6	Grammar Revision	37
4.7	Chapter Summary	38

X	Contents	
5	Evaluation	39
5.1	Method	40
5.2	Results	40
5.3	Discussion	44
5.4	Future Evaluations	48
5.5	Chapter Summary	48
6	Discussion and Future Work	49
6.1	Discussion	49
6.2	Future Work	50
	References	53
	Appendices	57
A	Empty .grm file	59
B	Evaluation Questionnaire	61
C	Grammars	63
C.1	General	63
C.2	Time	66
C.3	Situation 1	67
C.3.1	weather	67
C.3.2	event recommendation	69
C.3.3	ticket ordering	72
C.4	Situation 6	73
C.4.1	music	73
C.4.2	news	76
C.5	Situation 11	78
C.5.1	radio	79
D	Natural Language Understanding	81

Introduction

One of the working areas of computational linguistics is speech and language processing. It contains all topics related to processing human speech with a computer, e. g. speech recognition, speech synthesis, part of speech tagging, machine translation and natural language understanding.

In this work, we will focus on speech recognition and the characteristics of such systems. Speech recognition systems (e. g. Sphinx-4 [WALKER ET AL., 2004] and Nuance Recognizer [NUANCE, 2009]) are computer systems that map spoken utterances to strings of words. To achieve this transformation of recorded audio input a to written representation, several smaller processes are applied sequentially: The first step in the processing sequence is to transform the recorded audio input into frequency spectrums by means of a Fast Fourier Transformation [COOLEY & TUKEY, 1965]. The spectrums are passed on to a Hidden Markov Model which is a statistical model determining the most probable phoneme sequence [RABINER, 1989, RABINER & JUANG, 1993]. This phoneme sequence is forwarded on to a language model. There are two different models whose usage mainly depends on the task the speech recognition is applied for.

One of the language models is the statistical n-gram approach. For this, a large corpus is required which provides a collection of texts where probabilities for word sequences of length n are extracted from. These probabilities are used to calculate a probability of an entire sentence by multiplying the probabilities of all n-grams contained in the sentence. N-grams are often used in open-domain speaker-dependent dictation tasks.

The second language model consists of rule-based grammars. Their core features are hand-written rules which define the acceptable utterances for a system exactly. In this work, we focus on grammars, as they form the better suited language model in closed-domain speaker-independent dialogue systems.

In this thesis, we create grammars for an ontology-based dialogue system, embedded into an interactive bathroom. We choose the Ontology-based Dialogue Platform (ODP) for the implementation of our grammars

[PFALZGRAF ET AL., 2008]. as it provides state-of-the-art speech recognition based on the Nuance Recognizer and enables the developer to integrate natural language understanding. We extend the concept of grammar based language models by introducing a modular grammar architecture. This approach, previously suggested by Lumenvox [LUMENVOX, 2010] and Giraudo [GIRAUDO & BAGGIA, 2009] (see section 1.1), leads to a higher precision in recognition and a low word error rate, as the grammar modules are matched to the corresponding dialogue situations. In contrast to many other speech recognition systems with rule-based grammars which have one large grammar file containing all the necessary information, we split up the large file into several smaller files.

We design the grammars' architecture such that one main file contains all general information (e. g., names, places, temporal adverbs, general sentence constructs) and smaller files contain more specific information. These files can be activated when required.

The bathroom in which the grammars will be integrated is one of the use cases for the EU project IKS¹. For the bathroom, eleven situations have been defined to illustrate the project software's abilities. We use these situations for developing our grammars. Due to time constraints we will focus on the three highest-ranked situations of all eleven situations. For each of the six we write a grammar addressing one topic only: requesting weather information, asking for an event recommendation, ordering tickets for an event, navigating through a music collection, choosing a news channel and changing a radio station.

The six grammars are the result of a process whose first step is to create a state diagram for each part illustrating a possible course of a system-user dialogue. For each user's turn in the dialogue we devise semantically similar utterances. They are generalized with variables which stand for parts of utterances ("I would like to", "please show me") or collections of items (event locations, cities, numbers). The utterances containing the variables are then transferred to the appropriate grammar format. In our case, it is the Nuance Grammar Specification Language (GSL) format which is defined in the Nuance Grammar Developer's Guide [NUANCE, 2003] based on XML.

A grammar for speech recognition is often a mixture of regular and context-free grammar rules. Chomsky defines a classification of formal grammars [CHOMSKY, 1956, CHOMSKY, 1963] and arranges them in a hierarchy according to their complexity. In the hierarchy, regular grammars are the least complex type, followed by context-free grammars. More complex rules are not required for our dialogue system, as the complexity of the English language is at least context-free or even regular [YNGVE, 1960, CHOMSKY & MILLER, 1963].

After we have completed the grammar files, we proceed to testing them with two tools provided by the Nuance Recognizer. One determines whether a

¹ Interactive Knowledge Stack, see <http://www.iks-project.eu/>

written input sentence is covered by the grammar. This is useful for verifying whether the sentences we originally devised are indeed covered by the grammar. After a first test run, the grammars are modified slightly and we receive coverages of 87.56% on average. The other tool allows the generation of a specific number of sentences. With this, we are able to establish whether the grammars produce nonsensical output which was not the case.

In addition, we perform an evaluation of our grammars which is designed to determine how much of the sentences we have devised for the six situations are used by other persons. For this, we design an interview in which we describe a certain situation and the participants answer to our prompts.

While analysing the results of the evaluation, we detect that less than 2% of the participants' utterances are covered by our grammar. Thanks to the grammars' structure, we are able to achieve an improvement in the coverage by modifying few rules only.

In the future, we will implement all situations designed for the bathroom and add an NLU component to all of them. Furthermore, we will consider further findings from the evaluation programme and incorporate one- and two-word commands into the grammars, as they were the most commonly used utterances.

1.1 Related Work

The general aim of writing a grammar for a language is to cover all possible grammatical constructs. In the English Resource grammar, a head-driven phrase structure grammar developed by Copestake et al [COPESTAKE & FLICKINGER, 2000], this aim has been achieved. Although parts of this all-comprising grammar have been extracted for other projects, we choose to create our own grammars for two reasons: extracting only relevant parts from a grammar the size of ERG is difficult and time consuming; in addition, ERG does not support a modular approach which is the essential motivation for this thesis.

On the other hand, several companies and papers explicitly suggest writing small grammars with a modular design as we do in this thesis.

Lumenvox² proposes to "build modular grammars" and enable required modules only at any time in the recognition process in a publication on grammar design [LUMENVOX, 2010]. The modular architecture leads to better accuracy when only relevant modules are activated as there are less equal sounding words for one input. Moreover, small grammar files can be modified easier than one big file and errors can be detected faster.

A dialogue system grammar presented by Giraudo [GIRAUDO & BAGGIA, 2009] for EVALITA 2009³ which was developed

² <http://www.lumenvox.com>

³ An evaluation campaign of Natural Language Processing tools for Italian.

for the Loquendo Automatic Speech Recognition software, defines grammars for different components of an utterance. Several of the simple grammars are combined into a *complex grammar* which comprises all grammars for handling complete user utterances.

The Microsoft Speech Application software development kit (SASDK) manual suggests to build modular grammars for each dialogue turn which improves recognition performance and ensures correct handling of user responses. The SASDK predefines two grammar files where one contains general concepts like times, dates and numbers while the other one recognizes dual-tone multi-frequency keypresses.

1.2 Thesis Overview

In chapter 2, we present a historical overview of speech recognition in general and describe a typical structure of a speech recognition system. We introduce the two most common language models: the statistical n-gram approach and the grammar-based model. We describe two state-of-the-art systems, the open-source SPHINX and the commercial Nuance Recognizer.

We introduce formal grammars and their structure in chapter 3. We compare formal grammars to grammars used in speech recognition applications and present the syntax as well as the file structure of the format we used for developing our grammars.

In chapter 4, we show how the grammar files are created and which steps are necessary during the development process. We provide examples for rules and the environment in which the grammars will be used. Furthermore, we define requirements for the grammars which should be considered when developing further grammars for the same environment. Challenges in the creation of the grammars and testing process are also addressed.

Chapter 5 is dedicated to an evaluation we performed with our grammars. We use the evaluation data to analyse how many of the participants' utterances are covered by our grammars. We also demonstrate how to extend the grammars with few rules to include a considerable amount of the utterances collected in the evaluation.

In chapter 6, we indicate possible further developments of the grammars and present an example of including natural language processing in the grammars.

The appendices contain an empty `grm` file (see appendix A), the questionnaire used in the evaluation (appendix B), all grammar files created for this thesis (appendix C) and examples for NLU structures for utterances from two different situations (appendix D).

Speech Recognition

Speech recognition systems were first introduced in the 1940s and have improved constantly since then. Transformation from the spoken input to the corresponding transcript is performed by several steps. One of the important steps in each speech recognition system is the language model which is responsible for assigning a string of words to a sequence of phonemes. Two prevalent methods have to be distinguished: a statistical n-gram approach and a rule-based grammar.

In this chapter we take a look at the historical development of speech recognition systems before we describe subparts of a typical setup. Furthermore, we explain the features of different language models in detail and end the chapter by presenting state-of-the-art recognition systems as well as their main characteristics.

2.1 History

The first application in speech recognition can be traced back to the 1920s. Back then, a small robot dog called Rex was sold which seemed to respond to its own name. This feature was realized using a simple mechanism: A spring was released at 500 Hz acoustic frequency which corresponds to the first formant's¹ energy for "e" in "Rex". Subsequently, the dog moved and appeared to react to its own name [JURAFSKY & MARTIN, 2008, Chapter 9].

In the 1940s and '50s, digit recognizers were first developed which are still one of the easiest applications to realise, as digits are uttered separately. A single digit is recognized more easily than a string of words as its pronunciation is not modified by surrounding words and sounds. Furthermore, the vocabulary is limited to ten words which leads to further increase in recognition performance (we illustrate the reasons for better performance with smaller

¹ Formants describe the eigenfrequencies of a produced sound. The first formant represents the lowest eigenfrequency.

vocabulary in section 3.2). A digit recognizer published by Bell Labs in 1952 achieved an accuracy of 97 to 99% using speaker-dependent sound patterns [DAVIS ET AL., 1952].

The first phoneme² recognizer dates back to 1959 and was developed by Fry and Denes [FRY, 1959, DENES, 1959]. Their system used phoneme transition probabilities as constraint method for the recognizer. Although it was not successful at that time [SCHUKAT-TALAMAZZINI, 1995], this technique has found its way into modern speech recognition systems where it is applied in Hidden Markov Models (see section 2.2).

In the 1960s and '70s, fundamental techniques were developed for and established in speech recognition systems. One of the techniques allowed the consideration of different speaking tempi in the recognition process. This became necessary as the systems were no longer restricted to a single speaker. As a result, the recorded patterns had to be warped for matching the stored patterns.

In 1913, the Russian mathematician Markov proposed to apply a model for predicting the respective following letter in a written text [MARKOV, 1913]. Baum and colleagues converted this Markov Model into Hidden Markov Models (HMM) in the late 1960s for various prediction tasks [BAUM & PETRIE, 1966, BAUM & EAGON, 1967]. HMMs were not much in use until the 1970s, but since they proved to be efficient for a range of tasks, they quickly became one of the most used algorithms in speech recognition. Baker was the first to use HMMs in this field [BAKER, 1975]. He later founded Dragon Systems, a speech recognition company which now belongs to Nuance Communications.

At the IBM Thomas J Watson Research Center, three researchers namely, Jelinek, Mercer and Bahl, used HMMs in a speech recognizer at roughly the same time as Baker did. In the following years, IBM supported and propagated the use of statistical methods in speech recognition and hence is considered to be one of the companies setting the standard for using statistical approaches in various areas of natural language processing.

From the 1970s onwards, the Advanced Research Projects Agency (ARPA)³ initiated a project to develop a system which aimed at improving the recognition rate in a system with few speakers and achieving a semantic error rate⁴ of less than 10% which was low in comparison with other projects at that

² A phoneme is the smallest meaningful unit of sound in a language. Every sound that distinguishes two words which only differ in one sound is a phoneme. An example: “cat” (/kæt/) and “mat” (/mæt/). The words are the same apart from the initial sound. Thus, the sounds /k/ and /m/ are phonemes in the English language. [I. P. A., 1999]

³ ARPA is an agency belonging to the US Department of Defense. The name was changed in 1972 to Defense Advanced Research Projects Agency (DARPA), in 1993 changed back to ARPA in 1993 and as a last change, back again to DARPA in 1996.

⁴ The semantic error rate indicates the amount of wrongly assigned semantic structures by the parser while analysing the input utterance [STAHL ET AL., 1996].

time. When the systems that had been developed were evaluated, the best performance was delivered by a system relying on Baker's DRAGON.

In the mid-1980s, DARPA focused on speech recognition of read text. In this case, the speech rate is reduced and the vocabulary is limited in comparison to conversational speech. These factors lead to a good speech recognizer performance and a low word error rate⁵ (WER).

Over the years, DARPA's projects used bigger vocabularies: the first ones contained 1,000 words, the next 5,000 words. More recent projects use approximately 60,000 different words, which corresponds to the amount of words used in everyday language. The vocabularies increased in size as the aim was and still is to recognize correctly as many words as possible. Also, DARPA's projects evolved from read text to more natural domains, e. g. news, interviews or telephone conversations [JURAFSKY & MARTIN, 2008, Chapter 9].

At Carnegie Mellon University (CMU), Lowerre developed the HARP system in the mid '70s which defined an architecture still in use for the SPHINX system and others today (see section 2.4) [SCHUKAT-TALAMAZZINI, 1995, LOWERRE, 1976]. The HARP system fulfilled most goals that DARPA had defined for their project and even outperformed the DARPA system in some aspects [SCHUKAT-TALAMAZZINI, 1995, p. 12].

More recently, speech recognition can be found in domain-dependent, speaker-independent dialogue systems, dictation software, as well as command and control systems which are designed to work reliably in many different kinds of environments.

In the upcoming section, we present a detailed insight into the typical structure of a speech recognition system.

2.2 Typical Structure of Speech Recognition Systems

The speech recognition process can be seen as a sequence of smaller sub-processes which take the spoken utterance as input and produce a written or computer understandable representation as output.

In a first step, the input is recorded and split up into a spectral representation showing the frequencies for each sound. This can be achieved by, e. g., a Fast Fourier Transformation (FFT)⁶, or by means of the mel cepstrum⁷ with a sample rate of 100Hz. Every sound fragment is transformed into a form that

⁵ The word error rate describes the percentage of words which have not been understood correctly. It is a common measure for comparing different speech recognition systems.

⁶ The FFT is an algorithm for efficiently calculating the values of a discrete Fourier transformation. For a detailed explanation see [COOLEY & TUKEY, 1965].

⁷ A mel is a unit of pitch which is computed from raw acoustic frequencies. For more details on the calculation, see [STEVENS ET AL., 1937]. A cepstrum is an inverse discrete fourier transform which simulates the human hearing above certain fre-

features can be extracted from more easily than the recorded frequency signal. They are written into vectors serving as input for a Hidden Markov Model which uses them for determining the most likely phoneme sequence.

Generally speaking, an HMM can be represented by a finite state automaton where each transition is labelled with probabilities. Unlike the Markov Model, the sequence of events leading to a result is hidden in an HMM. An HMM for speech recognition is trained on a set of words which give probabilities for certain phoneme sequences. During the speech recognition process, phoneme sequences are used as input and the most likely word for this phoneme sequence is returned. See [RABINER, 1989, RABINER & JUANG, 1993, HUANG ET AL., 2001, JURAFSKY & MARTIN, 2008] for details on HMMs and their usage in speech recognition.

A word or a sequence of words obtained from the HMM's calculations is then passed on to a language model for further processing.

2.3 Language Models

The choice of a language model for a speech recognition system mainly depends on the system's purpose and task. There exist two main approaches either of which is used in most speech recognition systems: n-gram models and (hand-written) grammars [HUANG ET AL., 2001, Chapter 11]. Although there is an approach by Wang et al. [WANG ET AL., 2000] who combine the statistical approach and grammars, we choose the grammar-based model only. Nevertheless, a hybrid model can be adopted in the future (see chapter 6).

The recognizer's performance depends on several constraints which can partly be influenced by the composition of the language model: the input's quality⁸; the continuity of speech, either discrete or continuous; the grammar's accuracy; the lexicon's size and the amount of data the statistical model has been created from. We address these constraints in the discussion of the two language models below.

2.3.1 Statistical Approach Using N-Gram Models

The statistical approach relies on large text corpora⁹ for predicting the most likely word sequence or the most likely word to follow a certain sequence. The prediction is based on probabilities for word chains extracted from the corpus which are called n-grams. The n describes the length of the sequence, where one word is called a unigram, two words form a bigram (or 2-gram) and so on.

quencies. See [JURAFSKY & MARTIN, 2008, pp. 334], [TAYLOR, 2009] for more information.

⁸ i. e. the amount of noise during the recording

⁹ A corpus is a large collection of texts.

The most used n -grams in language models are bi- and trigrams, as the same sequence of two or three words is more likely to occur several times in a text which results in a higher relative frequency for the sequence than for $n > 3$. For example: The sequence “the man” can be found considerably more often in a text corpus than “the man loved Mary”. Thus, the probability for “the man” occurring in a text will be higher than for “the man loved Mary”. Consequently, we state that a higher n leads to less n -grams of the same kind in a text as opposed to n -grams with a small n which have high probabilities that can be calculated with more easily [JELINEK, 1997, JURAFSKY & MARTIN, 2008].

Over the years, several big corpora which can be used for recognition tasks have been accumulated. Among the first ones is the Brown Corpus, which was created in the 1960s in the USA and contains several kinds of written text (including, e. g., fiction, newspaper articles, learning books). It lists about one million words and 61,805 different word tokens [JURAFSKY & MARTIN, 2008, p. 120]. The Corpus of Contemporary American English was started in 1990 and it is still updated once or twice a year. It is the biggest available corpus of American English and is composed of over 410 million words and contains more than 60,000 word tokens [DAVIES, 2009]. The biggest corpus currently available has been created by Google in 2006 and lists 1 trillion words¹⁰ with 13.5 million unique word tokens. It was developed for research purposes automatically, providing n -grams from $n=1$ to $n=5$ and has been generated from publicly available web pages¹¹.

Bigram	Trigram
<s>Tim	<s>Tim drove
Tim drove	Tim drove to
drove to	drove to university
to university	to university </s>
university </s>	

Table 2.1. Bi- and trigrams for the sentence “Tim drove to university”. “<s>” and “</s>” stand for the beginning and end of a sentence respectively.

As we show in table 2.1, n -grams take the beginning and end of the sentence into account, too.

The probability of a sentence is calculated from the given words and is based on conditional probabilities¹². If we consider bigrams, the conditional

¹⁰ 1 trillion = 10^{12}

¹¹ The Google n -gram corpus can be accessed at <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13>

¹² Conditional probabilities are calculated for two statistically dependent events A and B : $P(B | A) = \frac{P(A \wedge B)}{P(A)}$. We calculate the probability of event B given that we have seen event A .

probability of a word (“drove” from the example above) is calculated by using one preceding word (“Tim”):

$$P(\text{drove}) = P(\text{drove} \mid \text{Tim}). \quad (2.1)$$

For trigram calculations, two preceding words serve as the condition:

$$P(\text{drove}) = P(\text{drove} \mid \langle s \rangle \text{Tim}). \quad (2.2)$$

The general formula for calculating the probability of any word w_n in an N -gram w_1^{n-1} ($w_1 w_2 \dots w_{n-1}$) is

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1}). \quad (2.3)$$

w_n is the n th word after a sequence of words which has the length $n - 1$ and N is the length of considered n -grams [JURAFSKY & MARTIN, 2008, Chapter 4]. An example: We consider 3-grams and the word sequence $w_1 w_2 w_3 w_4$. The conditional probability of w_5 occurring after $w_1 w_2 w_3 w_4$ is roughly equivalent to the conditional probability of w_5 given $w_2 w_3 w_4$:

$$P(w_5 \mid w_1^4) \approx P(w_5 \mid w_2^4). \quad (2.4)$$

More accurate values can be calculated is by means of a maximum likelihood estimation¹³ (MLE). If we consider bigrams for the sake of clarity, the formula with MLE looks like this:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}, \quad (2.5)$$

where $C(w_{n-1}w_n)$ denotes the number of bigrams in which word w_{n-1} occurs directly before w_n and $C(w_{n-1})$ the number of the unigram w_{n-1} [JURAFSKY & MARTIN, 2008, Chapter 4].

This formula can be used for calculating the probability of an entire sentence: all conditional probabilities are multiplied, resulting in a product chain based on bigrams:

$$\begin{aligned} P(\text{Tim drove to university}) &= P(\text{Tim} \mid \langle s \rangle) * P(\text{drove} \mid \text{Tim}) * P(\text{to} \mid \text{drove}) \\ &\quad P(\text{university} \mid \text{to}) * P(\langle s \rangle \mid \text{university}). \end{aligned} \quad (2.6)$$

The probabilities for the calculation are obtained from a corpus which is used for extracting n -grams¹⁴ of the desired length (two in our case). When

¹³ For this, n -gram counts are normalized: The probability of an n -gram is the result from dividing the number of its occurrences by the total number of n -1-grams in the corpus. All probabilities of a word in all possible n -grams in the corpus lie between zero and one and sum to one.

¹⁴ Many of the corpora mentioned above already provide n -grams of various lengths which can be used without having to extract them from the corpora.

calculating the probability of a sentence, the probability of a particular n-gram, e. g., “Tim drove”, is taken from the list of n-grams and multiplied with the probabilities for the other n-grams in the designated sentence.

Both a large training corpus and a large variety in words, i. e. a large vocabulary, are desirable for a statistical n-gram model. The large vocabulary allows the recognition of more words, while a small vocabulary has other advantages as there are less words sounding equal which the recognizer could confuse. However, the recognizer’s performance immediately depends on the size of the corpus the n-grams were extracted from. The bigger the corpus, the more accurate calculations can be performed with the n-gram model as more accurate probabilities for the n-grams have been determined than with a small corpus. Furthermore, a large corpus allows the utilization of longer n-grams.

A statistical n-gram model is considered most often for open-domain, speaker-dependent dictation systems. With speaker dependency, good recognition rates can be achieved as the system is trained on the intonation and pronunciation of one speaker. Moreover, the n-gram model is suitable for dictation systems because the many different word sequences that can occur would be difficult to cover with a grammar based approach. With n-grams, the number of sentences is not as limited as with a grammar based approach where only predefined sentences are accepted.

N-grams are also often used in recognition tasks for isolated words (i. e. digit recognizer, train information system). On one hand, this is difficult to be modelled with grammars and on the other hand, the n-gram model can likewise be trained on unigrams instead of bi- or trigrams.

2.3.2 Grammars

In the following section, we present the other language model with relies on rules to specify which utterances are accepted by the speech recognizer: rule-based grammars.

Grammars are mainly used for speech recognizers in dialogue system applications. With them, we can explicitly specify sentences for the speech recognizer. This is interesting for dialogue systems as most of them are created for a particular domain (domain-dependent) where users utter specific commands and sentences to receive information. The set of commands and utterances is finite and unlike in a dictation system, not a wide range of sentences has to be recognized.

Furthermore, the usage of grammars and an appropriate format (VoiceXML¹⁵, Nuance Grammar format) allows the integration of natural language understanding which is necessary if the grammar is integrated into a dialogue system. The commands uttered by a user have to be linked to a representation which the system can process (see chapter 6).

¹⁵ A standard for voice dialogues between humans and computers based on XML.

Grammar rules used for a language model are usually handwritten because variables and structures required for the rules can rarely be extracted automatically from written utterances. Wang et al [WANG & JU, 2004] have shown a method for creating grammar rules for alphanumeric concepts from regular expressions. All other rules so far have to be created manually.

The usual procedure for writing a speech recognition grammar is to analyse the sentences which are likely to be produced. These sentences serve as starting point for creating grammar rules. The sentences should fulfil the “closed vocabulary assumption” [JURAFSKY & MARTIN, 2008, p.129] which postulates that a grammar contains all words and utterances which can be used for a certain context. Consequently, a word or utterance which is not part of the defined sentences is not recognized by the grammar, although the closed vocabulary assumption states that such exceptions should not exist.

Other aspects to consider during grammar development are overspecification and underspecification. Overspecification means that a grammar generates ungrammatical and incorrect sentences which the user will not use or which do not make sense. By including these sentences, the system produces more errors: Correct sentences can be recognized wrongly which leads to misinterpretation of the utterances. Underspecification describes the fact that there are utterances that a user is very likely to articulate, but the grammar does not cover the utterance.

Most grammar specifications allow the creation of variables which entails clear and unambiguous rule structures. A variable stands for multiple items of the same kind, e. g., several cities which all can take a specific slot: “I want to go to CITY” where CITY can be any of an arbitrary collection of cities.

For determining whether the grammar rules correspond to what a human user would say, at least one evaluation is performed during the development process. With the results, the recognizer grammars are modified. However, a trade-off has to be made between the effort to adapt the grammar and the improvement of the speech recognizer’s performance. This is due to the amount of sentences collected in the evaluation as there are many different types of speakers and ways to phrase an utterance.

In comparison with the statistical model, the grammars show some disadvantages. As pointed out by Huang et al. [HUANG ET AL., 2001, Chapter 11], the interaction with grammar-based speech recognizers may seem brittle and rigid to a user. This impression can occur when the grammar is too small so that utterances are rarely recognized correctly. Additionally, the interaction can appear inflexible as a grammar-based dialogue system defines a strict course of actions which needs to be taken in order for the system to understand the input. Only certain utterances are accepted at a particular point in the dialogue. Most dialogue systems have difficulties with interpreting non-literal language, such as metaphors and other figures of speech (e. g., metonymy, personification or simile) which are difficult to include in a grammar.

Generally speaking, a speech recognizer's performance increases with decreasing grammar size. Large grammars are more likely to contain similar sounding words and utterances which can easily be confused [LUMENVOX, 2010]. A grammar provides high accuracy and low latency but is not as flexible as the statistical n-gram model. Grammar-based speech recognizers are well suited for command and control applications with a narrow and well defined vocabulary [HUANG ET AL., 2001, pp. 581].

A more detailed explanation of speech recognition grammars, their use and development as well as their differences to conventional context-free and regular grammars is given in chapter 3.

2.4 Current Systems

When looking at current speech recognition systems, two solutions with different language models have to be considered: One of them is the open-source software SPHINX¹⁶ developed by Carnegie Mellon University (CMU). The project was initiated in 1986 [LEE & REDDY, 1988] and is still developed and improved. When it was first released, it was the first speaker-independent, continuous-speech recognizer using a large vocabulary. All versions are still available, though only for the latest version (SPHINX-4) new releases are issued by CMU.

Sphinx-2 uses an n-gram language model and a variant of the standard HMMs: semicontinuous HMMs¹⁷ [HUANG ET AL., 1992]. In Version 3 in 1996, continuous HMMs are used, an n-gram model is used as language model and only one acoustic format was implemented [GERMESIN, 2006].

Sphinx-4 was developed and released in 2004, allowing different language models. The implementation permits a modular architecture, which means that Sphinx's components can be adjusted to individual needs.

The architecture of Sphinx-4 consists of three parts which are depicted in figure 2.1: the FrontEnd, the Linguist and the Decoder. The FrontEnd analyses the audio input and prepares it for the Decoder. In the Linguist, different language models, acoustic models and dictionaries can be specified. The language model component supports several different models ranging from finite-state transducer grammars over statistical based grammars to statistical trigram models. The Decoder then uses both outputs, from FrontEnd and Linguist, for generating possible results from the input. By allowing the use of different language models, Sphinx-4 can be adapted successfully to a variety of tasks. Moreover, the modularity allows the implementation of the best suited language model for a particular task [WALKER ET AL., 2004].

¹⁶ <http://cmusphinx.sourceforge.net>

¹⁷ In a semicontinuous HMM or semi-HMM, each hidden state corresponds to multiple output observations [JURAFSKY & MARTIN, 2008, pp. 860].

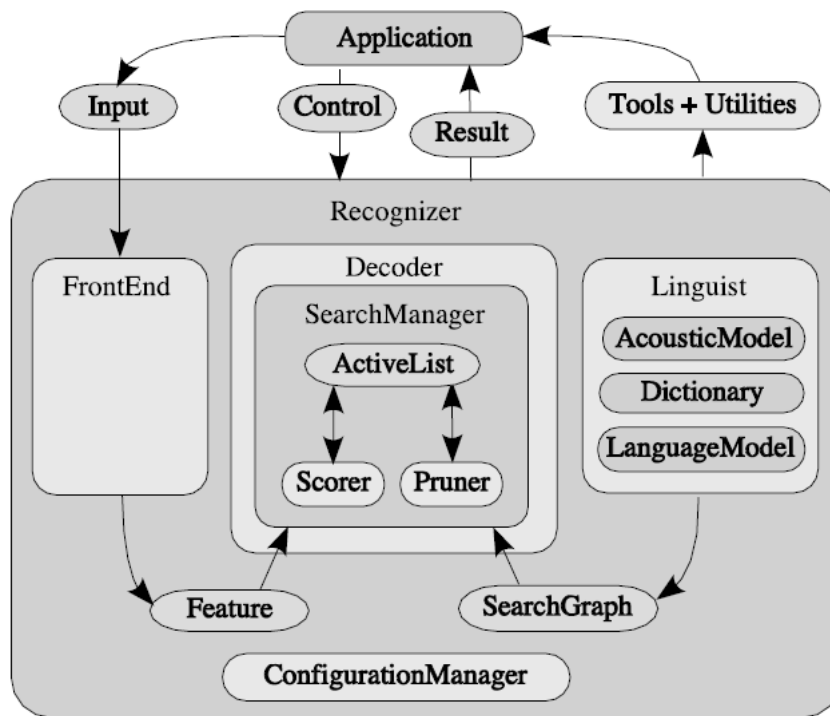


Fig. 2.1. The SPHINX-4 architecture with the central processing parts: the FrontEnd, the Linguist and the Decoder.

The other recognizer is the commercial Nuance Recogniser v9 by Nuance Communications¹⁸ which utilises a grammar-based language model. This system serves as foundation for the grammars developed in this thesis. The structure and functioning is elaborated in chapter 3.

The Nuance Recogniser provides an internal pronunciation dictionary containing approximately 100,000 words which can be extended when necessary. The grammar format is based on VoiceXML, a World Wide Web Consortium¹⁹ (W3C) standard. The recognizer follows several other W3C standards, such as SRGS²⁰, SISR²¹, NLSML²² and EMMA²³.

As a result from long research and development time, the Nuance Recogniser is one of the most accurately working systems currently available. Nuance claims that its software achieves a good performance even when the

¹⁸ <http://www.nuance.com>

¹⁹ <http://www.w3.org>

²⁰ Speech Recognition Grammar Specification

²¹ Semantic Interpretation for Speech Recognition

²² Natural Language Semantic Markup Language

²³ Extensible Multi-Modal Annotation

input contains background noise. The Recogniser is available in 44 different languages and dialects, an amount which has not been met by other companies. The software automatically adapts to local variants of a dialect or telephone connection characteristics (e.g., statics) for a more accurate performance. Provided the grammar supports these kinds of utterances, the vast amount of training data for the internal statistical language model allow more natural input from a user and the system can handle sentences where more than one answer is given in one sentence [NUANCE, 2009].

2.5 Chapter Summary

In this chapter, we presented an overview of significant developments which introduced lasting technologies to speech recognition systems: Hidden Markov Models, the statistical n-gram approach and speech recognition grammars have proven successful over the last forty years. An HMM is applied for calculating the most likely phoneme sequence from an input and its output is passed on to one of two language models: the n-gram approach relies on a statistical analysis of a corpus to achieve a small WER. The grammar-based approach uses hand-written rules to match a user's utterances. For a good performance, the grammar should be small but nevertheless cover all possible utterances for a certain context. Current systems use both kinds of language models and it greatly depends on the task involved which model is preferred.

Speech Recognition Grammars

In the following chapter we give an overview on the structure and characteristics of formal grammars. With this background information, we investigate the differences between regular grammars and grammars for speech recognition systems. We also explain syntax and composition of a `grm` grammar file for speech recognizers.

3.1 Chomsky Hierarchy

According to Chomsky, a grammar is defined as

$$G = (V, \Sigma, R, S). \quad (3.1)$$

V is a finite set of non-terminal symbols represented in capital letters. The alphabet or terminal symbols accepted by the grammar is represented by Σ . R defines the production rules for the grammar; S denotes the start symbol and has to be an element of V . With this quadruple a language can be described.

Chomsky classifies grammars into four types which are shown in table 3.1. The grammars are organized in a hierarchy based on their complexity [CHOMSKY, 1956, CHOMSKY, 1963]. In table 3.1, the grammars are arranged in descending order of complexity: Type 0 grammars cover every language, whereas Type 3 grammars only cover languages with a specific rule pattern. The Chomsky hierarchy is a subsumption hierarchy so that a regular grammar is a recursively enumerable grammar by definition but not vice versa.

In the table, the symbols denote the following: α , β and γ are arbitrary combinations of terminal and non-terminal symbols which may be empty; ϵ is the empty string; A and B are single non-terminals; x is any string of terminal symbols including ϵ .

Type	Grammar Name	Automaton	Rule Skeleton
0	recursively enumerable	Turing machine	$\alpha \rightarrow \beta, \alpha \neq \epsilon$
1	context-sensitive	Linear-bounded automata	$\alpha A \beta \rightarrow \alpha \gamma \beta, \gamma \neq \epsilon$
2	context-free	push down automata	$A \rightarrow \gamma$
3	regular grammar	finite-state automata	$A \rightarrow xB$ or $A \rightarrow x$

Table 3.1. Chomsky Hierarchy

For our purposes, we focus on context-free and regular grammars, as we do not require greater complexity for English language rules¹. We start by examining context-free grammars (CFG). With them, any single non-terminal can be rewritten as an arbitrary string of terminals and non-terminals (including the empty string) [JURAFSKY & MARTIN, 2008, Chapter 16].

A grammar G describes a language \mathcal{L} with

$$\mathcal{L}_G = \{w \mid w \in \Sigma^* \text{ and } S \xRightarrow{*} w\}, \quad (3.2)$$

where w is a string of terminal symbols, S the start symbol and Σ^* any combination of elements in Σ including the empty string. The asterisk above the \Rightarrow stands for “in zero or more derivation steps”. The asterisks after Σ and on top of \Rightarrow represent the Kleene closure which is a unary operation indicating zero or more occurrences of the preceding character [JURAFSKY & MARTIN, 2008, Chapter 2].

With this, the definition above reads as follows: A language \mathcal{L} is described by a grammar G such that any string w of terminal symbols in \mathcal{L} consisting only of terminal symbols in Σ can be derived from the start symbol S in zero or more steps with the rules in R and no w not in \mathcal{L} can be derived from S with these rules.

A typical example of a language described by a context-free grammars is:

$$a^n b^n : n \geq 1. \quad (3.3)$$

The rules in R of the corresponding grammar G are:

$$S \rightarrow ab \quad \text{and} \quad S \rightarrow aSb. \quad (3.4)$$

These rules ensure that any word in the language contains only a 's and b 's, all a 's stand left of all b 's and the number of a 's is the same as the number of b 's.

The language in equation (3.3) is not regular as the pumping lemma² for regular languages shows. It relies on properties of regular grammars

¹ In general, speech recognition grammars are either context-free or regular grammars. When it comes to grammar formalisms which aim at describing an entire language, more complexity is required [JURAFSKY & MARTIN, 2008, p. 566].

² [LEWIS & PAPADIMITRIOU, 1981, HOPCROFT & ULLMAN, 1979, BAR-HILLEL ET AL., 1961]

which are not properties of CFG. It supplies a condition where the information that the condition is not met by the grammar entails that the grammar must be context-free or of higher complexity. In Jurafsky and Martin [JURAFSKY & MARTIN, 2008, pp. 567], a comprehensive explanation and proof for this is given.

Regular grammars and the corresponding languages follow the most restrictions in the Chomsky Hierarchy and describe the smallest proportion of languages. Only rules matching the pattern

$$A \rightarrow xB \text{ or } A \rightarrow x \quad (3.5)$$

are allowed. The regular language

$$a^n b^m \text{ with } n, m \in \mathbb{N} \quad (3.6)$$

contains words where all a 's stand left of all b 's and the number of a 's and b 's is each greater than one and may be equal. However, no regular grammar can ensure that the number of a 's and b 's is equal.

Regular grammars can alternatively be described with regular expressions³ or equivalent finite-state automata. For every regular grammar, a deterministic and a non-deterministic finite-state automaton can be created⁴ which can be used for a visual representation of the grammar. Regular expressions permit the creation of one expression which carries the same information contained in the production rules for a grammar.

As previously mentioned, more complexity with regard to the grammar is not needed for natural languages, like English or German. As Chomsky states, English is a context-free language and cannot be regular due to certain syntactic constructs [CHOMSKY, 1956]. Nevertheless, Jurafsky and Martin [JURAFSKY & MARTIN, 2008, Chapter 16] remark that the sentences leading to the assumption that English is a context-free language are difficult to understand for humans⁵. For most phenomena of English, a regular grammar is sufficient. For syntactic matching, e. g., subject-verb agreement, a context-free grammar is necessary.

For speech recognition in dialogue systems, mostly regular grammars are constructed. As mentioned above, only a small part of a language is considered in a dialogue system and all issues with respect to syntactic matching can be solved by adding specific rules for each case.

After having looked at the formal definitions of grammars, we will present speech recognition grammars in the following section and point out the differences to formal grammars.

³ First developed by Kleene [KLEENE, 1956].

⁴ A deterministic finite-state automaton possesses exactly one transition for every item in the alphabet for every state. For more information, see [HOPCROFT & ULLMAN, 1979].

⁵ This has been shown by [YNGVE, 1960, CHOMSKY & MILLER, 1963].

3.2 Developing Speech Recognition Grammars

Speech recognition grammars are usually part of a domain-dependent dialogue system which is limited to understanding a closely defined set of sentences. In order to achieve this, we develop grammars which define every possible utterance for a particular domain and remain within a considerably small excerpt of the target language. We have designed our grammars exclusively for the English language.

There are attempts at writing a grammar which covers all possible grammatical constructs, e. g., the LinGO⁶ project. They created a Head Driven Phrase Structure Grammar⁷ (HPSG) for the English language, the English Resource Grammar (ERG). It aims at covering most phenomena of English, including semantics, and at being linguistically precise. According to Copestake, LinGO ERG is one of the biggest freely available grammars and is based on a type hierarchy [COPESTAKE & FLICKINGER, 2000]. The grammar's lexicon was taken from the Comlex project and is based on the Oxford Advanced Learner's Dictionary [GRISHMAN ET AL., 1994]. For the purposes of ERG, the mentioned type hierarchy had to be added manually to the lexicon. With respect to accuracy, the LinGO ERG produces correct semantic representations of 83% of the Verbmobil⁸ corpus for which it was originally designed. Although excerpts of LinGO ERG have been created for educational purposes, we chose to build a new grammar specifically designed for our own purposes.

The approach used in ERG comprises many features of the language. If we based our grammars on them, it would lead to a far more complex structure than we aim at. This notion holds for all available grammars and grammar formalisms: They cover the entire language and the amount of time necessary for extracting the relevant parts can be spent more efficiently by writing grammars just for the required parts.

Since maintenance will also be performed by non-linguists, it is essential that our grammars contain only linguistic features which every speaker of the English language is familiar with. In contrast to ERG, our grammars can be combined modularly for a specific domain. We will go into more detail on the grammars' modularity and construction in chapter 4.

The following sections focus on concrete examples of the syntax and the file structure of our grammars.

⁶ <http://www.delph-in.net/erg/>

⁷ The Head-driven Phrase Structure Grammar is a lexicalized generative grammar developed in the late 1980s. For a detailed explanation, see [POLLARD & SAG, 1994].

⁸ Verbmobil is a project initialized by the German Federal Ministry of Education, Science, Research and Technology (BMBF) "the development of a mobile translation system for the translation of spontaneous speech in face-to-face situations" which was chaired by Prof. Dr. Wolfgang Wahlster at DFKI from 1993 to 2000. For further details, see [WAHLSTER, 2000].

3.2.1 Grammar Format

We write the grammars for an Ontology-based Dialogue Platform (ODP) application developed by SemVox GmbH⁹ [PFALZGRAF ET AL., 2008, SCHEHL ET AL., 2008, PFLEGER, 2007, ROMANELLI ET AL., 2010]. The ODP grammar format is based on the Nuance Grammar Specification Language (GSL) [NUANCE, 2003] syntax as the underlying speech recognizer in ODP is the Nuance Recognizer. We illustrate the ODP syntax and explain the structure of a grammar file.

Syntax

A grammar rule usually looks like this:

VARIABLE [term]. (3.7)

The expression in capital letters denotes the variable and is used in phrase specifications to replace the term in square brackets. In order to be able to recognize and remember easily which variable represents which terms, the variable name should contain all necessary information for reconstructing the contents behind the variable. All variable names (for phrases and synonyms) may only be used once in a GSL-project. For distinguishing allowed input and variables, the input is always written in lower case letters and the variables in upper case letters. In addition, variables may contain the special character “_”. An example for a synonym GSL-rule:

NP_WEATHER [(the weather)]. (3.8)

This rule states that the variable NP_WEATHER is replaced by the words “the weather” whenever it occurs. The round brackets around both words indicate a conjunction. If there are several words between the square brackets, they form a disjunction:

FORECAST [forecast outlook prospects], (3.9)

meaning that the variable FORECAST can be replaced by either *forecast*, *outlook* or *prospects* when used in a phrase or more complex rule. Conjunctions and disjunctions can be used together in one rule, forming a more complex one:

S_SHOW_WEATHER (3.10)
[(?please show ?me

([NP_WEATHER NP_FORECAST]))], with

⁹ <http://www.semvox.de>

$$\text{NP_FORECAST [(the ?weather FORECAST)]}. \quad (3.11)$$

The question mark indicates optionality and the round brackets enclosing the disjunction ([NP_WEATHER NP_FORECAST]) are a syntactic requirement. If spelled out, the rule depicts the following utterances amongst others: *please show me the weather forecast, please show the weather prospects, show me the outlook*. As the rule in (3.10), variables may also be used on the right hand side of a rule. Grammar rules for Nuance and ODP may be used recursively which is a feature that they share with Chomsky’s definition of formal grammars.

Constructs in the ODP grammar format (see table 3.2) are similar to regular expressions. In ODP, there exist no character classes as in regular expressions because the focus lies on matching entire words rather than single letters or digits. A variable in ODP is the closest resembling pattern to character classes in regular expressions. The “.” in regular expressions which matches any character, can be compared best to the “GARBAGE” variable in ODP. GARBAGE can be used to match any input but has to be used with care as it matches as much of a given utterance as possible instead of a single character, often leading to incorrect recognition results. Hence, the GARBAGE variable should be avoided and there should be explicit rules for all utterances.

Characters	Interpretation	Example
VARIABLE	a variable	CITY
?word1	word1 is optional	?please
[word1 word2 word 3]	disjunction	[salzburg rome saarbruecken]
(word1 word2 word 3)	conjunction	(I would like)

Table 3.2. An overview of GSL constructs

3.2.2 File Structure

In this section, we present the structure of an ODP grammar file and focus on the parts which are relevant for the grammars we developed. The base construct of an entire file without any rules can be seen in appendix A, while a detailed explanation of an ODP grammar file design is explained in the ODP documentation [SEMVOX GMBH, 2009].

The W3C defines two forms for the syntax of speech grammars¹⁰: Augmented Backus-Naur Form and Extensible Markup Language (XML). In this work, we will focus on the XML-based grammar. A speech grammar filename extension in the XML format is *grm*.

¹⁰ <http://www.w3.org/TR/speech-grammar/>

The opening line states the language (en stands for English) and the grammar's file name:

```
<grammar lang="en" file-name="ami_general.grm">.
```

If we need to import files from other locations because the current file relies on variables defined there, we can import them by using

```
<import location="grammar/ami_general.grm"
  locator="jar"/>,
```

to specify the file's location.

The next tag defines phrases for a particular type of request. Several phrases are collected in one utterance tag which looks as follows:

```
<utterance name="DECLINE">
  <phrases>
    <phrase>no</phrase>
    <phrase>no thanks</phrase>
    <phrase>no not necessary ?thanks</phrase>
  </phrases>
  <semantic-interpretation>
    <object type="">
      <slot name="">
        </slot>
      </object>
    </semantic-interpretation>
</utterance>
```

The utterance specifies a unique name which can be used as a variable in other utterances and should specify the type of information that is passed on to the speech recognizer. There has to be at least one phrase in every utterance.

Each utterance has an obligatory `semantic-interpretation` tag where the phrases' content is mapped to an ontological representation. An ontology contains mappings from real-world objects and concepts to a computer-understandable representation¹¹. Restrictions and relations that exist between real-world entities are modelled accordingly in an ontology.

There has to be at least one `object` tag in the `semantic-interpretation` tag. `object` tags can contain `slot` tags and vice versa. `slot` tag can also contain variable tags which collect information from the phrase's variables.

The next two tags in a `.grm`-file are `named-entity-grammar` and `semantic-grammar-rule`. The `named-entity-grammar` tag is used for objects with proper names such as towns, persons, or songs. As for utterances, several formulations can be mapped to the same named entity. In the `semantic-grammar-rule` tag, semantic information can be added to a specific rule. As we do not require either of the two constructs because we have too

¹¹ For more information on ontologies and how to create them see http://protegewiki.stanford.edu/wiki/Main_Page.

few named entities and, so far, no semantic information to include, we will not explain them further. The ODP documentation [SEMVOX GMBH, 2009] gives a detailed explanation for these tags. If we extend the grammars for a larger context, we will include the tags in our grammar files (see chapter 6).

The variables specified by a grammar are collected in the `gs1` tag where the rules are defined. They do not specify an entire utterance but rather parts a phrase is composed of. An element in `gs1` is always composed of a rule name in capital letters and an expression of arbitrary complexity:

```
<gs1>
  S_SHOW_WEATHER [(?please show ?me ([NP_WEATHER
    NP_FORECAST]))]
  NP_WEATHER [(the weather)]
  NP_FORECAST [(the ?weather FORECAST)]
  FORECAST [forecast outlook prospects]
</gs1>.
```

The last part in a `.grm` grammar file is the lexicon which looks like this:

```
<lexicon>
  <entry key="">
    <definition value=""/>
  </entry>
</lexicon>.
```

Here, the pronunciation of foreign words which are included in the grammar can be specified explicitly. This is necessary if e. g. German or French words need to be included in an English grammar. In our case, we specify the pronunciation for Saarbrücken:

```
<lexicon>
  <entry key="saarbruecken">
    <definition value="za:bRYk@n"/>
  </entry>
</lexicon>.
```

The phonetic description is based on the computer phonetical alphabet¹² (CPA) which allows the typing of phonetic expressions with an ordinary keyboard [NUANCE, 2003, KLUGER-KRUSE, 1987].

3.3 Chapter Summary

In this chapter, we introduced formal grammars and their classification according to Chomsky. We also presented an overview of the properties of regular and context-free grammars as these types of grammars are used for speech recognition. In addition, we discussed the topic of using a large HPSG for our

¹² The CPA is better known as SAMPA which is widely used for transcribing words phonetically with a computer.

purposes and showed that it is easier to write the grammar by hand than to extract the required structures from a big HPSG. We gave examples on the syntax and structure of the designated grammar format. Moreover, we provided an `grm` file structure overview which explained the function and composition of the parts of a grammar file.

Implementation

After introducing the theoretical foundations in the previous chapters, we now present the grammars which we developed. We show the context in which they are implemented and we list requirements which we established for our grammars.

We write the grammars for situations taken from a use case of the EU project Interactive Knowledge Stack¹ (IKS) whose goal is to provide an open source technology platform for semantically enhanced content management systems. Several use cases were designed to illustrate the features which the platform will provide. We focus on a use case where the IKS software is integrated into a bathroom with an ambient intelligence (AmI) environment². The grammars we designed are part of the Dialog-based Ambient Interaction Manager as depicted on the right hand side in figure 4.1.

4.1 Modular Grammars

The central aim in this work is to develop modular grammars for speech recognition. We interpret the term “modular” as follows: There is one file which defines all general variables and rules. All grammar files for specific contexts can be combined with this general file by linking to it. Links to other general files (allowing temporal expressions) are optional whereas direct links between two specific files are not permitted (see figure 4.2).

We have chosen to design the grammars modularly for multiple reasons: better recognizer performance, easier maintenance, as well as a more clearly structured overview. We hope to achieve a better recognizer performance as only required modules are activated for the recognition process. With small

¹ <http://www.iks-project.eu/>

² An ambient intelligence environment is an environment which can react and respond to a human's presence.

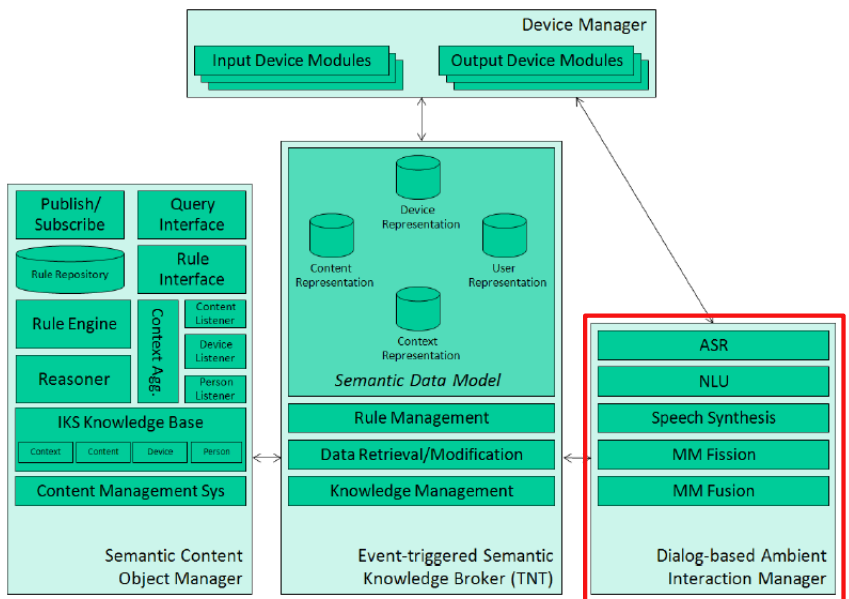


Fig. 4.1. The IKS use case architecture. The grammars are part of the Dialog-based Ambient Interaction Manager in the red framed section which is integrated in an AMI bathroom.

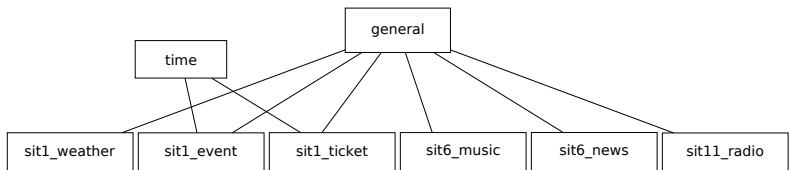


Fig. 4.2. An example of our modular structure: “general” defines all general concepts which are required for all specific files (“sit1.weather”, “sit1.weather”, etc). For some specific files it is necessary to include a supplementary “time” file.

modular grammars, less words can be confused by the recognizer as the vocabulary is smaller. The modularity will hopefully lead to good maintenance conditions: A new developer who is not familiar with our grammars is able to accustom himself quickly to them and their structure if they are organised in a number of small files rather than one big file. Furthermore, variables are used in the defining grammar only (with exceptions for general vocabulary files), avoiding cross-references which thus simplifies the process of renaming or restructuring variables. The modular structure leads to a better overview for determining which utterances are covered by which grammar file. As a result, desired information can be accessed faster. A whitepaper by Lumenvox proposes to use modular grammars for exactly the same reasons [LUMENVOX, 2010].

For this work, we created two grammars containing general variables: `ami_general.grm` (see appendix C.1) stores globally used information, e. g., days of the week, temporal adverbs, cardinal and ordinal numbers as well as cities. The other grammar, `ami_time.grm` (see appendix C.2), enables the creation of times on the clock, e. g. “three o’clock”, “4 a.m.” or “half past five”. This proved useful as time specifications are necessary in less situations than other temporal expressions (weekdays or the time of day). `ami_general.grm` is included in each of the subgrammars we introduce in the following section while `ami_time.grm` is only added when required.

4.2 Situations for the AmI Bathroom

During the IKS project’s planning stage, members of the project devised eleven situations in corporation with Duravit³ employees who are responsible for furnishing the bathroom. The situations are designed for illustrating the platform’s abilities and show to which extent the platform can be used in daily life. They describe clearly defined tasks that range from requesting the weather prognosis or the wardrobe contents to playing individual music and news channels. Each situation requires speech interaction for which we developed a grammar to be integrated into a dialogue system. In this thesis, we focus on the three most popular situations. They ranked highest in a series of user studies prior to this work [JANZEN ET AL., 2010]. The texts describing the three designated situations are shown below⁴. Grammars designed for this thesis and the situations are listed in appendix C.

The text describing the situations (written in italics) is the official formulation taken from the project document “AMI Case Requirements Specification” [JANZEN ET AL., 2010].

4.2.1 Situation 1

It’s Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom. Based on weather information and my calendar, free-time event suggestions are given (e. g., “Today, 8 p.m. – Sneak Preview at CinemaOne”). Do you want to order tickets?

From this situation, grammars for the subsituations requesting weather information (`sit1_weather.grm`), event suggestions (`sit1_event.grm`) and ticket ordering (`sit1_ticket.grm`) have been developed. They can be found in appendix C.3.

³ <http://www.duravit.de>

⁴ See [JANZEN ET AL., 2010] for a full list all eleven situations.

4.2.2 Situation 6

Happily, it's Saturday morning – weekend – I have taken a shower listening to my favourite music. Leaving the bathroom, my partner flits into the room. My music has become silent and my partner is welcomed by music from her/his own music collection. The music starts at the point in the playlist where my partner stopped listening the evening before. After a while, my partner says Stop music and the song fades out. My partner wants to see his/her personal news collage while taking a shower.

For this situation, we developed grammars for music selection (`sit6_music.grm`) and watching the news (`sit6_news.grm`). They can be found in appendix C.4.

4.2.3 Situation 11

It's Monday morning. I am in the bathroom, brushing my teeth and listening to the news on the radio. Then, I take a shower. Now, the news messages are displayed in form of pictures and text at the glass door of the shower.

For this situation, we developed a grammar specifying the medium for the news and performing changes with respect to the radio settings (e. g. volume, radio station) (`sit11_radio.grm`). It can be found in appendix C.5.

4.3 Grammar Creation

Prior to developing the grammars, we closely examined the situations designed for the AmI usecase. Each situation description defines a rough structure that gives clues as to how the dialogue might look. From these descriptions, we devised details for user- and system-initiated dialogues. With this, we wanted to make sure that the user is free to choose how to interact with the system and is not limited to the system's cues. For each situation, we created a state diagram which represents the course of actions for each of the six smaller situations. In figure 4.3, we show a state diagram for requesting weather information taken from situation 1.

In the following is a short dialogue between the system S, and a user U, Peter which we used for creating the grammars to illustrate the diagram:

```
U: Good Morning, bathroom!
S: Good Morning, Peter!
S: Here is the weather for you.
U: Please show me the weather for Sankt Gallen.
S: This is the weather for Sankt Gallen, Switzerland.
```

The user may initialize the dialogue by greeting the bathroom when he enters. As we aim for a natural interaction, this greeting is optional because normally, a room does not need addressing when one enters it. The bathroom,

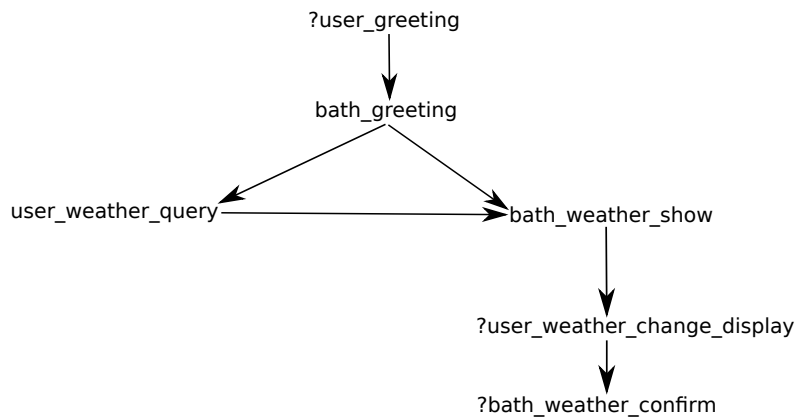


Fig. 4.3. A state diagram of a possible course of actions for the weather request.

however, is designed to realize when somebody enters and who it is. It utters a greeting. After the greeting, the bathroom can directly display the weather for the user's current position or adapt it to the user's plans according to his calendar. Alternatively, the user can ask for the weather in any city at any time. After seeing the weather displayed on the bathroom mirror⁵, the user can change the weather's settings (time and/or place), whereupon the bathroom confirms the change.

We took both dialogue and state diagram into account for designing utterances and phrases for each step in the diagram. At this stage in the development process, the sentences were spelled out completely (see right hand side of table 4.2). Then, we combined parts of utterances into variables, such as

WEEKDAY [monday tuesday wednesday thursday friday saturday sunday] (4.1)

or

NE.LOCATION [salzburg (sankt gallen) saarbruecken rome]. (4.2)

The rule in (4.1) maps the days of the week in a disjunction to the variable WEEKDAY whereas in rule (4.2), NE.LOCATION maps four towns to the identifier.

We combined single word variables as shown above into phrase variables for better structuring and overview in the utterance and phrase tags in the grammar files. For structuring the phrase variables, we used the concept of constituency. It is based on the assumption that words can be grouped into phrases which are interchangeable with others of the same kind and which appear in the same syntactic environments. Constituents were first introduced

⁵ The mirror serves as mirror and multi-touch display for certain pieces of information (weather, event list, music library).

in 1956 [CHOMSKY, 1956] and are used in many grammar formalisms that describe natural languages [JURAFSKY & MARTIN, 2008, Chapter 12].

Abbreviation	Phrase name	example
N	noun	cinema, opera
NE	named entity	rome, camera zwo
V	verb	want, would like
ADJ	adjective	right, left
ADV	adverb	here, there
NP	noun phrase	the weather
VP	verb phrase	would like to,
PP	preposition phrase	in rome, on monday
S	sentence	show the weather for today
Q	question	how is the weather

Table 4.1. Abbreviations for phrases which we used in the grammars

The names of phrase variables contain the phrase’s syntactic category which is meant to help a developer to see where a certain phrase can be placed in an utterance. We show an overview of the abbreviations we used in table 4.1. Here are two examples for phrase variables:

$$\text{PP_WEEKDAY} [(([\text{on for}] \text{WEEKDAY})] \quad (4.3)$$

$$\text{NP_FORECAST} [(the \text{?weather FORECAST})] \quad (4.4)$$

where FORECAST can be *forecast*, *outlook* or *prospects* and WEEKDAY as specified in rule (4.1). Both rules are examples for regular rules in our grammar (see table 3.1 and section 3.1).

The phrases are then combined with single words and other phrases to form ODP phrases. In table 4.2, a choice of possible utterances together with their corresponding rules can be seen⁶. The generated utterances are all written in small letters as specified by the grammar format. Behind every variable in capital letters, there is a reference in the grammar file that defines the words which are represented by a particular variable (see section 3.2.1). To see a list of all grammar files, please see appendix C.

When all rules and phrases are designed completely, the resulting grammar files can be tested on the ODP system. Although testing can be carried out with both written or spoken input, the former allows for a higher throughput of utterances since they do not need to be articulated. Furthermore, speech recognition mistakes can be neglected. The testing and its results are discussed in the next section.

We analysed the mistakes

⁶ The rules have not been expanded entirely.

Rule	possible sentences
Q_HOW_WEATHER	how is the weather
Q_HOW_WEATHER ?today here	how is the weather here how is the weather today here ...
Q_HOW_WEATHER [today tomorrow PP_WEEKDAY] ?PP_LOCATION	how is the weather today how is the weather tomorrow in rome how is the weather on monday how is the weather for thursday in sankt gallen ...
Q_HOW_WEATHER [this tomorrow PP_WEEKDAY] TIME_OF_DAY ?PP_LOCATION	how is the weather this morning how is the weather this afternoon in salzburg how is the weather tomorrow afternoon in rome how is the weather on monday evening how is the weather for saturday evening in rome ...
S_SHOW_WEATHER for [WEEKDAY N_LOCATION]	please show me the weather for monday show me the weather forecast for wednesday show the prospects for monday ...

Table 4.2. Selected sentences to request weather information from the database

After the testing had been finished, the recognizer's mistakes were analysed. Subsequently, the grammars were modified accordingly to achieve higher precision rates. The testing results are discussed in the next section.

4.4 Testing

We determined which sentences are generated and accepted by our grammars by applying two tools which are provided by the Nuance Recognizer.

The first one requires an input of an utterance and a grammar file and it determines whether or not the utterance is covered by the grammar. For more efficient testing, we used an implementation of the tool which requires a list of utterances as input and tests them on a given grammar file. The implementation returns the percentage of utterances covered by the grammar. The lists of utterances were created from phrases in the grammar files.

After a first test, we increased the percentage of accepted utterances by adding brackets to the rules which had been missing previously. In table 4.3, we present the final results for testing the grammars with designated test sets.

file name	accepted sentences
sit1_weather.grm	99.62 %
sit1_vent.grm	100.00 %
sit1_ticket.grm	58.33 %
sit6_music.grm	99.45 %
sit6_news.grm	68.00 %
sit11_radio.grm	100.00 %

Table 4.3. Results in percentage of how many of the test sentences were accepted by the corresponding grammar.

The other tool generates a specified number of utterances from a given grammar file. It does not apply the rules in the order specified in the grammar file but rather applies the phrases in a random order. Only phrases formulated in the utterance tag are generated by this tool while variables which are only specified in the gs1 tag are not considered. Furthermore, the tool does not produce a unique output and several utterances are generated multiple times. We eliminated the duplicates for a better overview and analysis of the result. Below, there are examples for generated sentences from sit1_event and sit1_weather:

- *what is on in the theatre this evening*
- *yes please show me where i can go*
- *what can i see this evening*
- *show plays in theatres in the area*
- *please show the events for this afternoon*
- *i would like to see the one at eleven thirtyfive*
- *how about on friday midday in rome*
- *how about today*
- *how is the weather in rome today*
- *show me the weather for sankt gallen this morning*

We analysed the generated sentences for three parts out of situation 1 (sit1_weather.grm, sit1_vent.grm and sit1_ticket.grm). We discovered that the situation grammars do not produce any nonsensical output. Only the generation of numbers in ami_general had to be improved after this part of the testing.

The results of the evaluation which we performed on the grammars' size and completeness are shown in chapter 5.

4.5 Requirements

We designed the grammars according to requirements that we illustrate and explain in this section. The requirements put a key focus on having grammars which can be maintained and extended easily by others who are rather unfamiliar with the situations and grammars.

The general requirement is to construct the dialogues' structure as naturally as possible so that a designated user is guided through the application without being unsure about questions and answers to give at any point in the dialogue. We achieved this by defining an exact outline for each situation's dialogue beforehand (see figure 4.3 for the diagram for situation 1).

As the modular architecture of the grammars is a key requirement (see section 4.1), we have not included it in this list.

Sentence length

The requests to the bathroom are short and precise. We decided that the user is allowed to include more than one piece of information into an utterance without making the utterance too long.

Short sentences lead to better recognition rates as fewer mistakes can be made during the decoding process by the recognizer. The shorter an input, the less possibilities there are for the recognizer to confuse similar sounding utterances. However, the level of confusion also depends on the grammars' structure and size (see paragraph below). From the user's point of view short sentences are also desirable, as he receives the information faster and his utterances are more often recognized correctly. Although we focussed on short sentences, we did not include one-word commands into our grammars (see chapter 6).

An example for a sentence of acceptable length:

Please show me the weather for Rome today. (4.5)

This sentence is too long:

Would you please be so kind as to show me the weather
for Rome in Italy for today from 1 to 6 pm? (4.6)

Grammar size

Each grammar file remains small and well-arranged. In combination with the modular structure, this leads to a better overview and easier maintenance possibilities.

We took care that situation-specific variables are defined only in the grammar file for the situation and that more general concepts are defined in `ami_general.grm`. In addition, we aim at creating phrases that cover as many utterances as possible without overgenerating and whose structure remains understandable.

Variables

Variables have to fulfil several requirements: the names have to be unique, each variable has to represent at least two words, and the name has to represent the content. In order to ensure that the variables can be understood easily when they occur in a phrase, it is important that the name gives enough information about what a variable represents. We avoided variables that stand for only one word to avoid making the rules more complicated than necessary. During the revision phase of the grammars, we abolished variables such as

PREP_IN, POLITE, V_FIN_ARE, (4.7)

standing for “in”, “please” and “are”, respectively. Variables are more sensible when several words from one category are combined e. g.

WEEKDAY and FORECAST, (4.8)

standing for the days of the week and nouns that are synonymous with “forecast” (see example 4.4).

To avoid overgeneration, a variable should not contain too much information, i. e. not represent many words. This is one of the reasons for creating a separate grammar for temporal expressions: When `ami_time.grm` is not included in a grammar file, numbers cannot be interpreted as time specifications.

Phrase structure

The structure of a phrase is shallow. By shallow, we mean that a variable directly represents the word and is not rewritten with other variables. The more rewrite rules there are, the more complex it will be to understand the phrases. As for the requirement of the grammars’ size, a shallow phrase structure leads to better maintenance possibilities and comprehensibility for others.

We decided to include some elements of Chomsky’s transformational grammar for the underlying phrases and constituents (e. g., noun phrases, verb phrases; see [CHOMSKY, 1956] for more details), but to maintain an easily understandable structure for persons with little linguistic background. This phrase contains too many variables:

Q_WHAT AUX_CAN PERSP_I V_SEE tonight. (4.9)

The same phrase with less variables:

what can i V_SEE tonight (4.10)

where V_SEE can stand for *see*, *watch* or *view*. The example in (4.10) contains less variables, is simpler and produces the same output as example (4.9).

Our phrases have been created from the sentence structure without transforming every constituent into a variable of its own.

Phrase scope

Since the entire project aims at the English language, we design grammars for English utterances. Furthermore, we exclude the recognition of disfluencies⁷ at this point of the implementation (see section 6.2).

We have focused on allowing a natural interaction with the system and have included some ungrammatical utterances, as this one shows where the user asks for the weather in Rome after the weather for another location has been displayed previously:

And how about Rome? (4.11)

Natural Language Understanding

While creating the phrases for the grammars, we kept in mind that the grammars will eventually require a natural language understanding (NLU) component. This is necessary when the grammars are integrated into a dialogue system and the utterances need to be mapped to a meaning which the underlying system can understand and process. The NLU component has to be designed according to the structure of the utterances and all information from an utterance has to be mapped to the corresponding parts in the ontology.

We created these requirements for our six specific grammars and attended to them. We hope that we have constructed a basis which other developers can rely on. Maybe our requirements can be ported to other domains and projects.

4.6 Grammar Revision

During the development process, we needed to address the following challenges and to consider them in revisions of the grammar files:

1. We needed to find all possible sentences for a particular situation, e. g., requesting the weather today in Rome. A user can articulate a multitude of different utterances to request the desired information. For this purpose, we first thought of utterances for ourselves and then performed an evaluation. We tested whether the utterances we devised cover the greater part of utterances for a certain situation. See chapter 5 for the results of the evaluation.

⁷ Disfluencies are disruptions in fluent speech and include hesitations (“um”, “uh”, “hm”), stutters (“I th-th-think”), restarts (“We should write, we may write”) and discourse markers (“you know”, “sort of”) (see Besser [BESSER, 2006]).

2. We had to address the fact that the grammar might be too small; as we are assuming a closed vocabulary, our grammar should contain every possible word and utterance for the situations. By performing an evaluation and extending the grammars henceforth, we tried to keep the amount of unknown words as small as possible.
3. Special attention was drawn to the design of a time grammar that only creates correct time expressions. As a combination with `ami_general` would have led to confusions during the recognition process, we extracted the variables for temporal expressions into a separate file.
4. Overgeneration describes the issue that utterances which should not be part of the grammar are generated and thus, the grammar recognizes nonsense input. This had to be considered for all grammars. Much of the overgeneration was eliminated during the revision and the remaining problematic rules were adjusted after final testing.
5. It was necessary to include utterances in the grammar files which allow either the user or the system to take initiative during the dialogue.

We successfully met these challenges and were able to consider all mentioned items in the current version of the grammars.

4.7 Chapter Summary

We presented the general environment and outlined the specific situations for which we created our speech recognition grammars. We demonstrated our reasons for choosing a modular grammar approach, explained in detail how the grammars were developed and gave an insight into our testing procedures. In addition, we illustrated the grammars' features using examples and gave information on the variables and abbreviations we used. Furthermore, we formulated a list of requirements which our grammars attend to. Lastly, we presented the challenges we faced during the development process.

Evaluation

In this chapter, we present the evaluation of our grammars which we used to determine whether others would use the utterances we anticipated. We present the evaluation method and results in this chapter as well as a proposal on how to expand the rules for including the findings from the evaluation. We confine ourselves to describing the results for situation 1 and 6.

The evaluation which we performed is the first in a series of evaluations necessary for an optimal performance and coverage of the grammars. More evaluations lead to a more detailed definition of requirements and allow us to adapt the grammars better to the users' needs. We show an example for an improvement cycle in figure 5.1: We start by defining requirements for our grammars and then design them accordingly. During the realisation, we build the grammars based on the design and transfer them to the appropriate format and syntax. The evaluation is used for finding out to what extent our grammars match the users' needs. After the evaluation, the results are analysed, the requirements adapted accordingly, and a new development cycle begins.

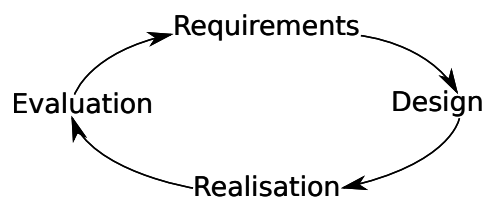


Fig. 5.1. The development cycle for our grammars.

In contrast to the testing of the grammars (see section 4.4), the evaluation is conducted for detecting which of our utterances would be used by others. The testing, by comparison, is necessary for determining whether the sen-

tences we designed are covered by the grammar and whether the grammars produce nonsensical output.

5.1 Method

The evaluation was conducted as an interview. We gave prompts encouraging the participants to articulate utterances they would use in a given situation. The prompts had been designed prior to the evaluation and can be seen in appendix B. The participants' utterances are the results we were interested in. Before each interview started, we briefly introduced the participants to the general situation they had to imagine for the evaluation.

We recorded all interviews with a table microphone. After all sessions were completed, we transcribed the target utterances and participants' remarks from the discussion after the experiment. The target utterances are the utterances formulated by the participants for requesting information from the bathroom. The remarks are addressed in section 5.3.

A total of 19 students of mixed backgrounds participated in the evaluation, eleven of whom had taken part in previous IKS experiments. Ten subjects had a background in either computational linguistics or computer science while the other subjects came from other study programmes. There were 17 German native speakers and two English native speakers.

The evaluation was conducted with groups of subjects; there was one group with one, five groups with two, and two groups with four participants. Each interview was carried out in English and lasted between 25 and 30 minutes. We provided visual aids in form of a doll's house bathroom and a slide presentation.

5.2 Results

There were two possible outcomes of the experiment: If we considered all possible utterances for the situations, the grammars cover all utterances used by the subjects. If we did not consider every possible utterance, the subjects provide us with new sentence structures and formulations which are not yet part of our grammars. The analysis of the recordings confirmed the latter case.

For a better overview, we divided the subjects' utterances into five categories: *question* which contains all utterances consisting of an grammatically correct questions, *sentence* which contains grammatically correct sentences, *abstract request* amasses all sentences which ask for the required information indirectly, *commands* in which all ungrammatical and one-word utterances are collected, and *unusual formulations* which contain swear words or otherwise unsuitable formulations.

For all situations, we discovered that subjects found it difficult to interact with the system naturally and they preferred using commands or short sentences.

Situation 1

The average length of user-uttered sentences for the entire situation 1 is 5.75 words per utterance and for each of the eleven questions, on average 3.98 utterances were suggested. For the averages, we treated all contractions (“what’s”, “don’t”, “can’t”) as one word, as well as times (“8:30”, “7 o’clock”, “4 p.m.”), locations (“Camera Zwo”, “Sparte4”) and titles of plays and movies (“Alice in Wonderland”, “Up in the Air”).

We address situation 1 in greater detail and thus consider utterances for the three parts of the situation (“requesting weather information”, “choosing an event” and “picking seats for the chosen event”) separately.

Weather request

According to our state diagram (see figure 4.3), the first part consists of requesting today’s weather for the current location and changing either day or place of the weather forecast. The participants were first asked to change the day and then the place (to London).

We give some examples of utterances from different categories in table 5.1 and go into detail on how to extend the grammar file to match the new utterances in section 5.3.

Participants provided 18 *questions* altogether, of which 15 differed from each other. For the *commands*, there were six utterances and for the *sentences*, there were four which were all different. The eleven *abstract requests* mainly focused on asking for the nature of the weather (rain, sun, wind) or the type of clothes that should be worn in order to be dressed appropriately. The proportion of the number of *questions* to the number of *sentences*, *abstract requests* and *commands* was roughly the same for changing the time or place as it was for requesting the weather for the current location.

Choosing an event

In this part, the subjects were asked how they would choose an item out of a list of events. The list consisted of the name of the location (opera, name of the cinema or club), the event’s starting time and the event’s title (“Alice in Wonderland”, “Otello”). Most participants used a unique item in the list to specify which event they prefer going to. Please see table 5.2 for example utterances. Words in angle brackets indicate any item out of that group, e. g., for title “Alice in Wonderland”, for time “eight thirty” and for location “Cinestar”.

The subjects showed a clear preference for either *sentences* or *commands* as method of selecting a particular item from the list. We recorded two identical

Category	Example
question answer	what's the weather forecast how is the weather going to be today what's the weather today in Saarbrücken ...
sentence	give me the weather for today please show me the weather ...
abstract request	is it going to rain will it be sunny do i need a coat ...
commands	weather forecast weather Saarbrücken weather today ...

Table 5.1. Example utterances for requesting weather information

Category	Example
question	are there any tickets left for <title>
sentence	please select <location>at <time> I want to go see <title> show me details for <location> ...
abstract request	I'll go to the club I want to go in <location1>, not <location2> ...
commands	<location>, <time>, <title> <title>at <time> <location> earlier <title> ...

Table 5.2. Example utterances for choosing an event from the list

questions. This finding stands in direct contrast to the other situations where most answers given by participants were *questions*.

When participants had to ask for a list of events, the programme of a particular cinema or viewing times of a movie, they mainly used *questions* (46%), the same amount of *commands* and *sentences* (each 22%) and few *abstract requests* (9%). Many of the participants using *commands* said that they rely on the system knowing the bathroom's location whereas the ones using *questions* more often specified the city. In table 5.3, we present a choice of utterances from this part of the evaluation.

Category	Example
question	what's happening tonight what's on in Saarbrücken what's in cinema tonight which plays are on what's playing at the Cinestar tonight where can I watch <title> ...
sentence	show me today's events please tell me what's going on in Saarbrücken tonight show me the Cinestar programme book me in for <title>, <time>at Cinestar ...
abstract request	party time tonight I feel like watching a movie tonight Is <title>still running at Cinestar ...
commands	events in <city> the opera <title>, <location> Cinestar, <title>tonight <title>, <time>. Give me that ...

Table 5.3. Example utterances for specifying which events should be searched for

We observed that participants used less utterances for locations they were not interested in. This fact became particularly apparent when they were asked to come up with utterances requesting the opera's playtime schedule. By far the most commands were expressed for selecting a particular movie which might be a result from the average age of the participants.

Picking seats

The last part of situation 1 was to choose seats from a cinema seating plan. We evaluated two different scenarios: In the first, the subjects were shown an empty seating plan and had to choose the seats for themselves. 42% of the utterances were *sentences* while the *abstract requests* and *commands* each account for 28% of the utterances. In all but one utterance the row and seat numbers were indicated explicitly.

The other scenario featured a seating plan with two seats in the second row which the bathroom had suggested. As there are only few persons who like to sit close to the screen, the subjects were asked to change the location of the seats. In this case, *commands* clearly dominated the utterances (44%): They ranged from giving the row and the seat over a rough location in the room (“back, middle”) to the general area (“front rows”, “somewhere in the back”). See table 5.4 for a longer list of utterances.

Category	Example
question	can I have something further back could you pick some seats in the last rows
sentence	please pick me seats which are more in the back give me seats <row>, <number>and <number> give me a middle row seat change seats to <row>, <number>and <number>
abstract request	it's too close to the screen avoid the first four rows row <row>would be better
commands	<row>, <number>and <number> one row back back rows, middle seat down not those

Table 5.4. Example utterances for changing proposed seats

5.3 Discussion

We compared the utterances given by the subjects to the ones we had in our grammars: in three from six parts the overlap tested was smaller than 2%. Subsequently, we extended the grammar for the most used utterances. In the

following, we demonstrate how to improve the grammar coverage for chosen grammar files.

For requesting the weather information from the system (defined in grammar `sit1_weather.grm`), we designed the variable

Q_WEATHER [(how is NP_WEATHER)]. (5.1)

In our evaluation, we found nearly the same questions as we can already generate but using “what” instead “how” as a question word. We can introduce a further variable:

QUESTION [how what] (5.2)

and extend the variable in (5.1) as follows:

Q_WEATHER [(QUESTION is NP_WEATHER)]. (5.3)

If we extend Q_WEATHER, we cover 8 of 18 of the questions for the weather sub-situation. Further utterances can be incorporated by adding permutations of the variables in the existing sentences. This way, the grammar recognises

what’s the weather today in Saarbrücken, (5.4)

as well as

what’s the weather in Saarbrücken today. (5.5)

When looking at the phrase

Q_WEATHER ([today tomorrow tonight PP_WEEKDAY]) ?PP_LOCATION (5.6)

which produces utterances like *how is the weather today*, *how is the weather tomorrow in rome*, the utterance

what’s the weather like (5.7)

can be incorporated by adding “like” into the disjunction:

Q_WEATHER ([today tomorrow tonight PP_WEEKDAY like]) ?PP_LOCATION. (5.8)

In the same way, we added another variable to

S_SHOW_WEATHER
[(?please show ?me (5.9)

([NP_WEATHER NP_FORECAST]))] to match *give me the weather* in addition to *show me the weather*:

S_SHOW_WEATHER [(?please SHOW_GIVE ?me (5.10)

([NP_WEATHER NP_FORECAST]))] with

SHOW_GIVE [show give]. (5.11)

With these small extensions, the coverage of `sit1_weather` increases from 1% to 14.85%.

The remaining *questions* cannot be incorporated as easily as the ones above as they contain future tenses, e. g., “is going to be” or “will be” which modify the word order and requires more work. Further evaluations have to show whether these extensions are necessary. The abstract requests will be the most difficult ones to integrate into the grammar as the information in these sentences is often given implicitly. For our grammars, we chose not to integrate the commands and leave them for future extensions.

The news grammar (`sit6_news.grm`) which is part of Situation 6, showed the best improvement in coverage. When we tested the participants’ utterances on the original grammar, it covered 11.11% of the utterances. After we had added four variables and seven phrases, the new grammar matched 31.48% of the utterances.

To achieve this, we added

NP_SHOWER [(in the shower)] (5.12)

and

PP_TV_CHANNEL [(FROM_AT TV_CHANNEL)] (5.13)

with `FROM_AT` standing for *from* or *at* and another rule which stands for broadcast names (e. g. *tagesschau* or *bbc news*).

With respect to the phrases, we added the following amongst others:

[show give] me the news?(NP_SHOWER) ?please (5.14)

which creates *show me the news please*, *show me the news in the shower* or *give me the news in the shower please*.

In addition,

?(can you) start the news ?broadcast ?please (5.15)

allows the user to say *can you start the news broadcast please*, *can you start the news please* or *start the news*.

The other grammar files also showed an improvement in the coverage but the percentage of covered utterances did not increase as much as with the two mentioned grammar files (see table 5.5).

Remarks

While analysing the data, we discovered the tendencies shown below in the subjects utterances. We also present some of the remarks the participants made in the discussion after the evaluation. Where possible, we try to give an explanation for each of the tendencies and remarks.

file name	original grammar	modified grammar
sit1_weather_request	1.00 %	14.85 %
sit1_vent_recomm	1.20 %	7.83 %
sit1_ticket_order	1.69 %	11.86 %
sit6_play_music	6.47 %	18.24 %
sit6_play_news	11.11 %	31.48 %
sit11_radio	1.42 %	11.21 %

Table 5.5. Results in percentage of accepted sentences how many of the test sentences were accepted by the original and the modified grammar.

1. Only some of the utterances contained a direct subject (“I”, “you”); most of them were impersonal. In our opinion, this is because the users are unsure how to interact with the system. On the other hand, most of the utterances are either commands (“give me”, “show me”) or questions which do not require a direct addressee.
2. The *questions* contained contractions where possible (“what’s”, “how’s”). As the interaction with the dialogue system is supposed to be natural, our grammars should cover contractions and colloquialisms¹.
3. Several participants included anaphora² in their utterances and stated that it would be a desirable feature for the system. For achieving this aim, the dialogue system needs to be aware of the preceding dialogue(s).
4. Participants stated that it would be more interesting and convenient to use utterances stating what they want to do (“I want to go to the cinema”) rather than explicitly asking for information (“Give me the cinema programme for tonight”). Integrating this feature into the dialogue system will require context information as well as an extensive grammar with a comprehensive NLU component.
5. Several subjects expressed the wish to be able to abbreviate long titles using a keyword. This would be convenient for long film titles and even more for song titles.
6. *Abstract requests* point out in which way we need to extend the abilities of our grammars and the system in general. Some examples: The list of events proposed by the bathroom should be extended if requested (“Show me more dance events”); If only cinema seats in the front row are still free, other screenings at a different time or in another theatre should be made available (“Is <title> also shown in <location>”); The weather request should be linked to a clothes recommendation when asked for (“I’ll be in London next week, what should I bring”);
7. Apart from the possibilities to use speech input, some subjects suggested that multimodal input methods would be quicker, especially for choosing

¹ e. g., ungrammatical sentences and informal words like “flick” for “film”

² Anaphoric references point to previously introduced entities in a text or dialogue. The most common anaphora are personal pronouns.

the seats in the cinema which can be realised by touching the desired seats. This feature is already considered in the bathroom's design. As the evaluation had to take place in absence of the real bathroom, this feature could not be demonstrated.

8. Although many participants included the location and "today" into their utterances, all of them stated later on that they assume that the bathroom takes the current location and day as default input.
9. Another aspect mentioned by many participants is that it is much easier to remember short commands than having to use long sentences. An argument for short commands was the situations' setting which is in the morning. Most people are not communicative in the morning and do not want to use long sentences to receive the requested information.

5.4 Future Evaluations

Our experiences have shown that groups of two participants give the most valuable results. Larger groups are only advantageous if the participants know each other beforehand.

Once the bathroom exists in reality, an extensive evaluation in the real environment with our speech recognition grammar is necessary to find out how and if the utterances differ from what we found.

The prompts for the utterances should neither bias nor prime the participants. Our prompts partly used words which the subjects reused in their utterances and it would be interesting to see what the utterances look like if the prompts are neutral although this is difficult to realize.

5.5 Chapter Summary

We presented the results from our evaluation, explaining the methods we applied. In the results we demonstrated which kinds of utterances were suggested by the participants giving a selection of examples. The discussion showed that the existing grammar rules can be modified easily and quickly to include between 15% and 30% of the new utterances. Then, we presented some remarks made by the participants about which additional features would be desirable in the bathroom and explained to which extent their wishes can be taken care of.

Our evaluation demonstrated that although our grammars covers some of the participants' utterances, there is an abundance of utterances which we have not considered yet. However, a part of the new utterances can be integrated into the existing grammars easily by modifying only a few rules. To see whether our extensions are successful another evaluation will be necessary.

Discussion and Future Work

In the previous chapters, we have presented our work and all issues connected with it. This chapter summarizes our findings before we give an overview of features which we want to implement in the future. Furthermore, we provide an extensive example of integrating NLU into the grammars.

6.1 Discussion

For speech recognition systems, developers have to choose one of two language models: the statistical n-gram approach or rule-based grammars. While the statistical approach is well suited for open-domain speaker-dependent dictation tasks, rule-based grammars can often be found in closed-domain speaker-independent dialogue systems.

As the speech recognition system for this thesis is used in an ODP application, we presented an implementation of modular rule-based grammars suited for its ontology-based architecture. The main focus of this work lay on designing a modular architecture for the grammars which increased recognition performance and led to a low WER.

The modularity is based on one grammar file containing all general information. All other files are linked to this general file (`ami_general.grm`).

As we required concrete situations for which we could develop the grammars, we built them for an ambient intelligence bathroom environment which serves as a use case for the EU project IKS. For the bathroom, members of the project and a bathroom furnishing company devised eleven situations which exhibit the project software's features. For our work, we took the top three situations out of the eleven which had ranked highest in a series of user studies prior to this work. We split the three situations up into six parts: weather requesting, event recommendation, ticket ordering, music navigation, news display and listening to the radio.

We analysed each of the three situations and extracted a state diagram for each of the six parts from them. The state diagram was used for creating

a dialogue which is typical for this part. We took the dialogue as basis for finding semantically similar utterances for each member in the dialogue. From the similar utterances, we created variables to combine some utterances into one more general expressions. This way, utterances like *please show me the weather*, *please show the weather forecast*, *please show me the weather prognosis* and *show the weather forecast* could be combined into

$$?please\ show\ ?me\ ([NP_WEATHER\ NP_FORECAST]) \quad (6.1)$$

with the question mark indicating optionality and the square brackets a disjunction (the round brackets around the square brackets are a syntactic requirement). NP_WEATHER stands for *the weather*, NP_FORECAST can be replaced by *the ?weather FORECAST* and FORECAST stands for *forecast*, *outlook* or *prospects*. We gave further information on grammars in chapter 3.

For all sentence constructs and collective items (e. g. towns, weekdays or numbers), we created a larger file which is integrated into every grammar file for the six parts. In addition, we wrote a grammar responsible for accepting times on the clock only. We excluded these temporal expressions from the general file as it was required for only two of the six small situations. Thus, we avoided overgeneration: numbers articulated by a user were less likely to be recognized as a temporal expression if this grammar was not activated. A more detailed description of the procedure as well as a list of requirements which our grammars fulfil can be found in chapter 4.

We conducted extensive testing of the grammars with tools provided by Nuance Communications, the company whose recognizer we use. With one of them we were able to determine whether the grammars we wrote will accept the sentences we previously devised, i. e. test the accuracy of the grammars. The other tool also helped to discover mistakes in the grammars: it received a grammar as input and returned a given number of utterances which the grammar can generate. With that result, we were able to eliminate any nonsensical output from the grammars by modifying them accordingly.

In addition to the testing, we performed an extensive evaluation for finding out whether the utterances we devised would be used by others as well. However, during the analysis of the results we detected that only 2% of the utterances by participants of the evaluation were covered by our grammars. By modifying the grammars slightly, we were able to establish a coverage between 7 and 30%.

In the following section, we illustrate features which can be integrated into the grammars in the near future for further improvement of the coverage as well as an integration of NLU.

6.2 Future Work

In future versions of the existing grammars, we want to consider disfluencies (see section 4.5). In our opinion, it requires the insertion of another process-

ing step which removes the disfluencies before the utterances are analysed with the language model. The inclusion of disfluencies is beneficial for the system as it is able to recognize utterances even if the user does not articulate them fluently. With it, we increase the performance of the recognizer.

The evaluation has shown that most of the participants expect the dialogue system to accept short commands with only a few words. As they are not included in the current grammars, a future revision will include them. This will roughly double the coverage of the utterances from participants in the evaluation.

It is planned to write grammars for all eleven bathroom situations. As the grammars containing general information already exist, only the situation specific grammars need to be designed for the remaining eight situations. We recommend to follow the procedure described in section 4.3 and attend to the requirements defined in section 4.5.

With our work we want to provide a basis for other speech recognition grammars and in the next development steps, we want to present a grammar structure with general sentence and question patterns in such a way that a grammar can easily be ported to other domains. Kaiser presented a promising approach for questions which we plan to adopt [KAISER, 2004].

Before the grammar can be integrated fully into a complete dialogue system, we have to complete the NLU tags in the grammar files. For this, the utterances require a computer-understandable ontological representation that the spoken input can be mapped to.

The NLU we integrate will provide a detailed feedback for the user when an utterance has not been understood by the dialogue system. The system will be able to specifically ask for the slots which have not been filled by an utterance. In addition, we want to give better feedback to the user when an utterance is not covered by the system's technology. We envision a feedback that expresses that the system understands the user but cannot handle the query appropriately. Another option is the inclusion of rapid prompting, as proposed by Cohen et al. [COHEN ET AL., 2004] which refers to the method of indicating a misunderstanding by saying "I'm sorry?" or "What's that?". If the system fails to understand the user twice, a standard failure message is issued.

In the following section, we draft a model NLU for three utterances taken from our grammars.

Natural Language Understanding

For the ODP framework, an NLU syntax using extended type feature structures (eTFS) [SEMVOX GMBH, 2008] has been defined. As the NLU component cannot handle disjunctions easily, we slightly modified the utterance below. All phrases have to be adjusted for integrating the NLU component.

We show an example for an utterance from situation 1:

```

<utterance name="S_SHOW_WEATHER_LOCATION">
  <phrases>
    <phrase>VP_WOULD_LIKE see [WEEKDAY
      N_LOCATION]</phrase>
    <phrase>VP_WOULD_LIKE see N_LOCATION
      PP_WEEKDAY</phrase>
    <phrase>VP_WOULD_LIKE see N_LOCATION PP_WEEKDAY
      TIME_OF_DAY</phrase>
  </phrases>
  <semantic-interpretation>
    <object type="WeatherRequest">
      <slot name="hasInitiator">
        <object type="User"/>
      </slot>
      <slot name="hasContent">
        <object type="location">
          <slot name="hasContent">
            <variable name="N_LOCATION"/>
          </slot>
        </object>
        <object type="time">
          <slot name="hasContent">
            <variable name="PP_WEEKDAY"/>
          </slot>
        </object>
      </slot>
    </object>
  </semantic-interpretation>
</utterance>

```

Please see appendix D for examples from situations 6 and 11.

The `semantic-interpretation` tag contains all information that is connected to the ontology. Objects define a complex object type and represent the type of the eTFS. Each object has at least one slot where the feature of the given object type is defined. Variables from the utterances are passed on in a `variable` tag. According to the ODP documentation [SEMVOX GMBH, 2009], variables have to be defined in the `named-entity` part of the grammar, otherwise they cannot be handled by the ontology.

The NLU structure is a direct representation of the concept structure from the ontology. Before we can integrate NLU for all utterances in the grammars, the IKS project's ontology has to be completed.

References

- [BAKER, 1975] Baker, J. 1975. The DRAGON system—An overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1), 24–29.
- [BAR-HILLEL ET AL., 1961] Bar-Hillel, Yehoshua, Perles, M., & Shamir, E. 1961. On Formal Properties of Simple Phrase Structure Grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14, 143–172.
- [BAUM & EAGON, 1967] Baum, L. E., & Eagon, J. A. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3), 360–363.
- [BAUM & PETRIE, 1966] Baum, L. E., & Petrie, T. 1966. Statistical inference for probabilistic functions of finite-state Markov chains. *Annals of Mathematical Statistics*, 37(6), 1554–1563.
- [BESSER, 2006] Besser, Jana. 2006. A Corpus-based Approach to the Classification and Correction of Disfluencies in Spontaneous Speech. Saarland, Germany.
- [CHOMSKY, 1956] Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3), 113–124.
- [CHOMSKY, 1963] Chomsky, Noam. 1963. Formal Properties of Grammars. Pages 323–418 of: Luce, R. D., Bush, R., & Galanter, E. (eds), *Handbook of Mathematical Psychology*, vol. 2. New York: Wiley.
- [CHOMSKY & MILLER, 1963] Chomsky, Noam, & Miller, George A. 1963. Introduction to the Formal Analysis of Natural Languages. Pages 269–322 of: Luce, R. D., Bush, R., & Galanter, E. (eds), *Handbook of Mathematical Psychology*, vol. 2. New York: Wiley.
- [COHEN ET AL., 2004] Cohen, K. B., Hunter, L., Dubitzky, Werner, & Azuaje, Francisco. 2004. Natural language processing and systems biology. Springer. Pages 147–174.
- [COOLEY & TUKEY, 1965] Cooley, James W., & Tukey, John W. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90), 297–301.
- [COPESTAKE & FLICKINGER, 2000] Copestake, Ann, & Flickinger, Dan. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. Pages 591–600 of: *Proceedings of LREC 2000*. Stanford University.
- [DAVIES, 2009] Davies, Mark. 2009. The 385+ Million Word Corpus of Contemporary American English (1990-2008+): Design, Architecture, and Linguistic Insights. *International Journal of Corpus Linguistics*, 14, 159–190.

- [DAVIS ET AL., 1952] Davis, K. H., Biddulph, R., & Balashek, S. 1952. Automatic recognition of spoken digits. *JASA*, 24(6), 637–642.
- [DENES, 1959] Denes, P. 1959. the design and operation of the mechanical speech recognizer at University College London. *Journal of the British Institution of Radio Engineers*, 19(4), 219–234.
- [FRY, 1959] Fry, D. B. 1959. Theoretical Aspects of mechanical speech recognition. *Journal of the British Institution of Radio Engineers*, 19(4), 211–218.
- [GERMESIN, 2006] Germesin, Sebastian. 2006 (January). *Spracherkennung mit dynamisch geladenen, spezifischen Akustikmodellen*. Saarland, Germany.
- [GERNSBACHER, 1991] Gernsbacher, M. A. 1991. Comprehending conceptual anaphors. *Language and Cognitive Processes*, 6, 81–105.
- [GIRAUDO & BAGGIA, 2009] Giraudo, Enrico, & Baggia, Paolo. 2009. *EVALITA 2009: Loquendo Spoken Dialog System*.
- [GRISHMAN ET AL., 1994] Grishman, Ralph, Macleod, Catherine, & Meyers, Adam. 1994. *Complex Syntax: Building a Computational Lexicon*.
- [HOPCROFT & ULLMAN, 1979] Hopcroft, John E., & Ullman, Jeffrey D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley.
- [HUANG ET AL., 1992] Huang, Xuedong, Allewa, Fileno, Hon, Hsiao-Wuen, Hwang, Mei-Yuh, & Rosenfeld, Ronald. 1992. The SPHINX-II Speech Recognition System: An Overview. *Computer, Speech and Language*, 7, 137–148.
- [HUANG ET AL., 2001] Huang, Xuedong, Acero, Alex, & Hon, Hsiao-Wuen. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Upper Saddle River, NJ, USA: Prentice Hall PTR. Foreword By-Reddy, Raj.
- [I. P. A., 1999] I. P. A. (ed). 1999. *Handbook of the International Phonetic Association : A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.
- [JANZEN ET AL., 2010] Janzen, Sabine, Maass, Wolfgang, Kowatsch, Tobias, Filler, Andreas, Romanelli, Massimo, Germesin, Sebastian, Becker, Tilman, Laleci, Gokce B., Ocalan, Cagdas, & Alpay, Erdem. 2010 (02). *AMI Case Requirements Specification*. Tech. rept. Interactive Knowledge Stack for Semantic Content Management Systems.
- [JELINEK, 1997] Jelinek, Frederick. 1997. *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press.
- [JURAFSKY & MARTIN, 2008] Jurafsky, Daniel, & Martin, James H. 2008. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. 2 edn. Prentice Hall.
- [KAISSER, 2004] Kaiser, Michael. 2004. *Question Answering by Searching Large Corpora with Linguistic Methods*. M.Phil. thesis, Saarland University.
- [KLEENE, 1956] Kleene, S. C. 1956. Representation of events in nerve nets and finite automata. Pages 3–41 of: Shannon, Claude, & McCarthy, John (eds), *Automata Studies*. Princeton, NJ: Princeton University Press.
- [KLUGER-KRUSE, 1987] Kluger-Kruse, M. 1987. *Computer Phonetic Alphabet*. *ES-PRIT Linguistic Analysis of the European Languages*, 7.
- [LEE & REDDY, 1988] Lee, Kai-Fu, & Reddy, Raj. 1988. *Automatic Speech Recognition: The Development of the Sphinx Recognition System*. Norwell, MA, USA: Kluwer Academic Publishers.
- [LEWIS & PAPADIMITRIOU, 1981] Lewis, Harry, & Papadimitriou, Christos. 1981. *Elements of the Theory of Computation*. Englewood Cliffs, NJ: Prentice-Hall.
- [LOWERRE, 1976] Lowerre, Bruce T. 1976. *The Harpy Speech Recognition System*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

- [LUMENVOX, 2010] LumenVox. 2010. Speech Recognition White Paper, Grammars.
- [MARKOV, 1913] Markov, A. A. 1913. Essai d'une recherche statistique sur le texte du roman "Eugene Onegin" illustrant la liaison des epreuve en chain ('Example of a statistical investigation of the text of "Eugene Onegin" illustrating the dependence between samples in chain'). *Izvestia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.-Petersbourg)*, 7, 153–162. English translation by Morris Halle, 1956.
- [NUANCE, 2003] Nuance. 2003. Nuance Speech Recognition System 8.5: Grammar Developer's Guide. 1005 Hamilton Avenue, Menlo Park, California 94025 U.S.A.
- [NUANCE, 2009] Nuance. 2009. Nuance Recognizer White Paper. White Papers.
- [PFALZGRAF ET AL., 2008] Pfalzgraf, Alexander, Pflieger, Norbert, Schehl, Jan, & Steigner, Jochen. 2008 (10). ODP Ontology-based Dialogue Platform White Paper. SemVox GmbH, Fuchstälchen 2, 66123 Saarbrücken.
- [PFLEGER, 2007] Pflieger, Norbert. 2007. Context-based multimodal interpretation : an integrated approach to multimodal fusion and discourse processing. Ph.D. thesis, Saarland University.
- [POLLARD & SAG, 1994] Pollard, Carl, & Sag, Ivan A. 1994. Head-Driven Phrase Structure Grammar. Chicago: University of Chicago Press.
- [RABINER, 1989] Rabiner, Lawrence R. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), 257–286.
- [RABINER & JUANG, 1993] Rabiner, Lawrence R., & Juang, Biing-Hwang. 1993. Fundamentals of Speech Recognition. Englewood Cliffs, NJ: Prentice Hall.
- [ROMANELLI ET AL., 2010] Romanelli, Massimo, Germesin, Sebastian, Damjanovic, Violeta, Adamou, Alessandro, Janzen, Sabine, Filler, Andreas, Conconi, Alex, Kowatsch, Tobias, & Becker, Tilman. 2010 (7). Model of Knowledge Based Interaction. Tech. rept. Interactive Knowledge Stack for Semantic Content Management Systems.
- [SCHEHL ET AL., 2008] Schehl, Jan, Pfalzgraf, Alexander, Pflieger, Norbert, & Steigner, Jochen. 2008. The babbleTunes system: talk to your ipod! Pages 77–80 of: *ICMI '08: Proceedings of the 10th international conference on Multimodal interfaces*. New York, NY, USA: ACM.
- [SCHUKAT-TALAMAZZINI, 1995] Schukat-Talamazzini, Ernst Günter. 1995. Automatische Spracherkennung. Vieweg Verlag.
- [SEMVOX GMBH, 2008] SemVox GmbH. 2008 (12). ODP Core Documentation – PATE. Tech. rept. SemVox GmbH.
- [SEMVOX GMBH, 2009] SemVox GmbH. 2009 (02). Dokumentation: Das ODP-Flex Grammatik- Framework. SemVox GmbH, Fuchstälchen 2, 66123 Saarbrücken.
- [STAHL ET AL., 1996] Stahl, Holger, Müller, Johannes, & Lang, Manfred. 1996. An Efficient Top-Down Parsing Algorithm For Understanding Speech By Using Stochastic Syntactic And Semantic Models. Pages 397–400 of: *Syntactic and Semantic Models, Proc. ICASSP*.
- [STEVENS ET AL., 1937] Stevens, S. S., Volkman, J., & Newman, E. B. 1937. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3), 208–208.
- [TAYLOR, 2009] Taylor, Paul. 2009. Text-to-Speech Synthesis. Leiden: Cambridge Univ. Press.
- [WAHLSTER, 2000] Wahlster, Wolfgang (ed). 2000. *Verbmobil, Foundations of Speech-to-Speech Translation*. Hardcover edn. Springer Verlag.

- [WALKER ET AL., 2004] Walker, Willie, Lamere, Paul, Kwok, Philip, Raj, Bhiksha, Singh, Rita, Gouvea, Ro, Wolf, Peter, & Woelfel, Joe. 2004. Sphinx-4: A flexible open source framework for speech recognition. Tech. rept. Carnegie Mellon University.
- [WANG & JU, 2004] Wang, Ye-Yi, & Ju, Yun-Cheng. 2004. Creating Speech Recognition Grammars from Regular Expressions for Alphanumeric Concepts. Pages 2161–2164 of: International Conference on Spoken Language Processing.
- [WANG ET AL., 2000] Wang, Ye-Yi, Mahajan, Milind, & Huang, Xuedong. 2000. A unified context-free grammar and n-gram model for spoken language processing. Pages 1639–1642 of: in International Conference of Acoustics, Speech, and Signal Processing.
- [YNGVE, 1960] Yngve, Victor H. 1960. A Model and an Hypothesis for Language Structure. Proceedings of the American Philosophical Society, 104, 444–466.

Appendices

A

Empty .grm file

```
<grammar lang="en" file-name="ami_general.grxml">
  <!-- UTTERANCES -->
  <utterance name="">
    <phrases>
    <phrase></phrase>
    <phrase></phrase>
    </phrases>
    <semantic-interpretation>
      <object type="">
        <slot name="">
          </slot>
        </object>
      </semantic-interpretation>
    </utterance>

  <named-entity-grammar>
    <file-name> </file-name>
    <nlu-slot> </nlu-slot>
    <named-entities>
    <named-entity>
      <phrase> </phrase>
      <nlu-info> </nlu-info>
    </named-entity>
    <named-entity>
      <phrase> </phrase>
      <nlu-info> </nlu-info>
    </named-entity>
    </named-entities>
    <semantic-interpretation>
  <object>
    <slot>
      <object/>
    </slot>
```

```
<slot>
  <variable/>
</slot>
</object>
</semantic-interpretation>
<path-to-phrase> </path-to-phrase>
</named-entity-grammar>

<semantic-grammar-rule>
  <nlu-slot> </nlu-slot>
  <phrases>
<phrase-mapping>
  <phrase> </phrase>
  <nlu-info> </nlu-info>
</phrase-mapping>
  </phrases>
<semantic-interpretation>
<object>
  <slot>
    <object/>
  </slot>
  <slot>
    <variable/>
  </slot>
</object>
  </semantic-interpretation>
</semantic-grammar-rule>

<gsl>
</gsl>

<lexicon>
  <entry key="">
    <definition value="" />
  </entry>
</lexicon>
</grammar>
```

B

Evaluation Questionnaire

Introduction

Ihr steht in eurem neuen Badezimmer. Es ist mit dem Internet und einer Datenbank verbunden. Hierber könnt ihr diverse Informationen abfragen. Ich möchte dich/euch bitten, mir für die folgenden Situationen Stz (auf Englisch) zu nennen, mit denen ihr die gewünschten Informationen abfragen würdet.

Experiment text

It's Thursday morning, you are in the bathroom preparing yourself for the. In order to know what to wear you want to see how the weather is going to be today in your city, Saarbrücken. What would you say?

1. What would you say to get the weather for another day?
You are going to another town, let's say, London, today.
What would you say to get the weather for London? (assuming that you can see the weather for Saarbrücken already)
2. You wish to change the city for which the forecast is shown.
What would you say?
And if you wanted to change the day?

We will look at the following situation from two different perspectives.

Now that you know what the weather is going to be like where you are, you can proceed to planning the rest of the day. You can see your calendar. The bathroom asks "Your calendar is empty tonight. Do you want to see a list of events for tonight?"

You receive a list of events. The bathroom asks:

1. Please pick an item from the list.
2. Which show do you want to see?

While browsing through your calendar, you discover that you have nothing planned for tonight. Please imagine the following scenarios.

You want to know which shows and events are on tonight. How would you ask?

You know that you want to go to the cinema. You want to know which movies are on tonight. How would you ask?

You want to find out which plays are on in the opera/theatre tonight. How would you ask?

You want to go to a specific cinema tonight. How would you specify the cinema and that you would like to see that cinema's programme?

You want to go to watch a particular play/movie. How would you ask?

You have decided that you want to go to Cinestar and watch Avatar at half past 7 p.m.. As you know that the cinema is full tonight, you want to reserve tickets and seats.

1. The bathroom suggests seats for you and asks "Are the seats okay for you?"
 - a) yes.
 - b) no. How would you ask for changing the seats?
2. You want to pick the seats' location. How would you ask?

The bathroom suggests seats for you and asks "Are the seats okay for you?"

 - a) yes.
 - b) no. How would you ask for changing the seats?

It's Saturday. You have slept in and after your partner is done in the bathroom, you enter. Your partner's music is still playing. You want it to stop and listen to your favourite music. What would you say?

Yesterday night, you listened to a good album which you would like to pick up where you left it yesterday. How would you say that?

What would you say to navigate through a playlist?

How would you ask for playing a particular song?

How would you ask for the next or the previous song?

After some time, you want to take a shower and watch your personalized news on a screen in the shower. How would you tell the music to stop?

How would you start your news broadcast?

How would you ask for a news broadcast from a specific TV-Channel?

On Monday morning, you are listening to the news on the radio. The radio is too quiet. How do you change the radio's volume?

You don't like the radio station. What would you say to change it?

You want to take a shower and want to watch the remaining news there. Your bathroom can automatically switch the news format from radio to text and pictures. What would you say in order for the news to be displayed in the shower?

C

Grammars

C.1 General

```
<grammar lang="en" file-name="ami_general.grxml">
  <utterance name="S_DECLINE">
    <phrases>
      <phrase>DECLINE ?THANK</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_DECIDE_ORDER">
    <phrases>
      <phrase>yes ?please</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <gsl>
    VP_WOULD_LIKE [(i ([V_MOD V_MOD_SHORT]) to)]
    FOR_DAY [(for ([WEEKDAY ?TIME_OF_DAY] this that
      LOC_ADV))] ]
    NP_NUMBER_ORDINAL [(the NUMBER_ORDINAL)]
    NP_ADJ_RL [(to the ADJ_RL)]
    PP_LOCATION [(in NE_LOCATION)]
    PP_WEEKDAY [ ( ( [ on for]) WEEKDAY ) ]
    BATH [bath bathroom]
    NEG_NECESS [(not necessary)]
    DECLINE [(no NEG_NECESS) no NEG_NECESS]
```

```

NE_LOCATION [salzburg saarbruecken (sankt gallen)
             rome]
V_MOD [want (would like) wish]
V_MOD_SHORT [(d like)]
ADJ_RL [right left]
PP_RL [ (on the ADJ_RL) ]
PP_MID [ (in the middle) ]
THANK [thanks THANKYOU]
THANKYOU [(thank you)]
DIRECTION [top bottom]
LOC_ADV [here there]
WEEKDAY [monday tuesday wednesday thursday friday
         saturday sunday]
TIME_OF_DAY [morning midday noon noontime afternoon
            evening]
LETTER [a b c d e f g h i j k l m n o p q r s t u v
       w x y z]
GREETBATH[( ([GREETING LEAVE]) ?BATH)]
GREETING [MORNING hello WAKE]
LEAVE [goodbye bye farewell (see you) (good night)]
MORNING [(good morning)]
WAKE [(wake up)]
ORDINAL_ONE_TO_NINE [first second third fourth fifth
                    sixth seventh eighth ninth]
ORDINAL_TEN_TO_NINETEEN [tenth eleventh twelfth
                        thirteenth fourteenth fifteenth sixteenth
                        seventeenth eighteenth nineteenth]
ORDINAL_TWENTY_TO_NINETY [twentieth thirtieth
                          fortieth fiftieth sixtieth seventieth eightieth
                          ninetieth]
CARDINAL_ONE_TO_NINE [one two three four five six
                     seven eight nine]
CARDINAL_TEN_TO_NINETEEN [ten eleven twelve thirteen
                          fourteen fifteen sixteen seventeen eighteen
                          nineteen]
CARDINAL_TWENTY_TO_NINETY [twenty thirty forty fifty
                           sixty seventy eighty ninety]
COMPLEX_ORDINAL [(CARDINAL_TWENTY_TO_NINETY
                 ORDINAL_ONE_TO_NINE)]
COMPLEX_CARDINAL [(CARDINAL_TWENTY_TO_NINETY
                  CARDINAL_ONE_TO_NINE)]
NUMBER_CARDINAL [COMPLEX_CARDINAL
                 CARDINAL_ONE_TO_NINE CARDINAL_TEN_TO_NINETEEN
                 CARDINAL_TWENTY_TO_NINETY]
NUMBER_ORDINAL [COMPLEX_ORDINAL ORDINAL_ONE_TO_NINE
                ORDINAL_TEN_TO_NINETEEN ORDINAL_TWENTY_TO_NINETY]
</gsl>

<lexicon>

```

```
<entry key="saarbruecken">
  <definition value="za:bRYk@n" />
</entry>
<entry key="salzburg">
  <definition value="za1C/bu@g" />
</entry>
<entry key="sankt gallen">
  <definition value="zaNkt gAl@n" />
</entry>
</lexicon>
</grammar>
```

C.2 Time

```

<grammar lang="en" file-name="ami_temp.grxml">
  <utterance name="TEST">
    <phrases>
      <phrase>PP_TEMP</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <gsl>
    PP_TEMP [(at TEMP)]
    TEMP [FULL_HOUR HALF_HOUR HALF_HOUR_CLOCK
          MINUTE_HOUR DIGITAL_24]
    FULL_HOUR [HOUR_SHARP HOUR_CLOCK_SHARP HOUR_CLOCK]
    HOUR_SHARP [(HOURS_AM AM_PM ?sharp)]
    HOUR_CLOCK_SHARP [(HOURS_AM OCLOCK sharp)]
    HOUR_CLOCK [(HOURS_AM OCLOCK)]
    HALF_HOUR [(half ?past HOURS_AM AM_PM)]
    HALF_HOUR_CLOCK [(half ?past HOURS_AM ?OCLOCK)]
    MINUTE_HOUR [(MINUTES_QUART TO_PAST HOURS_AM ?AM_PM)]
    MINUTES_QUART [MINUTES_30 quarter]
    DIGITAL_24 [(HOURS MINUTES)]
    HOURS [HOURS_AM HOURS_24]
    MINUTES [MINUTES_30 MINUTES_60]
    MINUTES_30 [five ten fifteen twenty twentyfive]
    MINUTES_60 [thirty thirtyfive forty fortyfive fifty
                fiftyfive]
    HOURS_AM [one two three four five six seven eight
              nine ten eleven twelve]
    HOURS_24 [twelve thirteen fourteen fifteen sixteen
              seventeen eighteen nineteen twenty twentyone
              twentytwo]
    OCLOCK [(oclock)]
    TO_PAST [to past]
    AM_PM [(am) (pm)]
  </gsl>
</grammar>

```

C.3 Situation 1

C.3.1 weather

```

<grammar lang="en" file-name="sit1_weather_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>

  <utterance name="Q_WEATHER">
    <phrases>
      <phrase>Q_HOW_WEATHER</phrase>
      <phrase>S_SHOW_WEATHER</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="Q_HOW_WEATHER_LOCATION">
    <phrases>
      <phrase>Q_HOW_WEATHER ?today here</phrase>
      <phrase>Q_HOW_WEATHER [PP_WEEKDAY today tomorrow
        tonight like] ?PP_LOCATION</phrase>
      <phrase>Q_HOW_WEATHER [PP_WEEKDAY WEEKDAY this
        tomorrow] TIME_OF_DAY ?PP_LOCATION</phrase>
      <phrase>Q_HOW_WEATHER PP_LOCATION ?[today tomorrow
        tonight (on WEEKDAY)]</phrase>
      <phrase>Q_HOW_WEATHER PP_LOCATION [PP_WEEKDAY
        WEEKDAY this tomorrow] TIME_OF_DAY</phrase>
      <phrase>S_SHOW_WEATHER for [WEEKDAY NE_LOCATION]
        </phrase>
      <phrase>S_SHOW_WEATHER for WEEKDAY TIME_OF_DAY
        ?PP_LOCATION </phrase>
      <phrase>S_SHOW_WEATHER for WEEKDAY PP_LOCATION
        </phrase>
      <phrase>S_SHOW_WEATHER for NE_LOCATION this
        TIME_OF_DAY</phrase>
      <phrase>S_SHOW_WEATHER for NE_LOCATION PP_WEEKDAY
        ?TIME_OF_DAY</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_SHOW_WEATHER_LOCATION">
    <phrases>
      <phrase>VP_WOULD_LIKE see [WEEKDAY
        NE_LOCATION]</phrase>
    </phrases>
  </utterance>

```

```

    <phrase>VP_WOULD_LIKE see NE_LOCATION
      PP_WEEKDAY</phrase>
    <phrase>VP_WOULD_LIKE see NE_LOCATION PP_WEEKDAY
      TIME_OF_DAY</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<utterance name="S_CHANGE_WEATHER_LOCATION">
  <phrases>
    <phrase>and ?now ?in NE_LOCATION [ today tomorrow
      PP_WEEKDAY tonight ] </phrase>
    <phrase>and ?now [PP_WEEKDAY today tomorrow tonight]
      PP_LOCATION </phrase>
    <phrase>and for NE_LOCATION</phrase>
    <phrase>QUESTION about NE_LOCATION</phrase>
    <phrase>QUESTION about [today tonight]
      ?PP_LOCATION</phrase>
    <phrase>QUESTION about [(on WEEKDAY) tomorrow]
      ?TIME_OF_DAY ?PP_LOCATION</phrase>
    <phrase>QUESTION about this TIME_OF_DAY
      ?PP_LOCATION</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<gsl>
  Q_HOW_WEATHER [(QUESTION is NP_WEATHER)]
  QUESTION [ how what ]
  S_SHOW_WEATHER [(?please SHOW_GIVE ?me ([NP_WEATHER
    NP_FORECAST]))]
  SHOW_GIVE [ show give ]
  NP_WEATHER [(the weather)]
  NP_FORECAST [(the ?weather FORECAST)]
  N_FORECAST [(weather ?FORECAST)]
  FORECAST [forecast outlook prospects prognosis
    report]
</gsl>
</grammar>

```

C.3.2 event recommendation

```

<grammar lang="en" file-name="sit1_event_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>
<import location="grammar/ami_temp.grm" locator="jar"/>

  <utterance name="S_ASK_EVENT_GENERAL">
    <phrases>
      <phrase>?yes ?please show ?me where i can go</phrase>
      <phrase>where can i go</phrase>
      <phrase>?please show ?me NP_POSS_EVENT for this
        TIME_OF_DAY</phrase>
      <phrase>what is [happening on (going on)] in
        NE_LOCATION</phrase>
      <phrase>what is [happening on (going on)] in
        ?(NE_LOCATION) [today tomorrow tonight]</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_ASK_EVENT_SPECIFIC">
    <phrases>
      <phrase>?please show ?me PP_EVENT_LOC_PLURAL
        ?PP_NEIGHBOUR</phrase>
      <phrase>what can i V_SEE [tonight (this
        TIME_OF_DAY)]</phrase>
      <phrase>what is on in NP_EVENT_LOC [(this
        TIME_OF_DAY) (on WEEKDAY)]</phrase>
      <phrase>what HAPPEN_PLAY at NP_EVENT_LOC</phrase>
      <phrase>what is on at NP_EVENT_LOC</phrase>
      <phrase>what HAPPEN_PLAY in the
        EVENT_LOCATION_PLURAL</phrase>
      <phrase>VP_WOULD_LIKE go to [NP_EVENT_LOC
        NE_EVENT_LOC]</phrase>
      <phrase>show ?me NE_EVENT_LOC s programme
        please</phrase>
      <phrase>?please show ?me NE_EVENT_LOC s
        programme</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_ASK_EVENT_DEICTIC">
    <phrases>
      <phrase>VP_WOULD_LIKE go there</phrase>
    </phrases>
  </utterance>

```

```

<semantic-interpretation>
...
</semantic-interpretation> </utterance>

<utterance name="S_ASK_SHOW">
  <phrases>
    <phrase>VP_WOULD_LIKE ?go see NE_SHOW</phrase>
    <phrase>VP_WOULD_LIKE ?go see NE_SHOW
      PP_TEMP</phrase>
    <phrase>where can i V_SEE NE_SHOW</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<utterance name="S_CHOOSE_EVENT_FROM_LIST">
  <phrases>
    <phrase>VP_WOULD_LIKE V_SEE the NUMBER_ORDINAL ?one
      ?(from the DIRECTION)</phrase>
    <phrase>VP_WOULD_LIKE V_SEE the one PP_temp</phrase>
    <phrase>VP_WOULD_LIKE V_SEE [NE_SHOW (the
      NUMBER_ORDINAL) (this one)]</phrase>
    <phrase>the NUMBER_ORDINAL ?one ?please</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<gsl>
  HAPPEN_PLAY [(is ( [happening playing] ) )]
  N_NEIGHBOUR [area neighbourhood]
  ADJ_CLOSE[closeby nearby]
  EVENT_LOCATION [cinema (movie theatre) playhouse
    theatre opera]
  EVENT_LOCATION_PLURAL [cinemas (movie theatres)
    playhouses theatres operas]
  EVENT [movie play opera]
  EVENT_PLURAL [movies plays operas]
  PP_EVENT_LOC_PLURAL [ MOV_CINEMAS PLAY_THEATERS
    OPERA_OPERAS ]
  MOV_CINEMAS [(movies in cinemas)]
  PLAY_THEATERS [(plays in ([playhouses theatres
    operas]))]
  OPERA_OPERAS [(operas in opera houses)]
  NE_EVENT_LOC[scala (camera zwo) madison cinestar]
  NE_SHOW [avatar (porgy and bess) hamlet]
  NP_LIST [(the list)]

```

```
NP_EVENT_LOC [(the EVENT_LOCATION)]
NP_POSS_EVENT [(?the ?possible events)]
PP_NEIGHBOUR [(in the ([area neighbourhood]))]
PP_CLOSE_EVENT_LOC [(in ADJ_CLOSE EVENT_LOC_PLURAL)]
PP_NEAR_EVENT_LOC [(in EVENT_LOC_PLURAL
PP_NEIGHBOUR)]
V_SEE [see watch view]
V_PRESENT [show present suggest]
</gsl>
</grammar>
```

C.3.3 ticket ordering

```

<grammar lang="en" file-name="sit1_ticket_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>

  <utterance name="S_CHOOSE_SEATS">
    <phrases>
      <phrase>LOC_ADV ?please</phrase>
      <phrase>these ?([seats ones]) ?please</phrase>
      <phrase>this ?([seat one]) ?please</phrase>
      <phrase>VP_WOULD_LIKE sit LOC_ADV ?please</phrase>
      <phrase>VP_WOULD_LIKE sit PP_ROW ?please</phrase>
      <phrase>VP_WOULD_LIKE sit PP_ROW [ PP_RL PP_MID
        ]</phrase>
      <phrase>VP_WOULD_LIKE sit in LETTER
        NUMBER_CARDINAL</phrase>
      <phrase>VP_WOULD_LIKE sit in LETTER NUMBER_CARDINAL
        and NUMBER_CARDINAL</phrase>
      <phrase>give me seats LETTER NUMBER_CARDINAL ?(and
        NUMBER_CARDINAL)</phrase>
      <phrase>reserve ?seats LETTER NUMBER_CARDINAL ?(and
        NUMBER_CARDINAL)</phrase>
      <phrase>VP_WOULD_LIKE sit a bit more to the
        ADJ_RL</phrase>
      <phrase>can i take ?seat LETTER NUMBER_CARDINAL
        ?(and NUMBER_CARDINAL)</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_CHANGE_SEATS">
    <phrases>
      <phrase>no ?please take [these (the ones to the
        ADJ_RL)]</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <gs1>
    PP_ROW [in the NUMBER_ORDINAL row]
  </gs1>
</grammar>

```

C.4 Situation 6

C.4.1 music

```

<grammar lang="en" file-name="sit6_music_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>

  <utterance name="S_PLAY_OWN_MUSIC">
    <phrases>
      <phrase>[play (continue playing)] MUSIC from my
        collection ?please</phrase>
      <phrase>please [play (continue playing)] MUSIC from
        my collection</phrase>

      <phrase>[play start] my ?favourite [MUSIC playlist]
        ?please</phrase>
      <phrase>please [play start] my ?favourite [MUSIC
        playlist]</phrase>

      <phrase>switch to my COLLECTION ?please</phrase>
      <phrase>please switch to my COLLECTION</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_PLAY_ALBUM_OR_ARTIST">
    <phrases>
      <phrase>[play start] MUSIC from [ALBUM ARTIST]
        ?please</phrase>
      <phrase>please [play start] MUSIC from [ALBUM
        ARTIST]</phrase>

      <phrase>play [ALBUM ARTIST] ?please</phrase>
      <phrase>please play [ALBUM ARTIST]</phrase>

      <phrase>play [ALBUM ARTIST GENRE] ?please</phrase>
      <phrase>please play [ALBUM ARTIST GENRE]</phrase>

    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_PLAY_SONG">
    <phrases>

```

```

    <phrase>[play start] SONG from [ALBUM ARTIST]
      ?please</phrase>
    <phrase>please [play start] SONG from [ALBUM
      ARTIST]</phrase>

    <phrase>[play start] SONG by ARTIST ?please</phrase>
    <phrase>please [play start] SONG by ARTIST</phrase>

    <phrase>play NP_ORD_SONG from ALBUM ?please</phrase>
    <phrase>please play NP_ORD_SONG from ALBUM</phrase>

    <phrase>play SONG ?please</phrase>
    <phrase>please play SONG</phrase>

  </phrases>
  <semantic-interpretation>
    ...
  </semantic-interpretation>
</utterance>

<utterance name="S_PLAYLIST_NAVIG">
  <phrases>
    <phrase>play NP_ORD_SONG from the [list playlist]
      ?please</phrase>
    <phrase>please play NP_ORD_SONG from the [list
      playlist]</phrase>

    <phrase>?play ?the ADJ_NAVIG song ?please</phrase>
    <phrase>please ?play ?the ADJ_NAVIG song</phrase>

    <phrase>V_CONTROL ?(NP_SONG) ?again ?please</phrase>
    <phrase>please V_CONTROL ?(NP_SONG) ?again</phrase>

    <phrase>play ?the ?track NUMBER_ORDINAL
      ?please</phrase>
    <phrase>please play ?the ?track
      NUMBER_ORDINAL</phrase>
  </phrases>
  <semantic-interpretation>
    ...
  </semantic-interpretation>
</utterance>

<utterance name="S_RESUME_ALBUM">
  <phrases>
    <phrase>V_RESUME [the my] ?last album
      ?please</phrase>
    <phrase>please V_RESUME [the my] ?last album</phrase>

```

```

<phrase>V_RESUME ?the album ?(from yesterday)
?please</phrase>
<phrase>please V_RESUME ?the album ?(from
YESTERDAY)</phrase>

<phrase>V_CONTROL ?(NP_SONG) ?again ?please</phrase>
<phrase>please V_CONTROL ?(NP_SONG) ?again</phrase>

<phrase>play ?the ?track NUMBER_ORDINAL
?please</phrase>
<phrase>please play ?the ?track
NUMBER_ORDINAL</phrase>
</phrases>
<semantic-interpretation>
...
</semantic-interpretation>
</utterance>

<gsl>
MUSIC [music songs]
ARTIST [(three doors down) (eels) (lady gaga) (bryan
adams) (anastacia) (nickelback)]
ALBUM [(seventeen days) (daisies of the galaxy) (the
fame monster) (room service) (heavy rotation)
(dark horse)]
SONG [(right where i belong) (let me go) (grace
kelly blues) (i like birds) (bad romance)
(speechless) (east side story) (this side of
paradise) (i can feel you) (the way i see it)
(burn it to the ground) (if today was your last
day)]
GENRE [rock pop (heavy metal) folk classic ska]
COLLECTION [(my ([collection playlist playlists
LIBRARY]))]
LIBRARY [(?music library)]
V_RESUME [resume continue (pick up) (go on with)
replay]
ADJ_NAVIG [next last following succeeding preceding]
V_CONTROL [pause play stop]
YESTERDAY [yesterday (yesterday ([night evening]) )]
NP_SONG [(the song)]
NP_ALBUM [(the album)]
NP_ORD_SONG [(the NUMBER_ORDINAL song)]
</gsl>
</grammar>

```

C.4.2 news

```

<grammar lang="en" file-name="sit6_news_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>

  <utterance name="S_SHOW_PERSONAL_NEWS">
    <phrases>
      <phrase>show ?me my ADJ_PERSON news ?collage
        ?please</phrase>
      <phrase>please show ?me my ADJ_PERSON news
        ?collage</phrase>
      <phrase>[show give] me the news ?(NP_SHOWER)
        ?please</phrase>
      <phrase>please [show give] me the news
        ?(NP_SHOWER)</phrase>
      <phrase>?(can you) start the news ?broadcast
        ?please</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_SHOW_OTHER_CHANNEL">
    <phrases>
      <phrase>?(can you) show ?me NP_NEWS ?(PP_TV_CHANNEL)
        ?please</phrase>
      <phrase>please ?(can you) show ?me NP_NEWS
        ?(PP_TV_CHANNEL)</phrase>
      <phrase>show me TV_CHANNEL ?please</phrase>
      <phrase>please show me TV_CHANNEL</phrase>
      <phrase>switch to TV_CHANNEL ?please</phrase>
      <phrase>please switch to TV_CHANNEL</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_NEXT_CLIP">
    <phrases>
      <phrase>show ?me the next ?clip ?please</phrase>
      <phrase>please show ?me the next ?clip</phrase>
      <phrase>next clip ?please</phrase>
      <phrase>please next clip</phrase>
      <phrase>skip ?this ?clip ?please</phrase>
      <phrase>please skip ?this ?clip</phrase>
    </phrases>

```

```
<semantic-interpretation>
  ...
</semantic-interpretation>
</utterance>

<gsl>
  PP_TV_CHANNEL [(FROM_AT TV_CHANNEL)]
  FROM_AT [from at]
  NP_NEWS [(the ?latest news)]
  NP_SHOWER [(in the shower)]
  TV_CHANNEL[cnn bbc ard zdf (rai uno) (france 2)]
  ADJ_PERSON [personalized personal]
</gsl>
</grammar>
```

C.5 Situation 11

C.5.1 radio

```

<grammar lang="en" file-name="sit11_radio_eval.grxml">
<import location="grammar/ami_general.grm" locator="jar"/>

  <utterance name="S_RADIO_ON_OFF">
    <phrases>
      <phrase>[switch turn] [on off] NP_RADIO
        ?please</phrase>
      <phrase>please [switch turn] [on off]
        NP_RADIO</phrase>
      <phrase>radio [on off] ?please</phrase>
      <phrase>please radio [on off]</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_SWITCH_TO_ANY_CHANNEL">
    <phrases>
      <phrase>switch NP_RADIO CHANNEL ?please</phrase>
      <phrase>please switch NP_RADIO CHANNEL</phrase>
      <phrase>VP_CHANGE [(a new) another] radio CHANNEL
        ?please</phrase>
      <phrase>please VP_CHANGE [(a new) another] radio
        CHANNEL</phrase>
      <phrase>VP_CHANGE CHANNEL ?please</phrase>
      <phrase>please VP_CHANGE CHANNEL</phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_SWITCH_TO_SPEC_CHANNEL">
    <phrases>
      <phrase>VP_CHANGE RADIO_STATION ?please</phrase>
      <phrase>please VP_CHANGE RADIO_STATION </phrase>
    </phrases>
    <semantic-interpretation>
      ...
    </semantic-interpretation>
  </utterance>

  <utterance name="S_SWITCH_TO_NEWS_CHANNEL">
    <phrases>
      <phrase>VP_CHANGE a ?radio CHANNEL with news
        ?please</phrase>
    </phrases>
  </utterance>

```

```

    <phrase>please VP_CHANGE a ?radio CHANNEL with
      news</phrase>
    <phrase>VP_CHANGE a news CHANNEL ?please</phrase>
  <phrase>please VP_CHANGE a news CHANNEL</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<utterance name="S_CHANGE_MEDIUM">
  <phrases>
    <phrase>[move switch project] ?(NP_NEWS) to ?the
      shower ?please</phrase>
    <phrase>please [move switch project] ?(NP_NEWS) to
      ?the shower</phrase>
    <phrase>i [want (am going)] to shower</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<utterance name="S_CHANGE_VOLUME">
  <phrases>
    <phrase>change RADIO_VOLUME ?please</phrase>
    <phrase>please [change increase]
      RADIO_VOLUME</phrase>
    <phrase>turn [up down] RADIO_VOLUME</phrase>
    <phrase>i [cannot (do not)] hear anything</phrase>
    <phrase>[make play] [it NP_RADIO] louder</phrase>
  </phrases>
  <semantic-interpretation>
  ...
  </semantic-interpretation>
</utterance>

<gsl>
  NP_RADIO [(the radio)]
  NP_NEWS [(the news)]
  VP_CHANGE [(switch to) find (look for) (turn to)]
  RADIO_STATION [(bbc radio four) (bbc radio one) (bbc
    radio two) (bbc radio three)]
  CHANNEL [channel station]
  RADIO_VOLUME [(the ((volume of the radio) (radios
    volume) (radio volume)))]
</gsl>
</grammar>

```

D

Natural Language Understanding

```
<utterance name="S_PLAY_ALBUM_OR_ARTIST">
  <phrases>
    <phrase>[play start] MUSIC from ALBUM
      ?please</phrase>
    <phrase>please [play start] MUSIC from ALBUM</phrase>
  </phrases>
  <semantic-interpretation>
    <object type="MusicRequest">
      <slot name="playAlbum">
        <object type="playTitle">
          <slot name="playAll">true</slot>
        </object>
      <object type="hasContent">
        <slot name="title">
          <object type="hasTitle">
            <slot name="hasContent">
              <variable name="ALBUM" />
            </slot>
          </object>
        </slot>
      </object>
    </slot>
    <slot name="artist">
      <object type="name">
        <slot name="firstName"></slot>
        <slot name="lastName"></slot>
      </object>
    </slot>
  </object>
</slot>
</object>
</semantic-interpretation>
</utterance>
```

```

<utterance name="S_SWITCH_TO_SPEC_CHANNEL">
  <phrases>
    <phrase>VP_CHANGE RADIO_STATION ?please</phrase>
    <phrase>please VP_CHANGE RADIO_STATION </phrase>
  </phrases>
  <semantic-interpretation>
    <object type="changeRadioStation">
      <slot name="searchFor">
        <object type="currentStation">
          <slot name="getCurrentStation"></slot>
        </object>
      <object type="hasContent">
        <slot name="stationName">
          <variable name="RADIO_STATION"/>
        </slot>
      </object>
    </slot>
  </object>
</semantic-interpretation>
</utterance>

```

Acknowledgements

At this point I would like to thank Prof. Wolfgang Wahlster who gave me the opportunity to write this thesis on modular grammars at the German Research Center for Artificial Intelligence.

My thanks also go to Dr. Tilman Becker who provided valuable advice at all points during the thesis and had the initial idea for the topic.

Sebastian Germesin accompanied me during the development and writing process, took care of getting ODP to run, and answered all my questions. Thank you for your help and advice!

I want to thank Marc Schulder who provided helpful advice for formulations and formatting, Casey Redd Kennington who eliminated Germanisms from the text. Carolyn Ladda was there when all became too much, and provided food as well as good advice.

My thanks go to Tobias Vockerodt who was and is always there, looked after me in the last week before handing in the thesis, helped with \LaTeX questions and looked at my thesis from another point of view. Thank you for being you.

My gratitude also belongs to my family, especially my parents who supported me during my studies by all means at any time.