Semantic-based Hidden Markov/Context Free Grammar Language Modeling

Diploma Thesis

Anton Prevosti Vives

Universitat Politècnica de Catalunya

Advisors:

Dipl.-Inform. Christian Fügen
Dr. Thomas Schaaf
Dr. Tanja Schultz
Prof. Dr.rer.nat. Alex Waibel
Prof. Dr. José Bernardo Mariño

October 2006

Recognition of a truth may in some cases contain as factors both previous knowledge and also knowledge acquired simultaneously with that recognition-knowledge, this latter, of the particulars actually falling under the universal and therein already virtually known.

Aristotle, Posterior Analytics
Book 1, First Chapter
Year 350 B.C.E.

Acknowledgments

The work for this diploma thesis has been hard and it could have never been done without the help and encouragement of so many people. First of all, thanks to Prof. Alex Waibel for making my stay in Universität Karlsruhe and Carnegie Mellon University possible. Thank you, Tanja, for spending some of your time explaining the idea for this thesis and giving me the guidelines to develop it. Thank you, Christian, for your advise, your support and unlimited patience with me. Thank you, Sebastian, for introducing me to Janus during my first weeks in UKA and sorry for the advisor exchange…I hope you're now okay about that fateful decision. Thank you, Thomas, for your important encouragement and help, specially for the friendly welcome in the airport with your wife and your totally cute kids. I would like to express my gratitude to Ye-Yi Wang, from Microsoft Research, for answering my several questions so quick and clear, each time I asked for help. Thanks to the rest of the crew of the Interactive Systems Laboratories, especially to Matthias Eck, Michael Bett, Dave Housman, Kornel Laskowski, Bing Zhao, Joy, Hua Yu, Fei Huang, Yong-Hui Li, Mohamed and Muntsin for the recordings and some other support. I also want to thank Pepe (Prof. Mariño) for becoming my supervisor at Universitat Politècnica de Catalunya (UPC, Spain) and Climent Nadeu, director of the European Masters in Language and Speech at UPC, for providing me this great opportunity .

Very special thanks go to Robert, Mar, Matthias Paulik, Uli Klee, Nancy, Tim, Sven, Eleni, Maricarmen, Lupe and Jorge, Mohammad, Borislava, Quim, Edu, Dani, Francesc, Vicens, Carmen, AlbertS, Raquel, Luis, Albert, Guillem, Jaume, Eudaldo, Marc, JordiF, JaumeS, JordiP, Mn Joan Carles, Mn Josep Mª, Cati, Mn Ignasi, Pere, David, Joanda, Carles, Maria A. and Roser for their priceless friendship and prayers which have been decisive to overcome my "lows" during this time. Thank you, Enrique (in the USA) and Miquel (in Spain), for your inestimable advice. Thank you, Ramon, for so many things I can hardly enumerate. Also thanks to my brothers and sisters, my grandpa and my parents. You've been the best loving support I could ask for. And finally, thank you, Claudia, for your understanding, patience and love.

Abstract

This thesis presents a hybrid language model, both statistic- and rule-based, having the structure of a Hidden Markov Model with some nodes modeled with n-grams and others with context free rules. An EM-algorithm is used to train the model parameters from a manually annotated corpus of sentences.

The performance is evaluated from the results of the decoding with a speech recognition engine integrated with the language model and compared to a baseline 3-gram model. The designed model shows a similar word error rate than the baseline, but outperforms it in understanding accuracy by 15.3%.

Table of Contents

List of Figures

1 Introduction

Language and speech technologies have been progressing enormously over the last decades. Hundreds of applications have been successfully developed and many of them are used daily by companies, handicapped persons, education centers, individuals, public institutions, etc. More and more new scenarios can be imagined where a language- or speech-based communication with a machine or a computer application can be the best, fastest, and more natural or even the only solution. Hence the need to render computers capable of treating and processing speech and language in an efficient and natural way. Using technologies from speech signal processing to natural language processing including artificial intelligence, there is a broad spread of disciplines that already have achieved significant results in this human-machine interaction paradigm: speech synthesis, speech recognition and understanding, machine (speech-to-speech) translation, information retrieval and extraction, question answering, topic identification, document summarization and classification, dialog management, etc.

The title of this diploma thesis "Semantic-based Hidden Markov/Context Free Grammar Language Modeling" already announces that it deals with *language modeling*, which is substantial area of research within the referred disciplines.

Generally, we can say that two main approaches to language modeling have been widely used for many years, namely, statistic- and knowledge-based. Although all statistical language modeling techniques get some inspiration from a previous knowledge of language, the difference between both approaches is that the former is data-driven while the latter is rule-based.

In the last decades, the statistical methods have successfully played the main role in language modeling for speech recognition, machine translation and other applications, thanks to the increased availability of online corpora and the continuing growth of computing power and storage capacity.

There is though an increasing interest in combining both approaches. The words of the premier promoter of statistical techniques for language modeling, Fred Jelinek, "we must put language back into language modeling" [Jel95], and the paper by Ronald Rosenfeld 'Two Decades of Statistical Language Modeling: Where do we go from here?' [Ros00a], are somehow motivating remarks to continue the work of the integrating of human linguistic knowledge with pure statistical methods.

* * *

Since this thesis uses a speech recognition system to integrate and evaluate the language models developed, the next paragraph (1.1) will shortly introduce the basic concepts of speech recognition to clarify the motivation and aim of this thesis, explained later.

Chapter 2 introduces the concept of language modeling and sets out an overview of the two main trends of research in this area: statistic- and knowledge-based. The last part of the chapter also deals with some hybrid models of recent research experiments.

Chapter 3 explains the core of the thesis: how the model structure is created, what text corpus has been selected, how it is used to train the model, how the training algorithms have been developed and how the resulting language model looks like.

Chapter 4 describes the speech recognition engine that has been used to test the language models, the Janus Recognition Toolkit (JRTk), and presents some functionalities of its decoder of importance to this thesis. This chapter deals as well with the making of the test-set and the evaluation and analysis of the results in terms of Perplexity (PPL), Word Error Rate (WER) and Task Error Rate (TER).

Finally, Chapter 5 includes the conclusions of the present thesis, the discussion about possible improvements and further work that still can be explored.

1.1 Basics of Automatic Speech Recognition

An automatic speech recognizer is a system able to transcribe human speech into readable text. It can be represented with a black box with an input (speech signal) and an output (transcribed text), as shown in Figure 1.1:

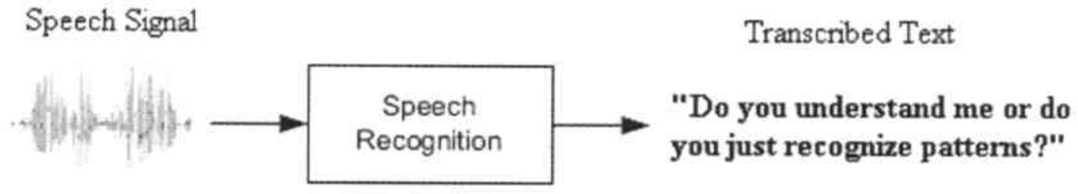


Figure 1.1: Black box diagram of a speech recognition system

Within this black box can be used any kind of process which achieves to output a text similar to the transcription that any person would make from the input speech signal. The more similar the two texts are (the one written by a human and the one resulting from the processes within the speech recognizer), the better the speech recognition system.

Optionally, an interpretation of the recognized utterance can be made, such that its meaning or intention can be determined. Strictly speaking, speech recognition only aims to transcribe text and therefore, subsequent semantic or pragmatic analysis is usually classified as language or speech understanding.

We usually distinguish between several levels in the structure of language: acoustic, phonetic/phonological, morphological/lexical, syntactic/semantic and discourse/pragmatic. Although these levels mainly work sequentially in speech production and perception (i.e., in our speaking and hearing systems), many processes use resources of various levels at the same time, i.e. they work in parallel, in order to improve their accuracy, their efficiency or even to be able to proceed.

When building a speech understanding system, a similar layer model is used. Therefore, within the black box of Figure 1.1, there is usually a chain of modules that approximately fit with the structure of language described before. These modules normally have the following names: acoustic model, pronunciation dictionary, language model and dialog manager, and relate to the structure of language as shown in Figure 1.2.

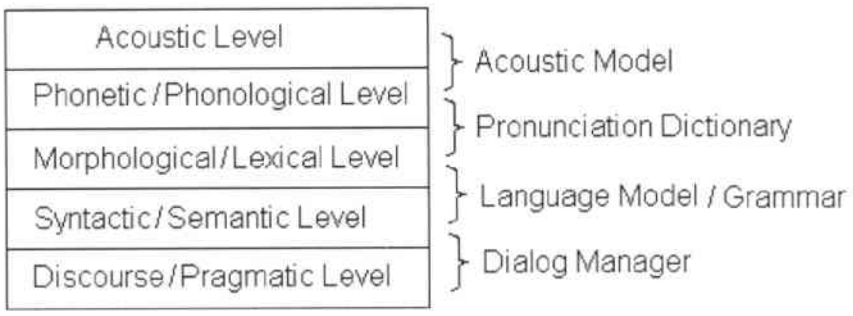


Figure 1.2: Linguistic levels and modules of a speech understanding system

Every module plays a role when recognizing/understanding an utterance. The acoustic model contains the information of the sounds or the features of the speech signal, the pronunciation dictionary establishes a link between the sounds or features and the possible words to be recognized, the language model (or grammar) deals with the feasible word sequences or sentence structures, and the dialog manager models the meanings of word sequences or sentences to yield a top level interpretation of the recognized utterance and even to keep track of the dialog or sequence of sentences spoken.

It is not yet completely known how these modules are to be created optimally and what is the best way to handle them in the recognition process. Rabiner and Juang [RJ93] distinguish three approaches:

- **the acoustic-phonetic approach**
- **the pattern recognition approach**
- **the artificial intelligence approach**

The **acoustic-phonetic approach** is based on a set of feature detectors (e.g. detectors of formants, pitch, energy, frication, etc.) followed by a feature combiner and a decision logic module that results in a lattice, which can be phoneme, syllable or even word based. This lattice, together with a module containing the vocabulary features of the words permitted, is used by a hypothesis tester who decides what sentence was spoken.

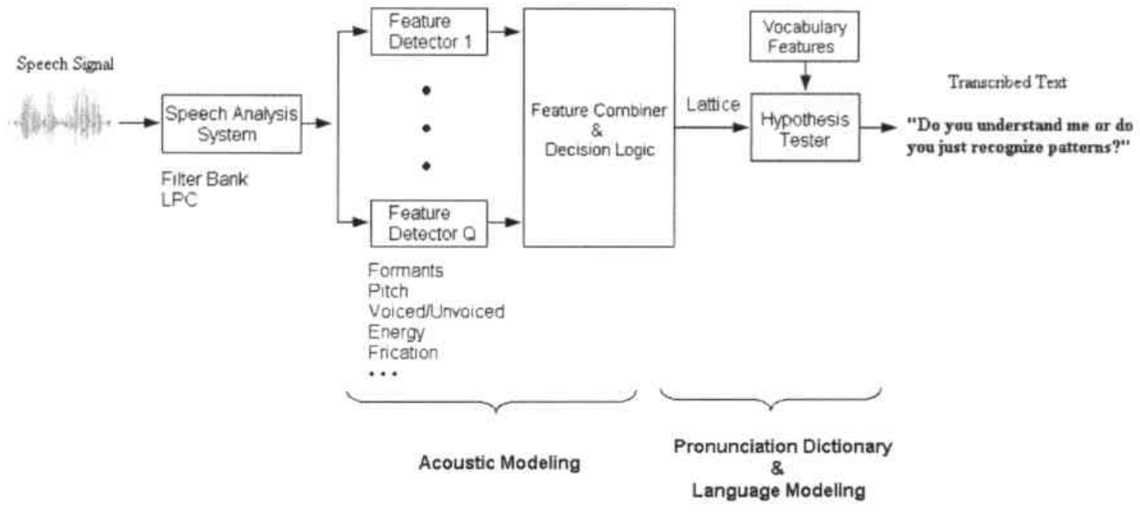


Figure 1.3: Acoustic-phonetic approach to speech recognition

Figure 1.3 shows the module diagram of this approach. The first module is the speech analysis system, a step common to all approaches, which provides to the next stages a spectral description of the speech over time with the help of filter bank, linear predictive coding (LPC), or other methods. Although the design of this module is critical for the performance of the speech recognition, it is normally not included into what is understood as acoustic modeling. The same happens with speech enhancement techniques (which would come before the speech analysis system) such as pre-emphasis filters, noise reduction methods and dereverberation algorithms, that are classified into the so-called preprocessing step.

The core of this approach relies in the feature combiner and decision logic module. It takes the feature information from the detectors and tries to find stable regions (time intervals where the features do not vary too much) and labels each segmented region with the phonetic unit that matches best with its acoustic properties. There are multiple strategies for the feature combining module to cover the sounds of a language, but all of them are based on an extensive knowledge of the acoustic features of phonetic units.

The analog approach can be applied to pronunciation dictionary creation and language modeling, e.g. with manually derived grammars, defining valid word constructions and legal word sequences. A combination of all the rules of these grammars can be integrated in between the decision logic, the hypothesis tester and the vocabulary features modules resulting in a complex recognition system, strongly dependent on the chosen strategy.

Since there is no parameter training nor any statistical inference in this approach, but a purely manually designed recognition strategy, and considering the extension of this approach to the other levels of language (not only the acoustic-phonetic), it should be properly called a knowledge-based approach. Its weakness lies on the suboptimality of the ad hoc methods, the lack of robustness and lack of adaptability to speaker, domain or environment changes and its difficult and time demanding designing process.

The **pattern recognition approach** is shown in Figure 1.4. This approach consists of two branches defining two phases:

- pattern training
- pattern classification and decision

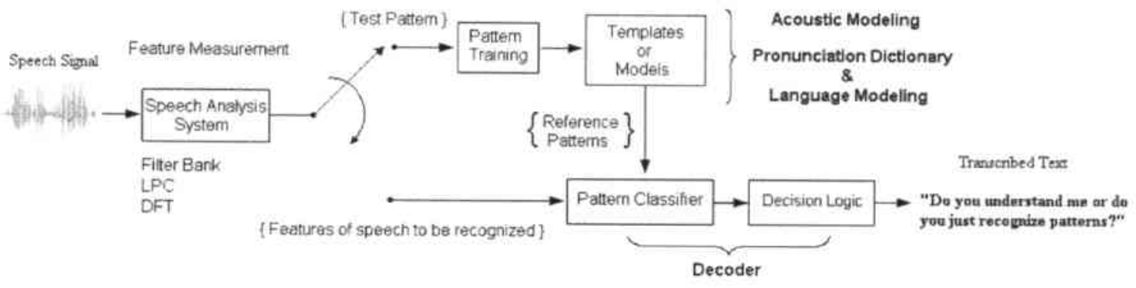


Figure 1.4: Pattern recognition approach to speech recognition

In the training step, several feature measurements of a set of training speech signals are conducted to define some test patterns. These patterns represent classes of sounds with similar characteristics. The pattern training module uses some type of averaging technique or model training for defining the reference patterns. These reference patterns can have the form of a template or what is commonly called an acoustic model.

The same approach applies as well to pronunciation dictionaries and language models, that can also be created through several statistical inference techniques.

These acoustic and language models, having been generated by statistical methods, have, in addition to some kind of model structure (e.g. a net, a tree, a chain of nodes), a set of parameters or estimated probabilities that discriminate between the several paths throughout the net, the branches throughout the tree, the transition between chain nodes or the feasible sound and word combinations. Once these parameters are calculated, the training step is completed and the recognizer is ready to work.

In the pattern classification and decision step, also called decoding step, the system seeks for the best match (according to a measure of similarity or distance) between a sequence of features of the speech to be recognized and the reference patterns contained in the acoustic model. Depending on the decoder (also called search engine), the decision of the transcription of the spoken utterance is made in conjunction with the information of the pronunciation dictionary and the language model.

Given a sequence of observed speech features X, the decoder looks for the optimal sequence of words $\hat{W}$ that corresponds to the spoken utterance. This is mathematically expressed as follows:

$$\hat{W} = \arg\max_{W} P(W \mid X) \tag{1.1}$$

where W is the sequence of words under testing. Applying the Bayes rule to the conditional probability $P(W \mid X)$ and afterwards eliminating the denominator $P(X)$, since it does not depend on W, we obtain:

$$\hat{W} = \arg\max_{W} \frac{P(X \mid W)P(W)}{P(X)} = \arg\max_{W} P(X \mid W)P(W) \tag{1.2}$$

which is called the maximum a posteriori (MAP) criterion decision rule. This rule coincides with the maximum likelihood estimate of W when the prior $P(W)$ is uniform (that is, a constant function). Thus, the main task of the decoder is to select in an efficient manner the word sequence W with the highest result of the multiplication of the modeled probability distributions $P(X \mid W)$ and $P(W)$. These estimated distributions are respectively the acoustic model and the language model computed in the training step.

Training strategies other than MAP, such as maximum mutual information estimation (MMIE) [RR95] [VOW$^+$97] or minimum classification error (MCE) [RR95] have also been successfully applied.

Because of the speaker pronunciation variability, the co-articulation effects and channel transmission characteristics, speech is a complex signal to be modeled. Nevertheless, over the last decades, Hidden Markov Models [Rab89] have shown excellent performance for statistical acoustic modeling, at the expense of making some simplifying assumptions on speech.

Statistical language models, typically n-grams, have also proven to be successful, particularly for large vocabulary recognition tasks. In Chapter 2, the different techniques for statistical language modeling will be discussed in detail.

The **artificial intelligence approach** is presented by Rabiner and Juang as a combination of the acoustic-phonetic and the pattern recognition approaches. The main idea is to incorporate linguistic knowledge from any language level (from acoustics to pragmatics) into the models and its training and use this knowledge simultaneously in the decoding process.

For example, errors in low level word or sentence hypotheses, say in the acoustic or in the lexical level, could eventually be corrected by syntactic or semantic rules that would detect the error and reject the given hypothesis or assign it a low probability.

The so-called machine learning and adaptation techniques would allow, for example, automatic acquisition of new word and sentence forming rules or new model structures, capability of real time parameter adjusting (in the decoding phase), etc. They could be applied to data-driven modeling methods as well as to rule-based methods.

For example, some of the information in the decoding process can be used as feedback to adapt the acoustic and/or language models to improve performance as new speakers, environments or tasks are introduced.

* * *

According to the nature of the utterances the system can deal with, there are four main categories of speech recognizers to take into consideration:

Isolated words: only one word at a time is recognized and stops are required between words. An acoustic model for each word is typically used. The language model can be as simple as a list of possible words.

Connected words: a pause among words is no longer required. The acoustic models are linked versions of isolated word acoustic models.

Continuous speech: subword units such as syllables, phonemes, subphonemes, polyphones, context-dependent subpolyphones, etc. are the units modeled in each acoustic model. Thus, utterance boundaries are determined by the recognizer itself. Language modeling becomes more and more important as vocabulary grows.

Spontaneous speech: the recognizer tries to model speech as spoken by persons in daily situations without a special effort to vocalize, to avoid disfluencies (typically "uhs" or "ums") or repetitions and to construct perfectly built sentences. The acoustic model, as well as the language model, has to be accordingly adapted to cover this speech modality.

There are some other classification criteria for speech recognizers, such as number of speakers, vocabulary size or complexity of the language model.

One last basic issue in speech recognition is the **evaluation**. Once the recognition system is build and trained it is desirable to estimate its performance for comparison with other methods, other systems, or even to decide if the training should continue.

To be able to evaluate the quality of the system, a similarity (or likeness) measure has to be defined, typically the recognition accuracy or its complementary, the word error rate (WER), the percentage of words not correctly recognized, or the sentence error rate (SER), the percentage of sentences not perfectly recognized.

1.2 Motivation and aim of this Diploma Thesis

When developing statistical language models, e.g. n-grams, for spontaneous speech in a specific domain, usually there is not sufficient data available to obtain accurate probability estimates, especially when the system has to be tested in several languages. Even if enough text data were available, it has been shown that n-gram models begin to saturate within a few billion of words and are not likely to improve their quality by a significant factor.

On the other hand, decoding along a knowledge-based model, e.g. a manually derived context free grammar (CFG), in restricted domains gives a large advantage over the standard n-gram approach in terms of velocity and precision, but does not cover satisfactorily characteristic spontaneous utterances, does not handle as well with unknown words and does not provide enough robustness.

As Pereira states in [Per00], a great divide has existed in research on models of spoken and written language in approximately the last forty years. Formal linguistics in the Chomsky tradition and information theory in the Shannon tradition have formed two separate (and often opposed) fronts against the problem of language modeling.

Nevertheless, some considerable advances with a unified view have been made over the last years. A mixture of both approaches reconciles concepts such as *sentence analysis* and *parsing* with *word occurrence counting*, *cognitive inferenc*e with *parameter estimation*, *ontology* and *knowledge base creation* with *text corpora collection*, *heuristic processes design* with *probabilistic techniques programming*, *machine learning* with *statistical training* and finally, *speech understanding* with *speech recognition*.

* * *

The main aim of this Diploma Thesis is to create a language model for speech recognition that takes advantage of both approaches and overcomes the problem of lack of data. Based on the latest experiments by Wang and Acero [WA01]-[WA03c], a combination of n-grams and context free grammars will be developed.

The language model created will have the form of a Hidden Markov Model (HMM) based on a previously defined semantic schema. This schema is supposed to encode the semantic constraints of the domain. A CFG Rule Library will model the emissions of some HMM states, providing domain-specific and some cross-domain constraints. The rest of HMM states will be modeled with n-grams trained with semantically annotated sentences against the semantic schema. This will provide the necessary statistical information to obtain a robust language model. The amount of required training sentences will be significantly reduced since only parts of them will be used for the n-grams. In addition, the CFG rules will be simpler than those of a grammar based recognizer because they will not be expected to cover entire sentences, but only small units.

In addition to Wang's system, this Diploma Thesis will introduce an expanded HMM/CFG structure with an extra node at the end of each task: the "Post-task node". Further, a multilingual capability of the model, in the sense of fast and easy portability to other languages, will be achieved by defining the semantic schema in an interlingua-based format. Thus, the system will be portable to other languages since the schema will not have to be changed. Only the language dependent CFG rules will be translated to the new language. The initial goal of this study was to test the system in Spanish, English and Chinese, but only the English system has been built due to time limitations.

The portability to other domains is also simplified, since it can performed by simply defining a new semantic schema and changing the domain-specific CFG rules. Herein, only the hotel-reservation domain will be implemented. The domain used in this thesis covers a considerably wider spectrum of tasks, sentence structures and vocabulary, in comparison to Wang's chosen domains (the calendar domain for MiPad [Hua01] and the ATIS "Airline Travel Information System" domain [Dah94]).

The most significant advantage of this approach for developers is that they do not have to be experts in linguistics to create the semantic schema, define the CFG rules and annotate the training sentences. The manually written rules will not be excessively complex since the n-grams and the statistical information of the HMM are supposed to supply the lack of syntactic constraints.

Another advantage of this language modeling technique is its semantic output besides of the sentence hypothesis. The task and slots associated to the recognized sentence is additional information for the next step in the recognition or translation system where the language model is used. This means that a dialog manager, a translation model, an information retrieval engine, a topic detection or any higher level module can make use of the semantic information provided by the language model.

2 Overview of Language Modeling

Language is the means of communicating using sounds or conventional symbols. Its study is made by different disciplines: linguistics, philology, sociology, anthropology, philosophy, artificial intelligence, computer science, psychology, mathematics, physical acoustics, etc. Each discipline provides a spotlight from a different viewpoint and with different characteristics of light intensity, color, beam width, etc., but all of them contribute with some light on the object that is natural language. Other human languages such as gesture languages and artificial languages, for instance programming languages, can also be modeled, but here we will focus on human spoken or written language as it is usually understood.

The study of language can also have different purposes: a *comprehension of language itself*, of its structure and operation, of its acquisition, perception and generation, of its spoken usage and its various written literary forms, of its relation with human knowledge, of its role in a social context, of its place in mental processes, etc. and a *comprehension of language in order to* set up treatments for speech therapy, enhance communication skills, improve pedagogical techniques, build electro-acoustical engines for speech recording, storage, transmission and generation or computational systems dealing with language such as word processors, speech recognizers, speech understanding systems, machine translators, speech synthesizers, etc.

In a scientific context, a *model* is an object M (an artifact, a symbolic system, a process, etc.) having some similarities with another object O (the original object being modeled), which allow to set up analogies from M to O and infer properties of O. The analogies between M and O can refer to their structure, their functions, their behavior, etc.

Language modeling in an engineering context is the attempt to create an object LM (typically a computational structure) having similarities with a given language L (human language is assumed), which allows to create a system that uses the analogies between LM and L to make it capable of dealing with L, e.g. transcribing spoken utterances of L, translating texts from L to another language L_2, correcting automatically grammatical errors in texts that are supposed to be from L, generating texts and spoken utterances of L, etc.

Since this definition of language modeling is general, a basic distinction between language modeling and acoustic modeling is usually made. Within a speech recognition framework, the difference between language modeling and acoustic modeling is not that the former models written language and the latter models spoken language. As pointed out in Chapter 1, the difference lies in that acoustic modeling focuses in the structure and behavior of *sounds* and language modeling focuses in the structure and behavior of *sentences* or *word sequences*. Therefore, there can be language models for spoken language as well as for written language. A third component between both is the pronunciation dictionary, that can also be called word pronunciation model, since it models the multiple *pronunciation forms* of each word of the language based on the sound set provided by the acoustic model.

Within this context, a key question is how a language model can be created. A first principle is that the language model LM, as defined above, does not intent to be equal to L, but just to have some similarities with L that allow to infer analogies, predictions, hypotheses, etc. from LM. Consequently, the main aim of the language model in an engineering context (in opposition to a linguistic or any other theoretical approach) is to provide processing functionalities to the system to which it belongs. Thus, consideration must be given to the goals of the language model within the system. For instance, a language model for text generation will probably not look the same as a language model for speech recognition since they have different purposes.

Another often forgotten principle, is that an enhanced comprehension of language increases the ability to improve the language model [Lee04]. There is a wide trend of thought that tends to reject theoretical studies and concentrate their efforts only on implementation issues, being satisfied with a superficial understanding of the object modeled. One might say that they stopped focusing on the object being modeled and almost only look at the model. However, serious research has a theoretical background both on the procedures and techniques of research, and on the object of research. This is similar to state that the object of the model is at least as important as the model itself. Gaining understanding of language in depth can only bring advantages.

Now rises the question of what similarities the created model, *LM*, will have with the object modeled, *L*. Based on both principles, i.e. understanding how language works and knowing what the language model has to accomplish, to provide the desired functionalities to the system, it has to be decided what part of the structure of language or which behavior of language within the syntactic/semantic level will be emulated. The similarities chosen and the way the model acquires these similarities define the language modeling technique. In addition, the feasibility of the idea and its implementation issues have to be considered.

Depending on the nature of the language model and the system, different evaluation criteria can be adopted. Language modeling techniques usually adjust the similarities of *LM* and *L* with the goal of optimizing a prescribed evaluation parameter.

As a side note, in a speech recognition system, the language model can be usually excluded. It is not necessary for recognition, even though it helps enormously in its accuracy improvement. It is, however, indispensable for speech understanding.

Next we will examine the main language modeling techniques having the above principles in mind.

2.1 Statistic-based Language Modeling

In the introduction of his famous work "A mathematical theory of communication" from 1948, Claude E. Shannon stated:

> "Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design." [Sha48]

This understanding of communication led Shannon to establish very fruitful foundations of information coding and transmission through noisy channels or what is commonly known as information theory. Its underlying principles have also been used successfully in the field of language modeling during about the last 25 years.

One of the basic concepts behind the information theory is the treatment of the message as a stochastic process with defined measurable statistical characteristics. In this fashion, what becomes important is the *repetition* of the symbols of the message or of the symbol sequence structures in a representative corpus of sentences, since statistics' sole "raw material" is the repetition of events. The basic unit of the whole theory is the *count* of each event, of each behavior and of any deep or surface structure of the information.

Now, we are able to identify what similarity between the language model *LM* and the language *L* a statistical approach uses to make predictions. It is precisely the *frequency* of words, word sequences, grammatical structures or even deeper linguistic structures. Rosenfeld defines this approach as follows:

> "Statistical Language Modeling (SLM) is the attempt to capture *regularities* of natural language for the purpose of improving the performance

of various natural language applications. By and large, statistical language modeling amounts to estimating the probability distribution of various linguistic units, such as words, sentences, and whole documents." [Ros00a]

However, any statistical approach to language modeling requires some previous knowledge of language, since it must know the kind of linguistic unit or structure whose regularity is trying to capture and have at least a basic idea of how this unit or structure behaves. Therefore, this approach can be also basically defined as data-driven, since it departs from some elemental knowledge of the object modeled to capture the distribution of the prior known units or of the prior known unit properties.

The statement that such an approach can even *learn* new things in addition to the probability distribution of the underlying linguistic knowledge of the method is certainly something that has to be meticulously analyzed. This subject, nevertheless, cannot be treated herein, as it would lead to an extensive study.

Statistical language modeling techniques have mainly focused on estimating the probabilities of *surface* units of language, typically words and word sequences. Only a few techniques have been successful in trying to capture hidden structure regularities of language in the syntactic or semantic level and then use them to improve word accuracy. This is why the main goal of practically all statistical language modeling techniques is to estimate the so-called *prior probability distribution* over all possible sentences, spoken utterances or some other word-based linguistic units:

$$\hat{P}(W) \qquad \forall W = w_1 w_2 ... w_m \tag{2.1}$$

The term w_i should be understood in the broad sense of "word": besides the entrances of a dictionary, other spoken utterance units like "uhs" or "ums" or some vocal noises might also be included for spoken language modeling. Other symbols in written text, for instance, orthographic punctuation marks "! ? . , ; ( )", abbreviations "i.e. e.g. & $" or even smilies "☺ :D ;)" can be considered "words" to be modeled, even though they might have to be treated differently in some cases.

Almost all main statistical techniques decompose the probability of W into a product of conditional probabilities:

$$P(W) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2) \cdots P(w_m \mid w_1 w_2 \ldots w_{m-1}) = \prod_{i=1}^{m} P(w_i \mid h_i) \tag{2.2}$$

$$\text{where } h_i \overset{def}{=} \{w_1 w_2 \ldots w_{i-1}\} \text{ is called the \textit{history} of } w_i$$

and differ from each other in the way they estimate $P(w_i \mid h_i)$.

An extension of this approach is to consider the estimation of a joint probability of word sequences and syntactic structure (e.g. parts of speech or syntactic phrases), $P(W, Syn)$, or word sequences and semantics (e.g. tasks or semantic classes), $P(W, Sem)$ or even the three of them altogether, $P(W, Syn, Sem)$, in order to use word, syntactic and semantic information simultaneously in the recognition process.

The evaluation of a statistic-based language model is commonly done by computing the probability that the model assigns to test data, or the derivative measures of *cross-entropy* and *perplexity*. For a given set of test sentences $T = \{W_1, W_2, \ldots, W_N\}$, the probability of the test set is the product of the probabilities of all sentences in the set:

$$P(T) = \prod_{i=1}^{N} P(W_i) \tag{2.3}$$

The *cross-entropy* $H_p(T)$ of the model is defined as:

$$H_p(T) = -\frac{1}{W_T} \log_2 P(T) \tag{2.4}$$

where W_T is the length of the text T measured in words.

The *perplexity* $PP_p(T)$ of the model is defined as:

$$PP_p(T) = 2^{H_p(T)} \tag{2.5}$$

Perplexity is the average branching factor of the language according to the model. The difficulty of recognizing a language with perplexity PP is comparable to the recognition of another language with a uniform distributed vocabulary of size PP. Lower cross-entropies and perplexities are better, indicating that the language is more "predictable" by the model.

Next we will look into the main and most successful technique, called *n-grams*, and its major problems, namely, *smoothing* and *vocabulary clustering*. Then we will briefly introduce several other methods with a main statistical basis.

2.1.1 N-grams

The n-gram attempt to estimate $P(w_i \mid h_i)$ starts by treating language as a Markov source of order $n-1$, using the approximation that the conditional probability of a word depends only on the past $n-1$ words:

$$P(w_i \mid h_i) \approx P(w_i \mid w_{i-n+1} \ldots w_{i-1}) \tag{2.6}$$

where the value of n balances the *variance* and the *bias* of the estimate. Common values of n are 2 (bigram), 3 (trigram) or 4. Large corpora usually demand trigrams or 4-grams. The estimation of the probability is done with large amounts of texts. The quality of the corpus is determinant to get a representative probability distribution of the domain. If the coverage is good, the more quantity of text is available for training the more the accuracy of the model.

A straightforward estimation can be done by simply counting in the corpus how many times each word sequence of length $1, 2, ..., n$ appears in the text and normalizing it:

$$\hat{P}(w_i \mid w_{i-n+1}...w_{i-1}) = \frac{c(w_{i-n+1}...w_i)}{c(w_{i-n+1}...w_{i-1})} \tag{2.7}$$

maximum likelihood (ML) estimate for n-grams

A plain n-gram language model relies exclusively on the frequency of apparition of each word sequence in the corpus and therefore has theoretical and practical limitations.

Regardless of how big the training corpus is, there will always be n-grams that seldom appear or even appear just once. These n-grams get a probability very close to zero, and most of them have no linguistic reason to be considered more improbable than others. Several *smoothing techniques* and *vocabulary clustering methods* have been developed to overcome this problem of lack of density of data for some word combinations, called *sparseness*. As Rosenfeld states:

> "Ironically, the most successful SLM techniques use very little knowledge of what language is. The most popular language models (n-grams) take no advantage of the fact that what is being modeled is language – it may as well be a sequence of arbitrary symbols, with no deep structure, intention or thought behind them.
>
> [...] But one can only go so far without knowledge."

To outperform n-grams, we ought to analyze their weaknesses. A clear limitation is that they do not capture long distance dependencies of words nor deep relationships between linguistic units in a sentence. This often results in ungrammatical and nonsensical errors. N-gram language models also suffer from data fragmentation and lack flexibility to add new vocabulary and to adapt to new domains or genres.

2.1.2 Smoothing and count cut-offs

Smoothing techniques basically reserve some of the probability mass of the ML estimation for distribution among infrequent or unseen n-grams. They are also often called discounting techniques, because the re-distribution of mass probability can be seen as a re-distribution of counts. As explained in [CG96], almost all techniques can be classified into *back-off* or *interpolated* models.

Back-off models can be described with the following equation:

$$P_{smooth}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i \mid w_{i-n+1}^{i-1}) & if \;\; c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) P_{smooth}(w_i \mid w_{i-n+2}^{i-1}) & if \;\; c(w_{i-n+1}^{i}) = 0 \end{cases}$$

$$where \;\; w_p^q \overset{def}{=} w_p \ldots w_q \tag{2.8}$$

Back-off smoothing probability distribution

where $\alpha(w_i \mid w_{i-n+1}^{i-1})$ is the distribution used for n-grams with nonzero counts and $P_{smooth}(w_i \mid w_{i-n+2}^{i-1})$ is the lower-order distribution we *back-off* to with unseen n-grams. And $\gamma(w_{i-n+1}^{i-1})$ is a scaling factor to making the sum of $P_{smooth}(w_i \mid w_{i-n+1}^{i-1})$ equal to unity. Katz smoothing and Kneser-Ney smoothing are some of the most popular back-off n-gram models.

In addition to *back-off* smoothing techniques, *interpolated* models use the lower-order distribution $P_{smooth}(w_i \mid w_{i-n+2}^{i-1})$ for n-grams with nonzero counts. Thus, their conditional probability distribution can be expressed as follows:

$$P_{smooth}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \alpha'(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})P_{smooth}(w_i|w_{i-n+2}^{i-1}) & if \;\; c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1})P_{smooth}(w_i|w_{i-n+2}^{i-1}) & if \;\; c(w_{i-n+1}^{i}) = 0 \end{cases}$$

$$where \;\; w_p^q \overset{def}{=} w_p \dots w_q \tag{2.9}$$

Interpolated smoothing probability distribution

The Jelinek-Mercer smoothing and the Witten-Bell smoothing are among the most used interpolated n-gram language models.

Count cut-offs is a simple technique to prune language models in order to reduce their size. It deletes those n-gram counts below a certain cut-off threshold. This threshold has to be carefully chosen to prevent discarding significant n-grams, which would lead to a perplexity increase.

2.1.3 Vocabulary clustering

The sparseness problem of data can also be battled via vocabulary clustering. This technique involves building classes of words or phrases and estimating the n-gram probabilities replacing each word belonging to a class with a class ID. Named entities are very suitable for being clustered into classes, but not always a manual classification results in perplexity improvements. Automatic clustering methods, e.g. k-means, have shown good results for large corpora and wide domains.

2.1.4 Other techniques

Even if n-grams are still in the top of statistical language modeling, some other data-driven methods have been developed and tested in recent years: *decision trees*, *exponential models* (also called Maximum Entropy models) and *adaptive models* (cache-based). They try to overcome common problems of n-grams, such as data fragmentation, inflexible dependency of each word with its $n-1$ preceding words, equal treatment to all kinds of words and poor generalization capability.

2.2 Knowledge-based Language Modeling

"Evidently, one's ability to produce and recognize grammatical utterances is not based on notions of statistical approximation and the like. The custom of calling grammatical sentences those that 'can occur', or those that are 'possible', has been responsible for some confusion here. It is natural to understand 'possible' as meaning 'highly probable' and to assume that the linguist's sharp distinction between grammatical and ungrammatical is motivated by a feeling that since the 'reality' of language is too complex to be described completely, he must content himself with a schematized version replacing 'zero probability, and all extremely low probabilities, by *impossible*, and all higher probabilities by possible'. We see, however, that this idea is quite incorrect, and that a structural analysis cannot be understood as a schematic summary developed by sharpening the blurred edges in the full statistical picture."[Cho57]

The above statements from *Syntactic Structures* (1957) by Noam Chomsky along with other principles formulated in his revolutionary work, motivated and set the theoretical base of fruitful research in linguistics and computer science. His contribution to the theory of formal languages and to the creation of generative grammars is unquestionable. He has been criticized because of his apparent rejection of statistical approaches to language. But what he really firmly pointed out was the insufficient descriptive power of a statistic-based model because of its nature and the lack of data, usually called the parse data problem.

Chomsky was the first to introduce systematically the logic formalism into linguistics. On of his contributions is his understanding of language as an infinite set of sentences generated by a grammar. He defined (for the first time) grammar as a formal system characterized by a set of units and a set of well-formed rules. The use of the notion of recursivity (originally from mathematics) in language analysis and his hierarchy of generative grammars also have influenced many linguists and computer scientists.

In this section we introduce the concept of knowledge-based language modeling. Basically, the goal is to create a grammar that optimally accomplishes the required function within the engineering system it is part of. Usually a grammar constitutes the framework to *parse* an input sentence. Parsing is commonly understood as an alignment of the sentence with a grammatical structure, e.g. a parse tree. Another function of grammars can be to *transfer* a representation of a sentence into another representation, e.g. to translate the sentence into another language. There are also grammars for *generation* of language and grammars for semantic *interpretation* of an utterance.

Here we will only examine what kinds of formal generative grammars exist and why semantic grammars seem to be very suitable for speech recognition.

2.2.1 Phrase Structure Grammars

A *language*, in a general sense (artificial or natural), is a set of *sentences*; each *sentence* is a chain of one or more symbols (words) belonging to the vocabulary of the language. A *grammar* is a formal and finite specification of this set and can have different forms. It usually consists of a formalism based in a set of rules. This method of specification is called a generative grammar or a phrase structure grammar. It has four components:

V_n : non-terminal vocabulary

V_t : terminal vocabulary (or words of the language).
The union of both vocabularies is defined as $V = (V_n \cup V_t)$

P : set of rules, also called productions with the form $\alpha \to \beta$
where α is a sequence of one or more symbols of V_n ($\alpha \in V^+$)
and β is a sequence of zero or more symbols of V_t ($\beta \in V^*$)

S : start symbol from V_n

The basic operation of a phrase structure grammar is to *rewrite* a sequence of symbols into another sequence according to the set of rules *P*. Any sentence of the language can be generated by starting with the initial symbol *S* and doing rewriting operations sequentially until a rewriting rule produces only terminal symbols.

Grammars have different generative power. The most powerful correspond to the so-called *recursively enumerable languages*. They are basically infinite subsets of sentences that can be enumerated even though it cannot always be determined if a given sentence belongs to the language or not. They are generated by the so-called *unrestricted grammars*, which include all other less powerful grammars classified into the following Chomsky hierarchy table:

Type	***Grammars***	***Rule restrictions***	***Automaton***
0	Unrestricted or Free	None: $\alpha \to \beta$	Turing Machine
1	Context sensitive	The RHS includes at least the symbols of the LHS: $A \to \beta / X\_Y$ Or alternatively: $xAz \to x\beta z$	Linear-Bounded Automaton
2	Context Free	In the LHS can only be one symbol: $A \to \beta$	Push-Down Automaton
3	Regular or Finite State	The rules can only have this two forms: $A \to t\ B$ $A \to t$	Finite State Automaton

where $Type\ 3 \subset Type\ 2 \subset Type\ 1 \subset Type\ 0$. This means that each grammar type is a subset of any higher type grammar (Type 0 is the highest one). The right column of the table indicates the corresponding *automaton* of each grammar. The column in the center specifies the restrictions to the rules. The capital letters *A* and *B* represent a single non-terminal symbol. The expression *X_Y* represents a specific context where *A* appears. The letter *t* denotes a single terminal symbol.

The less powerful grammars adopt several names, depending on what field they are used: *regular grammars* in formal grammar theory, *finite state automata* in automata theory, *transition networks* in symbolic computational linguistics, *Markov chains* in information theory and statistics. It is important to note that an n-gram model is simply a Markov chain (of order $n + 1$) with probabilities associated to each state and state transition.

Context free grammars (CFG) are among the most used. Many programming languages are context free and can be parsed with a Push-Down automaton. Any CFG can be equivalently converted into special *normal forms*, typically the Chomsky Normal Form, which basically try to optimize the formalism for particular types of processing. The question if natural language is context free has been a controversial issue for many years. More recently, some examples of natural languages (Dutch, Swiss German and others) have proven to be context sensitive. Nevertheless, CFGs are powerful enough to cover the main phenomena of natural language and they are sufficiently simple to allow *efficient* search through them.

Context sensitive grammars and unrestricted grammars require complex algorithms to be treated and have not been particularly suitable to establish the most common grammatical restrictions. Consequently, other grammars have been developed trying to capture the natural language behaviors that pure CFGs do not cover.

Many other grammars exist, such as link grammars, transformational grammars, unification grammars, feature grammars, augmented context free grammars, dependency grammars, systemic grammars, tree adjoining grammars, etc. but their treatment is beyond the scope of this thesis.

A grammar is normally not operative by itself. An algorithm that analyses the input sequences in order to determine its grammatical structure with respect to the given grammar is necessary to make the grammar functional. This computer program is called *parser*. Many parsing strategies exist responding to different needs. Common parsers are LL, LR, CKY, Earley, etc.

A decoder able to read the grammar rules and use them to formulate the recognition hypothesis together with the acoustic model is required for using a grammar as a language model, e.g. in a speech recognizer. Such decoders already exist, though limited to specific grammar types (usually regular or context free grammars) and still not very efficient and fast.

2.2.2 Semantic Grammars

Moreno Sandoval [Mor98] declares the autonomy of syntax from semantics within formal symbolic systems. He argues that a formal system consists of a syntax (defining well-formed rules) and a semantics (which interprets the meaning of the well-formed expressions) and that the inference rules of the syntax can be built without bearing the semantics in mind. This is generally true; however, the relationship between them has not yet been advantageously used in language modeling techniques.

A syntactic grammar describes the relationships between syntactic categories (nominal phrase, adverb, adjective, etc.) and how they are used to generate a sentence. A semantic grammar normally uses phrase structure rules as well, but the constituents are classified in terms of their function or meaning (e.g. time, direction, affirmation, etc.) rather than in terms of syntactic categories.

Nevertheless, if only syntactic rules are taken into account, ignoring the semantics of an utterance, many sentences hypothesis would happen to be well-formed, but meaningless. Chomsky's example from [Cho57] is very revealing:

(1) *colorless green ideas sleep furiously*

(2) *furiously sleep ideas green colorless*

Sentence (1) would be probably considered well-formed by any syntactic parser based on POS tags, but is obviously nonsensical. Sentence (2) is not well-formed and also nonsensical. Let us consider these two further sentences:

(3) *i would like a room with a shower*

(4) *i would like a shower with a room*

Both (3) and (4) are syntactically well-formed, but (4) is not semantically clear. It seems to be no syntactic knowledge that can determine that (1) and (4) are not acceptable. A trigram model would probably assign a very low probability to (1), because all sequences of 3 words are very rare, if not unseen. Therefore, the semantic problem with (1) could be solved with a trigram model (or even with a bigram model). But all trigram probabilities of (3) will probably have similar values as those of (4) and the model will have more difficulties in deciding that (4) is not acceptable.

Now, it becomes clear what a semantic grammar can contribute to recognition tasks, besides of the semantic information that it may output to a dialog manager or to any higher level module after it.

Another remarkable advantage of semantic grammars is that they are known to be more robust against spontaneous speech effects [FMS04].

2.3 Hybrid Models and related work to this thesis

As early as 1989, Fred Jelinek stated:

> "The purpose of a language model for speech recognition is not an exact analysis for meaning extraction, but an apportionment of probability among alternative futures. This provides an opportunity for creative use of appropriate **grammatical principles** that are on the whole accurate even if open to counter-examples."

"The difficult-to-realize aim of language modeling should be to set up a general model structure **capable of self-organizing itself** to an efficient (best is too much to hope for) solution on the basis of a training corpus."[Jel89]

More than 15 years later, it seems that this aim of language modeling, a self-organizing structure that appropriately uses grammatical principles open to counter-examples, still has to overcome the over-simplistic most successful pure statistical language model: n-grams.

Smoothing techniques and some vocabulary clustering methods have substantially improved the accuracy of n-grams, but human knowledge of language has still not been successfully articulated in a suitable structure to achieve the goals of large vocabulary language modeling. Only grammars for very restricted domains have shown good results.

In this section, we describe different attempts to combine linguistic knowledge with statistical corpus knowledge made in the last years. Since this area is still relatively unexplored, many of these attempts were unsuccessful, although some have shown promising results. We sort these techniques depending on their balance between the data-driven and rule-based character.

2.3.1 Putting language into statistics

As Rosenfeld reports in [Ros00b], there are several levels of language that can be exploited getting linguistic information from them and using it in statistical language modeling methods.

Lexical relations between words can serve to group similar words (in a lexical, not phonetic sense) into so-called clusters. Each cluster is assigned a tag, e.g. DAY_OF_THE_WEEK and an n-gram LM is trained substituting each cluster tag for the words belonging to it. The clusters or classes can be semantic-based, Part-Of-

Speech based or automatically generated through statistical methods, e.g. K-means. The idea behind these techniques is to reduce the dimension of the estimation problem and hence its variance. However, they only have shown improvements over the baseline n-gram LM in very specific domains.

At a *syntactical* level, several efforts have tried to integrate linguistic knowledge with statistical language modeling: inducing probabilistic CFGs out of an annotated and parsed corpus, training a probabilistic link grammar or a dependency grammar, collecting linguistic classifications of the history of words and creating a predictive model out of it. Lately, an attempt to incorporate CFG rules into n-grams has been made [Fle04], but with little success. Generally speaking, experiments in these directions yielded small improvements, but more and more new methodologies emerge with promising approaches to language modeling.

The *semantics* is a especially interesting level of language to exploit, since many applications that use a language model can also use a semantic or intentional interpretation of the utterances they deal with. However, not all semantic approaches to LM provide an interpretation as an additional output. For example, interpolation of topic-based n-gram LMs, capture topic coherence and word correlations by using an n-gram cache, reduction of dimensionality of the topic space using Singular Value Decomposition (SVD) in combination with n-grams, etc. References to the experiments can be found in the quoted paper by Rosenfeld.

2.3.2 Putting probabilities into language

The most common statistical approaches to language modeling estimate the prior probability distribution:

$$\hat{P}(W) \qquad \forall W = w_1 w_2 \ldots w_m \tag{2.10}$$

and try to reduce the perplexity, which solely depends on these probabilities and the length of each sentence.

This approach results in an optimization of word accuracy in a speech recognizer that can be of interest if the main objective is to exactly transcribe the spoken words. Often systems seek a posterior semantic interpretation of the utterances in order to act consequently to the speakers request, answer, information given, order, etc. Thus, many computational systems dealing with language have as ultimate goal language understanding rather than word accuracy.

To this end, a syntactically or semantically structured language model should probably be optimized on the basis of a **joint probability distribution** of the word sequences W and the syntactic structure behind the model *Syn*, or the semantic underlying interpretation *Sem* respectively.

$$\hat{P}(\ W,\ Syn,\ Sem\ ) \tag{2.11}$$

This approach would start from a previously defined syntactic, semantic or syntactic/semantic structure and would make this structure self-organizable (following Jelinek's expression) and open to counter examples, i.e. making it trainable from an annotated corpus. The self-organization and training can be made in two forms:

- departing from one or a couple of simple small structures or rules and **expanding** the model and making it more and more complex as far as the training data requires.

- departing from a **fixed structure** (e.g. a CFG in the form of a finite state automaton) and aligning it with the annotated corpus, in a way that probabilities can be assigned to the different parts of the model structure.

Some similar experiments conducted in recent years can be classified into this example-based grammar learning techniques: an *unsupervised* grammar induction technique of HMMs presented in [SO04], a semi-supervised grammar learning technique reported in [WM01], a supervised acquisition method for a statistical understanding model detailed in [MBIS04], an interesting "learning-by-doing" example for grammar development in [Gav00b], a very recent work explained in [Jon06] creating n-gram models from interpretation grammar generated corpora. Many other attempts are still under construction trying to benefit from the available knowledge of language and from the newest estimation and machine learning methods.

2.4 Multilingual Models

In numerous new scenarios persons from different countries, with different languages, have to communicate, triggering increasing interest in multilingual applications such as multilingual speech recognition or even multilingual speech-to-speech translation.

The term *multilinguality* indicates a **single** system working simultaneously with more than one language rather than several multilingual systems one besides another. These systems, might have a multiligual front-end, as reported in [WTSW02], a multilingual translation model, e.g. the interlingua-based machine translation system in [SBVW06], or a multilingual language model as detailed in [FSS+03].

2.5 Multidomain Models

Multidomain models are similar to multilingual models in the sense that they can deal with various domains at the same time within a single system. The difference is that while multilingual systems make both the acoustic and the language models capable of treating multiple languages, multidomain systems only consist of a language model capable of dealing with several languages. The reason is that a change of domain has usually no considerable impact on the phonetics, but it certainly has on the vocabulary, the sentence structures and semantics.

[HHP01] reports two methods for a multilingual model, one with a parallel search mechanism and the other with a combination of two domains in a single joint network. However, the multidomain capability introduced into the system results in a small accuracy degradation.

2.6 Portability

Language independent systems (also called interlingua) or easy portable systems to other languages are very desirable, because of the potential savings of cost and time. For domain specific systems, a simple portability to other domains is highly desirable as well.

The following issues must be considered to make a language model easily portable to other domains/languages:

- the training of the model in another domain/language should require a reasonably small amount of training data, since it is often very difficult to obtain corpora for specific domains, specially in different languages other than English.

- if required, a simple annotation of data is desirable. If possible, an automatic annotation or parsing is obviously the most "portable" solution.

- if domain/language specific rules are required, they should be as simple as possible, precluding the need for assistance by a linguist expert.

3 Semantic-based Hidden Markov/Context Free Grammar Language Modeling

Ye-Yi Wang, in his paper "Is Word Error Rate a good indicator for Spoken Language Understanding Accuracy?", poses an important question about the language modeling paradigm for speech recognition as follows:

> "Most (if not all) of the approaches treat understanding as a separate problem, independent of the speech recognition. A two pass approach is often adopted, in which a domain-specific n-gram language model is constructed and used for speech recognition in the first pass, and the understanding model obtained with various learning algorithms is applied in the second pass to 'understand' the output from the speech recognizer." [WA03c]

But this two pass approach is *suboptimal* if we are looking for a reduction of the overall understanding error rate and it yields *poor accuracy* when training the n-grams for a specific application, since it is very difficult to obtain large amounts of training data for some domains. Adressing these two problems, Wang states:

> "More important than word error rate reduction, the language model for recognition should be trained to match the optimization objective for understanding."
>
> "It is thus desirable to include prior knowledge (e.g., domain knowledge and grammar models for domain-independent concepts) in a language model whenever this is possible [...] to compensate for the lack of language modeling training data." [WA03c]

It could be added that both speakers and listeners have the experience of not focusing on how to built a sentence, but on the content of what is to be communicated. Indeed, in the process of acquisition of language a child learns to distinguish between different utterances motivated by their meaning and not by their syntactical or superficial structure. The learning process of well-constructed sentences is therefore understanding-based in the first stages of a child's life. Afterwards, usually in school, children are taught grammar rules directly. A person with a mature competence of language has usually internalized grammar rules and speaks and understands seeing language as a whole, always seeking for the meaning of what he/she says or what is said. Recent philosophical investigations [Can87] show that "understanding something" is equivalent to "saying in mental words" in a kind of mental language, that can be transformed into oral or written language.

This **unified view of language** as a *means* and *act of understanding* (or equivalently, an *act of expressing* something meaningful, previously understood) is what this thesis seeks in a theoretical level. A **unified building process** of the language model, i.e. statistic- and rule-based has been chosen to take advantage of both approaches. Limitations of time and resources have only permitted an initial attempt to implement a system based on these principles.

This chapter describes the system built for this thesis. It is based on the last papers of Ye-Yi Wang on SGStudio, an example-based grammar learning/development tool designed basically for speech recognition purposes. The present work tries to generalize and expand Wang's system from various viewpoints.

Basically two tools have been programmed: *SemanticTools* and *GrammarDeveloper* which can be integrated within a sole development framework. *SemanticTools* helps the developer to create the semantic schema that will be the basic structure of the language model. *GrammarDeveloper* takes the semantic schema as input together with the training data in order to train the model probabilities and outputs a finished language model in a hybrid grammar/n-gram format.

3.1 Defining the semantics

The title of this thesis announces that the language model proposed here is semantic-based, i.e., it has an underlying structure that tries to cover the possible semantic contents of any sentence within a previously chosen domain. We call this the *semantic schema*. It can be seen as a simplified version of the so-called *frames* from the artificial intelligence field, introduced by Minsky in [Min75].

The *hotel-reservation domain* has been chosen for this study. It is a subdomain of the Basic Travel Expression Corpus (BTEC) [TSK02], a known text corpus with spoken dialogues of basic tourism and travel situations. Typical of this domain are sentences asking for available rooms, indicating directions, greeting, ordering hotel services, requesting price information, etc.

The tags used in the definition of the semantic schema are based on the Interlingua format of the NESPOLE! Project [LGW$^+$03], also called Interchange Format. As explained in the referred paper, the semantic tags, which define *domain actions*, are composed of two parts: the *speech act* and its corresponding *concepts*.

For example, a *domain action* called "request_information-disposition-room", has these two parts: the *speech act* "request_information", also called *speaker intention* or *task*, and the *concepts* "disposition" and "room".

As the schema is defined in XML format, the tasks are *top-level tags* and the concepts are nested within them in *slots*, as shown in the following box:

```
<top-level tag>
        <slot #1/>
        <slot #2/>
        ...
</top-level tag>
```

The decision of the tasks and slots that our semantic schema ought to have is critical. Our idea is to define a generic semantic framework rather than a very detailed and exhaustively specified grammar. This flexible schema will leave to the statistical part of the model more room for data-driven constraints. The choice of the tasks and slots has been based on some initial semantic context free grammars from the Interactive Systems Laboratories at the Universität Karlsruhe (TH) that cover the whole hotel-reservation domain. These grammars are in the SOUP format by Marsal Gavaldà [Gav00a].

The following box shows two examples of top-level rules from these grammars[1]:

```
s[request_information-disposition-room]
    (*+RHETORICALS [super_disposition=] [super_room-spec=])

s[thank-action-person]
    (*+FILLER-FOCAL *ACKNOWLEDGE gracias por [super_action=] [super_person-spec=] *+FILLER-FOCAL)
```

Now, if we look, for example, at the part of the grammar with the rules for "[super_room-spec=]":

[1] '+' preceding a token indicates repeatability
'*' preceding a token indicates optionality
'*+' preceding a token indicates optionality and repeatability

```
[super_room-spec=]
        (*+FILLER *[operator=range] [room-spec=] *+C_ROOM-SPEC)
        (*+FILLER DISJUNCT-PHRASE *con [room-spec=] C_ROOM-SPEC)

C_ROOM-SPEC
        (*+FILLER *[operator=] *+FILLER [room-spec=] *+FILLER)
        (*+FILLER *[operator=] *+FILLER [pre-mod:room-spec=] *+FILLER)

[room-spec=]
        (*ROOMS-PREP *+ROOM-MODS-PRE ROOMS *+ROOM-MODS-POST)

ROOM-MODS-POST
        ([super_object-number=])
        (*+FILLER-FOCAL [super_contain=] *+FILLER-FOCAL)
        (*+FILLER-FOCAL [mod:super_exclude=] *+FILLER-FOCAL)
        (*+FILLER-FOCAL [super_excluded-from=] *+FILLER-FOCAL)

ROOM-MODS-PRE
        (*+FILLER-FOCAL ORDER-REF *+FILLER-FOCAL)
        (*+FILLER-FOCAL *OBJ-MODS-PRE-PRE [super_modifier=] *+FILLER-FOCAL)
        (*+FILLER-FOCAL *OBJ-MODS-PRE-PRE [super_portion=] *+FILLER-FOCAL)
        . . .

ROOMS
        ([bedroom])
        ([double_room])
        ([family_room])
        ([suite])
        ([room])
        . . .
```

we see how complex and detailed these grammars are, even being semantic-based. They are analysis grammars thought for pure rule-based parsing. For this reason, they have to go above the limits of semantics and include implicitly many syntactic rules. In addition, these rules have filler tags that cover spontaneous speech effects and make the grammar more flexible.

The model we want to built has to leave some degrees of freedom to be used by its data-driven approach. On the other hand, we desire a language independent semantic schema that can be easily ported into any language. The SOUP grammars are obviously extremely language dependent.

Therefore, we have simplified the given grammars keeping the domain sentences of the BTEC corpus in mind. The *top-level Left Hand Sides* (candidates to be a root of a parse tree) have been directly taken as the *tasks* of the schema with little modifications. The *Right Hand Sides* of the top-level rules have been analyzed more carefully. Fillers, rhetoricals, post- and pre-arguments and, in general, optional terms of the rules (those preceded by an asterisk) have been left out. Only terms that are not optional and that represent a semantic concept have been taken as the *slots* of the semantic schema. Accordingly, the semantic schema corresponding to the two SOUP top-level rules shown before looks as follows:

```
<Task name="request_information-disposition-room">
      <Slot name="super_disposition"/>
      <Slot name="super_room-spec"/>
</Task>
<Task name="thank-action-person">
      <Slot name="super_action"/>
      <Slot name="super_person-spec"/>
</Task>
...
```

As we will see in the next section (3.1.1.), the slots have to be additionally simplified by taking lower terms from the parse tree. For example, instead of "super_room-spec", the auxiliary non-terminal "ROOMS" has been taken (auxiliary non-terminals are arguments that almost are in the last level above the terminal words). Additionally, words are thus left out letting the data-driven method cover them. The lower the terms taken of the grammar are, the fewer the words corresponding to a slot.

This large simplification of the grammar rules is also done with another purpose. The development of complex grammars as the above referred is very time-consuming and requires qualified persons with deep linguistic knowledge and parsing experience. Our goal is to built a system that can be easily ported by non experts into other domains and languages.

The final semantic schema defined in this thesis for the hotel-reservation domain consists of 26 tasks and 14 different slots, listed in the following table:

Task name	Slots
affirm	no slots
apologize	no slots
exclamation	no slots
give_information-arrival	TEMPORAL
give_information-direction	DIRECTION, LOCATION
give_information-existence-object	OBJECT, LOCATION
give_information-object-property	OBJECT, OBJECT-PROPERTY
give_information-personal-data	PERSONAL-DATA, PERSONAL-OBJECT
give_information-price	CURRENCY, PRICE
give_information-time	TEMPORAL
give_information-wish	OBJECT, ACTION
greeting	no slots
how_to	ACTION
negate	no slots
request_information-accomodation	ACCOMMODATIONS
request_information-direction	LOCATION, ACTION, OBJECT
request_information-existence-object	LOCATION, OBJECT
request_information-feasibility-action	ACTION
request_information-object-property	OBJECT, OBJECT-PROPERTY
request_information-personal-data	PERSONAL-DATA
request_information-price	PRICE
request_information-room	ROOMS, ROOM-PROPERTY
request_information-time	ACTION, EVENT, TEMPORAL
request_information-wish	ACTION, OBJECT
suggest-action	ACTION, OBJECT, TEMPORAL
thank	no slots

3.1.1 Context Free Grammar Library

A set of rules is needed in addition to the semantic schema to define the several word sequences that each slot (LOCATION, TEMPORAL, ACTION, etc.) represents. These rules, as already seen for the slot "ROOMS" (in the box with the rules for "[super_room-spec=]"), can be taken directly from the SOUP grammars, though they can be slightly modified to adapt to the given corpus. Although the rules can be context free (this is why the set of these rules is named Context Free Grammar Library) they often are as simple as a list of words.

```
ROOMS
        (hotel room)
        (reservation)
        (room)
        (rooms)
        (twin room)
        (suite)
        (suite room)
        (vacancies)
        (vacancy)

CURRENCY
        (cents)
        (dollar)
        (dollars)
        (euro)
        (euros)
        (pounds)
...
```

Intending to make the system more universally compatible, a grammar format different from SOUP has been adopted for writing the rules of the CFG Library, namely the Java^{TM} Speech Grammar Format (JSGF), which defines a BNF-style[2], platform-independent, and vendor-independent Unicode representation of grammars. Therefore, a conversion from SOUP to JSGF has been made. Some rules within this Library are *domain dependent* (e.g. those for ROOMS, OBJECT, LOCATION, etc.) while others are *domain independent* (also called cross-domain, e.g., CURRENCY, PRICE, TEMPORAL, etc.).

[2] BNF is the "Backus-Naur Form", an often used grammar notation: non-terminals are delimitated by '<' and '>', the symbol '=' is used to separate LHS and RHS expressions, rules with the same LHS are grouped into a single BNF definition separating each with '|', etc.

3.1.2 Annotation of the BTEC corpus

Once the semantic schema and its corresponding rules in the CFG Library are defined, we are able to semantically annotate the corpus that will be used for training.

The BTEC corpus is a multilingual text corpus with spoken dialogues of basic tourism and travel situations that was provided with a manual classification between the following subdomains:

ID	SUBDOMAIN	# OF SENTENCES
01:	Restaurant	15,061
02:	Airlines, airports	15,538
03:	Emporium	18,087
04:	Drinkery	2,346
05:	Bank	2,016
06:	Post office	1,817
07:	Hospital	9,992
08:	Personal services (haircut, etc.)	1,480
09:	Transportation	18,926
10:	Travel	23,963
11:	Others	28,467
12:	Hotel	16,741
13:	Security and problem-solving	7,886
Total:		162,320

In this thesis we have implemented a system for the hotel-reservation domain called hotel domain (ID=12) in the classification of the BTEC. Not all sentences have a translation into all languages of the corpus. In our case, only 8,777 sentences from the 16,741 of the hotel domain have all their English, Chinese and Spanish versions.

As usual in any initial contact with a given corpus, a first cleaning and normalization step has been necessary to eliminate some detected errors and to transform some punctuations marks or word/letter writing forms (e.g. abbreviations or accents) into a single known, conventional form. It is false that the more data for training, the better. Corpora should match the demanded domain and be consistent. Having less data is better than to have large amounts of data with erroneous material that may cause language model pollution [Füg04]. In our case, the corpus cleaning and normalization has been very time-consuming, since many errors were found, specially for the Spanish version.

Then, the sentences have been semantically annotated against the schema. To this end, the sentences were read one by one and each of them was assigned to one of the 26 tasks from the schema. The words of each sentence that matched any of the slots of its task, have been annotated as well, assigning them their corresponding slot tag. The following box shows three examples. The annotation has been performed also in XML format.

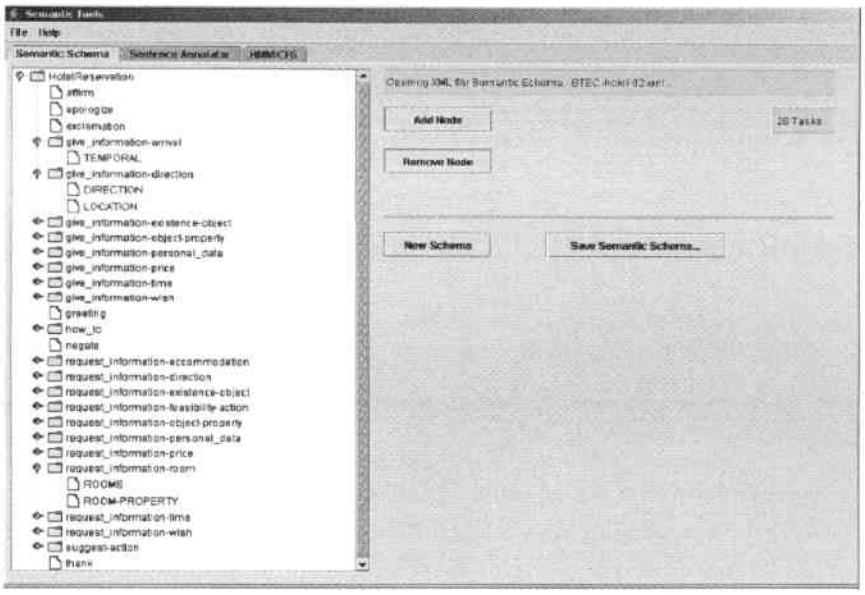
```
<request_information-room text="i'd like a twin room please">
      <ROOMS text="twin room"/>
</request_information-room>

<request_information-room text="i want a suite with a shower">
      <ROOMS text="suite"/>
      <ROOM-PROPERTY text="shower">
</request_information-room>

<thank text="thank you sir">
</thank>

...
```

The slots are in order of apparition in the original sentence and there is no limit in the number of slots a sentence can have. Totally, **5,667 sentences** have been annotated from the hotel domain of the English version of the BTEC corpus with the help of *SemanticTools*.

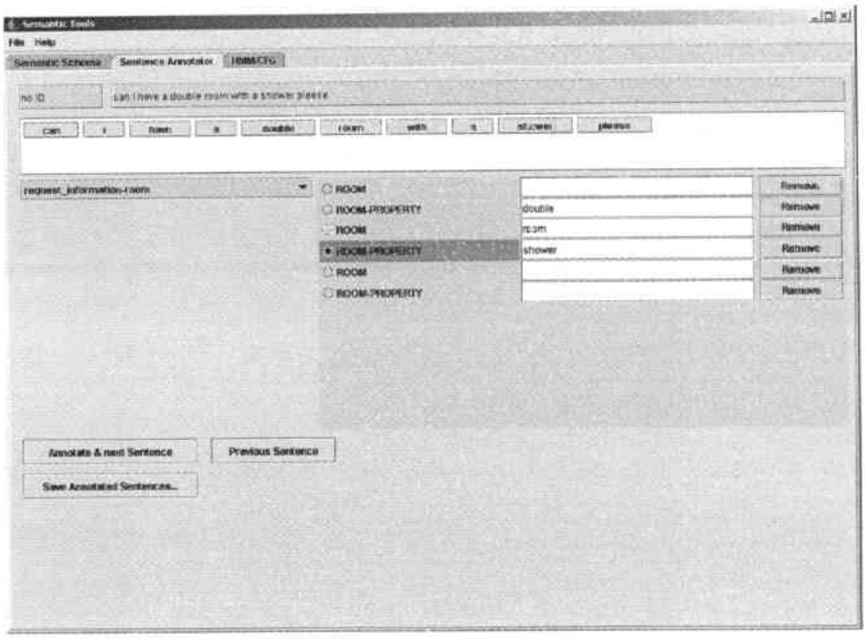
3.1.3 Semantic Tools

The manual annotation of sentences is one of the most time-consuming and exhausting tasks of the text pre-processing step. Therefore, the program *SemanticTools* has been written to make it easier and faster. *SemanticTools* has been programmed in Java using the Eclipse Platform 3.0.1. The graphical user interface (GUI) of *SemanticTools* permits a simple way for creating the semantic schema and to annotate the training sentences accordingly. It consists of two panels:

- *Semantic Schema*: this panel allows the user to create new semantic schemas or modify old ones. The schema is represented in a graphical tree form, but it can be automatically saved in a file in XML format.



- *Sentence Annotator*: in this panel the sentences of the corpus can be assigned to any of the tasks defined in the semantic schema with the help of a graphical checkbox component. The sentences are split into words and each word can be separately selected with a click of mouse and assigned to a slot of the chosen task. The annotation is saved in a file in XML format.



no ID | can I have a double room with a shower please

can | i | have | a | double | room | with | a | shower | please

request_information-room

○ ROOM		Remove
○ ROOM-PROPERTY	double	Remove
○ ROOM	room	Remove
● ROOM-PROPERTY	shower	Remove
○ ROOM		Remove
○ ROOM-PROPERTY		Remove

The importance of this tool relies in the user friendly framework that allows a convenient annotation of the corpus. The use of the mouse makes the typing unnecessary and thus the process very fast.

3.2 Building and training the model

Once the semantics of the domain is defined, the model has to be built and trained. The program *GrammarDeveloper* was written for this purpose.

3.2.1 Grammar Developer

The input of the application consists of the semantic schema (in XML format), the CFG Library (in JSGF format) and the annotated sentences (also in XML format). The schema and the rules are used to create the model structure and the sentences are the data for estimating the model probabilities. The output of *GrammarDeveloper* is a trained language model consisting of a JSGF file and many associated n-gram files (in ARPA[3] format).

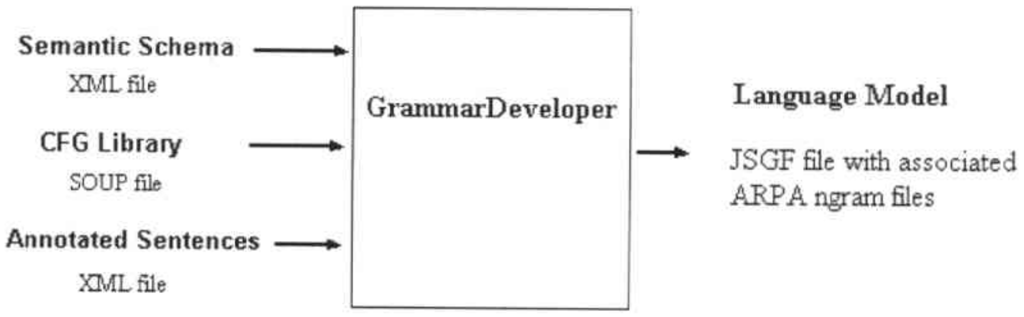


Figure 3.1: Black box diagram of *GrammarDeveloper*

3.2.2 Creating the model structure

The semantic schema, which is solely a list of tasks with concepts within each of them, has to be transformed into a structure or a net that physically describes the different sentences covered. This structure requires an initial point, an end point and a wide variety of paths. The language model created herein will be used by a sentence-based speech recognition decoder. Thus, each recognized sentence will follow a route through the model from the initial node to the end one.

Since the semantic schema is language independent and, therefore, does not define syntactic rules, we do not have any clues on how the structure of the model has to be only with the available semantic information. There are only rules for the slots, i.e. for the concepts. This means that with the CFG Library only the formation rules for the slots of each task are defined, and the rest of the sentence remains undefined.

[3] ARPA stands for Advanced Research Project Agency

We will build is a net with nodes and transition arcs between them. Each node can be seen as a "state" of the recognition process that covers some part of the sentence being recognized. The different word sequences each node can generate (or recognize; from now on, we will speak about the model in a dual way: *recognizing* or *generating* a sentence) will be either "learned" using the training algorithm described in the two next sections (3.2.3 and 3.2.4) or will be ruled by the CFG Library. The transitions between nodes will also be assigned to a probability estimated by the training as well.

The resulting structure is a mixture of a Hidden Markov Model (HMM) and a Context Free Grammar. The nodes corresponding to a concept of the semantic schema are modeled by a rule in the CFG Library (e.g. ROOMS) and all other nodes are left for the training to define what word sequences they can cover. This is the justification for the name of this thesis. Another possible name could have been "A probabilistic context free semantic grammar for language modeling".

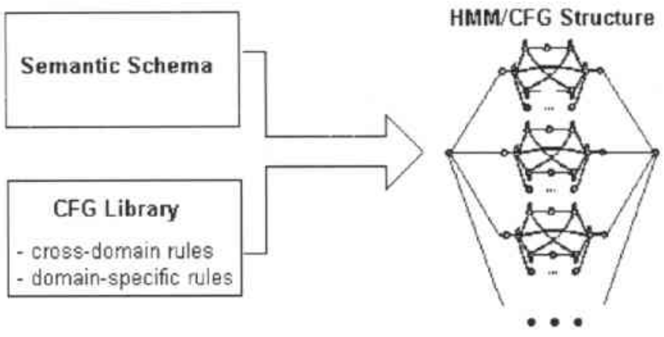


Figure 3.2: Creation of the HMM/CFG structure

The HMM/CFG structure will have as many branches as tasks in the schema. And each task will have as many sub-branches as slots (or concepts) defined for it in the schema. Since we do not want to introduce any previously established syntactic rules for the sentences in each task (besides the CFG nodes, also called *fixed nodes*) we will place non-fixed nodes (all other nodes that are not fixed) before and after each fixed node. Additionally, there will be an initial and an end non-fixed node for each task.

We will illustrate this with an example. Figure 3.3 shows the part of the HMM/CFG structure that corresponds to the task "request_information-room", which has two slots: ROOMS and ROOM-PROPERTY. There is a fixed node for each slot (those with non-rounded angles, NT3 and NT6) plus two additional Pre- and Post-nodes. There are two more nodes, "Pre-request_informarion-room" and "Post-request_information-room" at the beginning and at the end of the task.

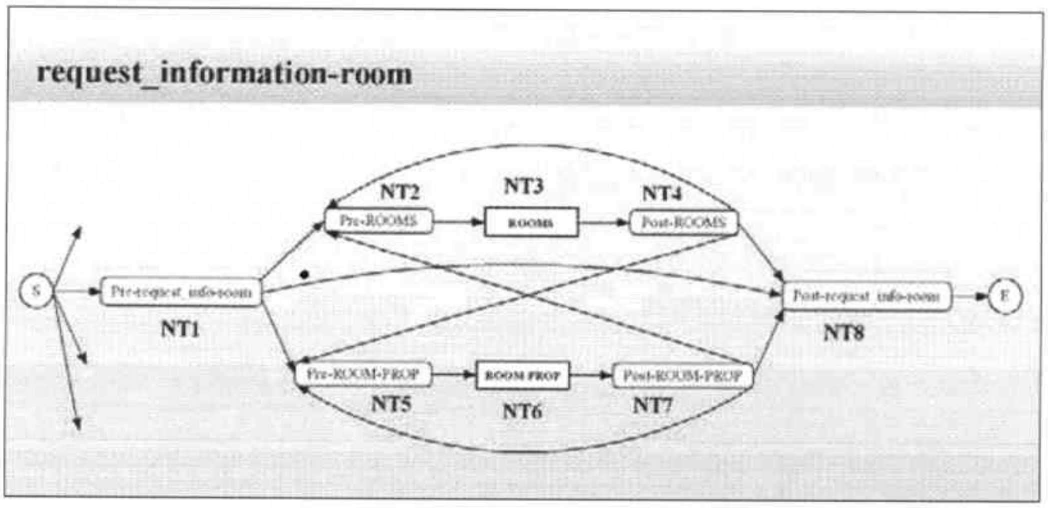


Figure 3.3: HMM/CFG structure for the request_information-room task

As shown in the picture, not all transitions are allowed. Once we are at the Pre-task node (NT1), we can choose to go up to the ROOMS-branch, down to the ROOM-PROP-branch or directly to the Post-task node (NT8). In this last case, the path through this task is completed with the last transition to the end node. In other cases, a Pre-slot has to be gone through before going into the slot (looking at the corresponding rule in the CFG Library). After that comes the Post-slot node. At this point three possibilities emerge: repeat the same slot-branch, jump to the other slot-branch or go to the Post-task node and finish the sentence.

Every given sentence from the training corpus has a single possible path through the model, depending on the apparition of words that match with any of the concepts.

Although the training will determine exactly the words that the Pre- and Post-nodes will cover, they have been added with an intention. Pre- and Post-task nodes could respectively cover typical initial words in a sentence, such as "hello", "good morning", etc. and common ending words, for example "please", "sir", etc. Pre- and Post-slot nodes could cover the surrounding words of the concepts (e.g. "with a" seems to be suitable for NT5).

3.2.3 Alignment: Depth First Search

As soon as the HMM/CFG structure is built, it has to be trained (Figure 3.3) with the semantically annotated sentences.

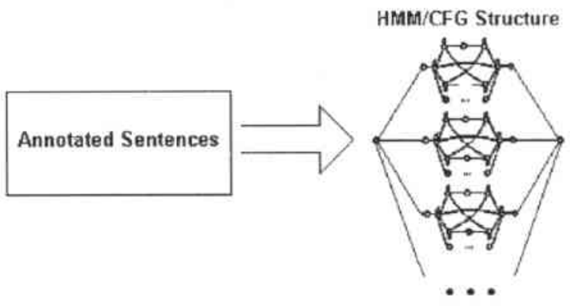
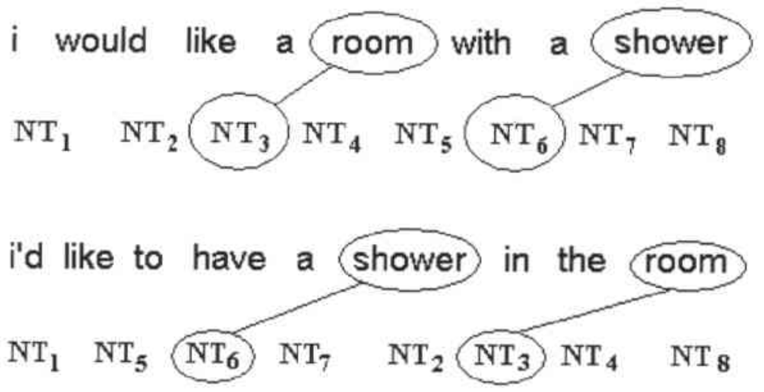


Figure 3.4: Training of the model with annotated sentences

To this end, two steps are necessary: an *alignment* of the sentences with the model and an *estimation* of the model probabilities.

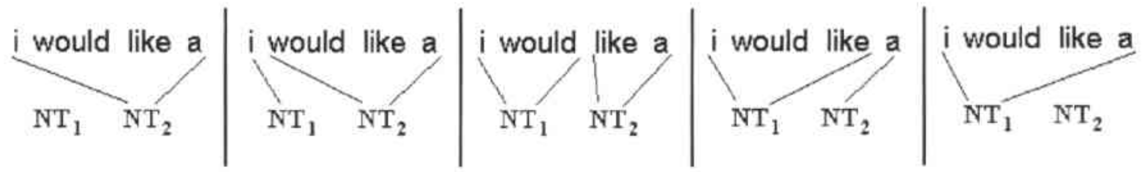
The alignment is simply the process of taking all annotated sentences, one by one, and finding the path through the HMM/CFG that fits with the annotation of this sentence. There is only one possible path for each sentence, since the order of apparition of slots determines the way to go. Consider the following two examples for the task "request_information-room":


i would like a room with a shower
NT1 NT2 NT3 NT4 NT5 NT6 NT7 NT8
i'd like to have a shower in the room
NT1 NT5 NT6 NT7 NT2 NT3 NT4 NT8

where NT_k is the notation for "non-terminal", which is the same as "node" in our case.

Once the alignment with the slots is found (NT_3 [ROOMS] with "room" and NT_6 [ROOM-PROP] with "shower") the rest of the alignment is straightforward: the NT-sequence NT_1 NT_2 gets aligned with "i would like a" and the NT-sequence NT_4 NT_5 gets aligned with "with a" for the first sentence. The alignment for the second sentence becomes obvious.

To perform the alignment process, we have programmed a *depth first search*: an algorithm for finding paths through a net. However, this process leaves an **indeterminacy problem**, because the words not aligned with a slot get aligned with **pairs of NTs**, as seen in the two examples above. To obtain a many-to-one alignment (this is exactly what we need, because we ought to know the coverage of each node independently of other nodes), we make a *segmentation* of each word sequence aligned with a pair of NTs and then decide which segment corresponds to which node. Of course, there are many segmentations for each word sequence. For example, for the word sequence "i would like a":



A priori, we cannot decide between any of these five possible segmentations, because there is no explicit data for discriminating them and selecting the "good" one.

If we look at the second sentence "i'd like to have a shower in the room", we see that the initial words "i would like" (with the contraction expanded) are the same for both sentences. Thus, we would probably say that the "logical" segmentation is the fourth one, aligning "i would like" with NT_1 and letting the remaining words "a" and "to have a" align with NT_2 and NT_5 respectively.

However, this hypothesis is only based on the two example sentences given before and depends solely on intuition. The alignment is made with more than 5,000 annotated sentences, and there is no straightforward way to identify the best segmentation.

The adopted solution to this indeterminacy problem consists in generating **all possible segmentations** of each word sequence (as done before) and align each of them to the corresponding nodes. The algorithm used in the training step will then take care of the problem. The result of the alignment process for the task "request_information-room" is displayed as shown in Figure 3.5.

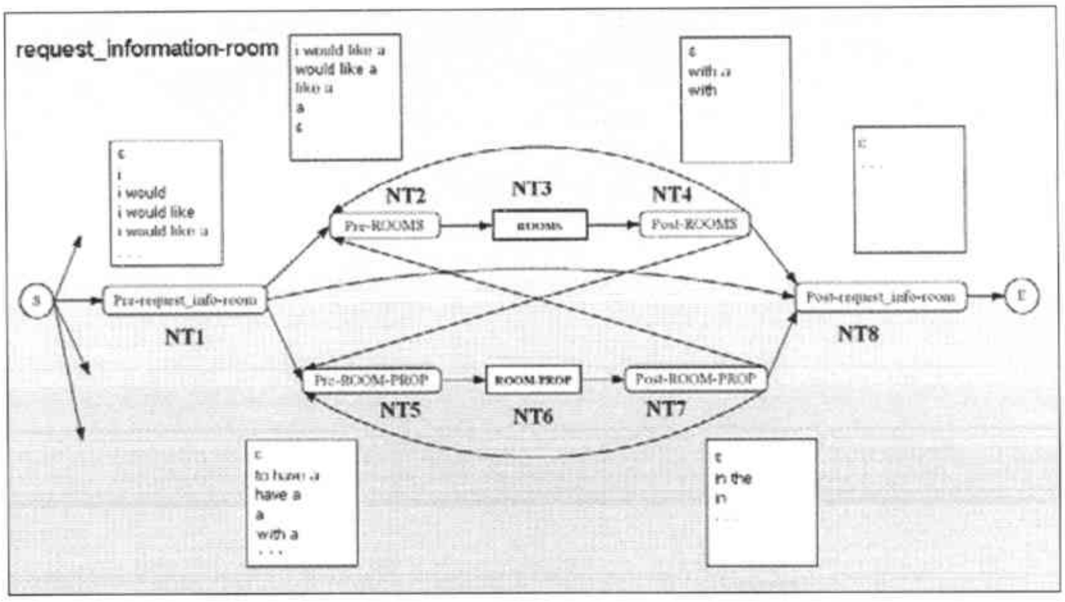


Figure 3.5: Alignment results for request_information-room

During the alignment, the word sequences resulting from the segmentations are accumulated in each slot and saved for the training step. The special character "ε" (Epsilon) represents the *nullstring*, being simply a sign to express that no word has been aligned with this node in a segmentation (the first and the last segmentations of the example for "i would like a" make NT_1 and NT_2 get aligned with ε respectively).

3.2.4 Probability estimation: Expectation Maximization Algorithm

After the alignment completed, the model has a defined structure with the rules for each *slot node* available in the CFG Library and a list of word sequences for each *non-fixed node*. This list of word sequences represents the coverage of each node, but does not have a probability distribution for them. Until now, all members of the list are equally probable in the node where they are. All transition arcs of the model are equally probable as well. A graphical representation of the whole model with the probabilities to be computed is shown in Figure 3.5. In this drawing all nodes are round for simplicity.

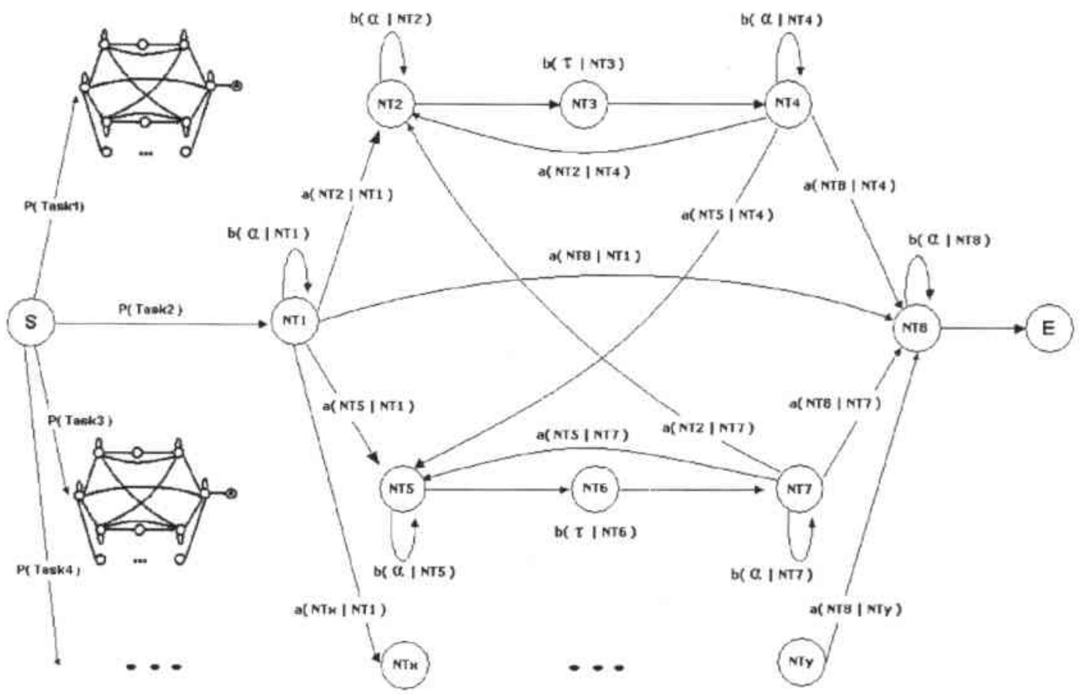


Figure 3.6: Generic HMM/CFG structure with transition and emission probabilities

From the picture, four different types of probabilities can be distinguished:

- task probabilities: P(Task1) ... P(TaskN)
- transition probabilities: a(NT_s | NT_t) between connected nodes
- CFG probabilities: b(τ | NT_k) $\forall$ slot nodes
- emission probabilities: b(α | NT_k) $\forall$ non-fixed nodes

where τ stands for any rule in NT_k and α is any of the word sequences assigned to NT_k as a result of the segmentations of the alignment.

The estimation of these probabilities uses the results of the alignment as the information to calculate the statistical parameters. The available data after the alignment is:

- paths of each sentence through the model
- lists of word sequences in each non-fixed node
- C(NT_s NT_t, w): the *count* of how many times each word sequence *w* has been aligned to the pair of nodes NT_s NT_t

With the information of the paths of sentences obtained from the alignment, we estimate task probabilities, transitions probabilities and CFG probabilities with a Maximum Likelihood estimation (ML). The formulas are:

Given the annotated corpus of sentences S=$\{s_1, s_1, ..., s_L\}$

$$\hat{P}(Task_i) = \frac{C(Task_i \to s)}{L} \tag{3.1}$$

(divide the count of sentences asigned to Task$_i$
by the total number of sentences in the corpus)

$$\hat{a}(NT_t \mid NT_s) = \frac{C(NT_s\ NT_t, w)}{\sum_{\forall k} C(NT_s\ NT_k, w)}$$

(divide the count of word sequences aligned with NT$_s$ and NT$_t$ by the total count of word sequences aligned with NT$_s$ and any other NT) (3.2)

$$\hat{b}(\tau \mid NT_k) = \frac{C(NT_k \to \tau)}{\sum_{\forall i} C(NT_k \to \tau_i)} \tag{3.3}$$

(divide the count of the rule τ in NT$_k$
by the total count of rules in this node)

If the counts of the *optimal* segmentations of the word sequences with respect to the pairs of nodes were available, the emission probabilities would be straightforward to estimate with a simple ML equation:

$$\hat{b}(\alpha \mid NT_k) = \frac{C_{opt}(NT_k \to \alpha)}{\sum_{\forall i} C_{opt}(NT_k \to \alpha_i)} \tag{3.4}$$

However, the optimal segmentations are not available. We consider them as existing, but not directly known from the data. These optimal segmentation counts are exactly the *hidden variables* that the Hidden Markov Model has to encode to estimate the emission probabilities of each non-fixed node.

Thus, the Expectation Maximization algorithm (EM-algorithm) [DLR77] has been implemented to estimate the counts derived from these optimal segmentations. These estimated counts will be called *expected counts*. The resulting estimation formula is:

$$\hat{C}_{\mathrm{EM}}(NT_k \rightarrow \alpha) = \sum_{N,w} C(N,w) \, \frac{P(NT_k \rightarrow \alpha)}{\lambda_1^m(1,n+1)} \sum_{ij:\alpha=w_i \ldots w_j} \lambda_1^{k-1}(1,i)\lambda_{k+1}^m(j+1,n+1) \qquad (3.5)$$

where $\lambda_s^t(p,q)$ is the probability that the NT-sequence NT_s NT_t covers the word sequence $w_p \ldots w_{q-1}$

A more detailed description of the components of the formula for the expected counts can be found in the Appendix and the mathematical deduction for it is described in [WA03a].

Using the expected counts instead of the optimal counts (which we do not have), the ML estimate of the emission probabilities is:

$$\hat{b}(\alpha \mid NT_k) = \frac{\hat{C}_{\mathrm{EM}}(NT_k \rightarrow \alpha)}{\sum_{\forall i} \hat{C}_{\mathrm{EM}}(NT_k \rightarrow \alpha_i)} \qquad (3.6)$$

Since the algorithm to compute the expected counts is *iterative*, we prefer to talk about *training* instead of *estimation*. Figure 3.7 shows the whole iterative process:

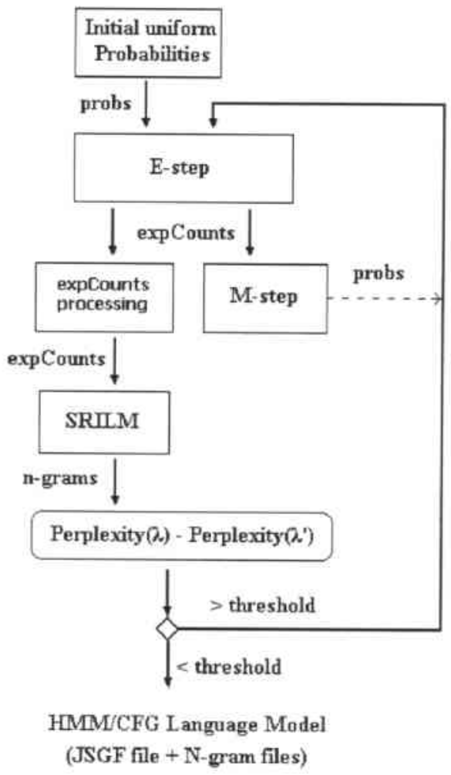

Initial uniform Probabilities
probs
E-step
expCounts
expCounts processing
M-step
probs
expCounts
SRILM
n-grams
Perplexity(λ) - Perplexity(λ')
> threshold
< threshold
HMM/CFG Language Model (JSGF file + N-gram files)

Figure 3.7: Flow diagram of the training of the model

The first task in the algorithm is to initialize the emission probabilities $b(w_i, NT_k)$ with a uniform distribution. After that, the algorithm enters into a loop with basically three steps:

- **E-step:** the expected counts are estimated from the data and from the emission probabilities of the last iteration.
- **M-step:** the new emission probabilities are computed by normalizing the expected counts resulting from the E-step.
- **SRILM:** after a processing of the expected counts, the SRI language modeling toolkit [Sto02] is used to create the n-gram files considering the expected counts as normal counts.

After each iteration, the perplexity difference between the actual model and the model from the previous iteration is computed. If this difference is grater than a prescribed threshold, another iteration is conducted. Otherwise, the estimation improvement is considered sufficiently small and the training is finished. The training can also stop if a maximum number of iterations reaches a limit specified in the configuration file.

The figure below represents the finished model, with its transition probabilities, its n-gram files associated to each non-fixed model and the CFG rules associated to the slot nodes:

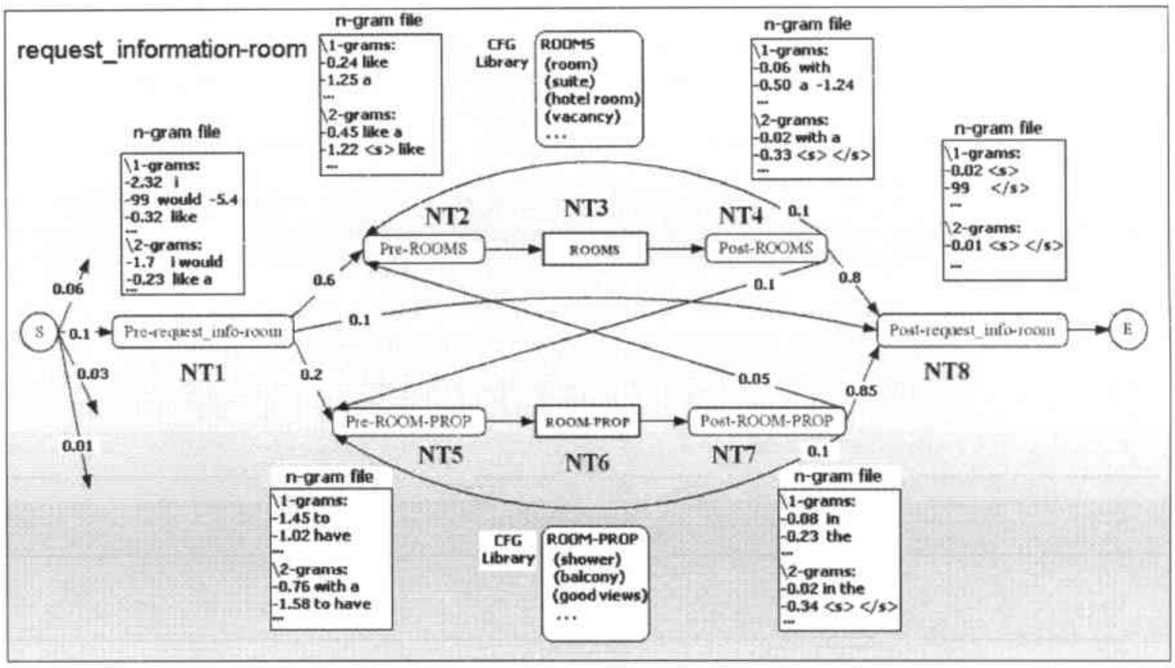


Figure 3.8: Trained HMM/CFG

Although not shown in the picture, each CFG rule has also its estimated probability after the training, as already explained, via an ML estimation.

3.2.5 Tuning the model parameters

GrammarDeveloper has been programmed to easily build new models with other training corpora, other semantics and grammars, and other algorithm parameters. A configuration file serves for this purpose, making the definition and creation of models straightforward. Given a training set and a test set with their corresponding semantic schema and CFG library, the training parameters and the SRILM parameters can be adjusted before creating the model.

Training parameters to be tuned are: threshold for the perplexity difference between iterations, to determine when the algorithm stops; maximum number of iterations allowed, as an alternative method of finishing the training; and the parameter indicating if the CFG rules probabilities are to be trained from the data or if they should be set with uniform distribution.

As any n-gram estimation tool, SRILM has multiple parameters that can be tuned. Among them: the order of the n-grams (value of 'n'); the smoothing method used; and the inclusion of the '<UNK>' token for unknown words. Additionally, our system permits choosing between local node, local task and universal vocabulary for use in the n-gram estimation.

3.3 Porting the model into other languages

The model has been designed to be easily ported into other languages and domains. Since the semantic schema is language independent, once it is defined for a domain, it can be used for other languages by only changing the CFG Library, annotating the sentences in the new language and training the model again.

In this thesis, only a model for English in the hotel-reservation domain has been implemented. Future work will test the system in Spanish and Chinese as well.

4 Test and Evaluation

Although the model built has been tested in a speech recognition system, it could have been readily tested in a topic detection engine, in a speech translation system, in a question answering system, or in any other language processing system that can make use of a semantic-based language model. An integration process would be needed to make the model readable by the system and compatible with its internal framework.

In this Chapter, the test of our HMM/CFG language model performed with the Janus Recognition Toolkit (JRTk) [FGH+97] is described. Its actual decoder [SMF+01], called the Ibis decoder, allows using context free grammars as well as statistical n-gram language models as linguistic knowledge sources [FSS+03] and its last version (August 2006) permits the use of both at the same time in a hybrid linguistic knowledge source called *UnifiedLM*.

This Chapter also describes the making of the test-set and the evaluation and analysis of the results in terms of Perplexity (PPL), Word Error Rate (WER) and Task Error Rate (TER).

4.1 The Janus Recognition Toolkit

The JRTk has been developed by the Interactive Systems Labs at the Universität Karlsruhe (TH) in Germany and at the Carnegie Mellon University in the USA. It is part of the JANUS speech-to-speech translation system [LWL+97].

It provides a flexible Tcl/Tk[4] based environment enabling researchers to build state-of-the-art speech recognizers and to develop, implement, and evaluate new methods. This environment implements an object oriented approach that, unlike other toolkits, is a programmable shell with transparent, yet efficient objects, rather then a set of libraries and precompiled modules.

4.1.1 The Ibis decoder

The version of Janus used in this thesis is V5.0 patch-level 14 which features the Ibis decoder. This is a one-pass decoder based on a re-entrant single pronunciation prefix tree and uses of the concept of linguistic context polymorphism. It is, therefore, able to incorporate full linguistic knowledge at an early stage. It makes possible to decode in one pass, using the same engine in combination with a statistical n-gram language model as well as context free grammars. It is also possible to use the decoder to rescore lattices very efficiently. This results in a faster process compared to the decoder in previous versions of the JRTk which needed three passes to incorporate full linguistic knowledge. The object hierarchy for the Ibis decoder looks as follows:

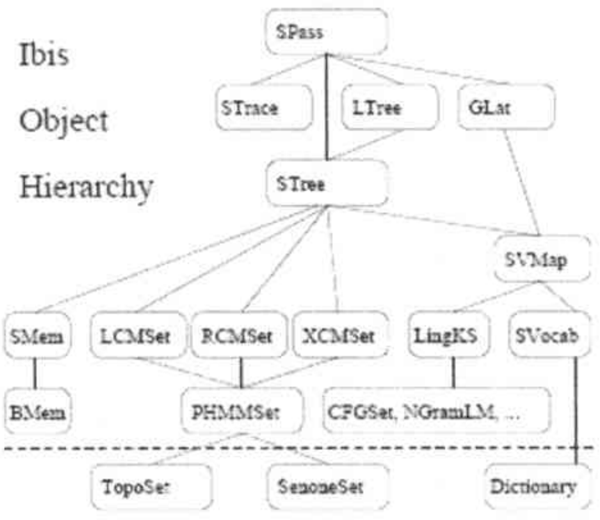


Figure 4.1: The Ibis Object Hierarchy

[4] Tcl/Tk is a free script language for general purposes available at http://www.tcl.tk/

The language model interface in the Ibis decoder is built with a linguistic knowledge source object called *LingKS*. This object constitutes a common interface for all types of language models, e.g. n-gram LMs, CFG grammars, phrase LMs, interpolation of several LingKSs, etc.

The CFG framework of Ibis supports the following formats: SOUP, JSGF, PFSG (Probabilistic Finite State Graph format, which is used by SRILM) and FSM (AT&T Finite State Machine text file format). This framework can be used both for decoding and parsing.

4.1.2 The UnifiedLM linguistic knowledge source

Since the format of our HMM/CFG model was not supported by the Ibis decoder, a new object has been programmed by Christian Fügen: the *UnifiedLM*. The LMSet object was used for using UnifiedLM as a special linguistic knowledge source of Ibis. The LMSet is implemented as a set of linguistic knowledge sources of the type CFG and n-gram representing a separate linguistic knowledge source, whereby the LingKSs can be linked together with the help of tags used in the CFGs or n-gram LMs. This enables references in one language model to other sub language models.

For the UnifiedLM the "head" LM is a CFG consisting of the specification of the different speech acts and several n-gram "sub" LMs specifying the different pre- and post-slots and pre- and post-tasks.

Additionally, a special modality of the search algorithm has been programmed in order to search paths through the complex structure of the model and compute their probabilities (or their score, defined as the negative logarithm of the probability). The search algorithm used for this purpose is a breadth first search.

4.2 Test set

A set of 150 sentences has been manually selected from the BTEC corpus, trying to cover all 26 tasks defined in the semantic schema and at the same time be sufficiently varied. The selection was made long before the EM-algorithm was programmed with no consideration of any other criteria. For those tasks with more training sentences, more test sentences were also selected. The following table shows the distribution of the test sentences among the tasks compared to the number of training sentences:

Task name	# test sentences	# training sentences
affirm	7	98
apologize	3	47
exclamation	0	29
give_information-arrival	4	30
give_information-direction	5	33
give_information-existence-object	1	174
give_information-object-property	7	344
give_information-personal-data	6	197
give_information-price	5	44
give_information-time	4	69
give_information-wish	9	547
greeting	10	106
how_to	4	86
negate	1	51
request_information-accomodation	3	98
request_information-direction	6	364
request_information-existence-object	8	401
request_information-feasibility-action	5	422
request_information-object-property	2	22
request_information-personal-data	11	270
request_information-price	9	304
request_information-room	12	458
request_information-time	11	360
request_information-wish	1	89
suggest-action	13	949
thank	3	75
TOTAL	150	5667

The 150 sentences have been recorded for four different speakers: one female and three males. Thus, the test set consists of a total of 600 sentences corresponding to about **one hour of recorded speech**. The recording tool used was the version 0.9 of the Data Collection Tool by Tanja Schultz and expanded by others. The recordings were made with a headset microphone from Plantronics and the external sound card TELEX H-551 Digital. The recording format is: 16KHz sample rate, Lin16 sample encoding, one mono channel and Little Endian in the byte order format.

The *acoustic model* that we have used for our experiments was trained on nearly 95hrs of close talking meeting data mixed with 180hrs of Broadcast News data. It is a slimmed down version of a system, which was used in the NIST's RT-04S[5] evaluation [MJF+04]. It is a fully continuous system consisting of 6000 codebooks with 185k Gaussians over a 42-dimensional feature space based on MFCC (mel-frequency cepstral coefficients) after *linear discriminant analysis* and global *semi-tied covariance* transforms with utterance based *cepstral mean subtraction*. Incremental constrained MLLR (maximum likelihood linear regression) is used in decoding to compensate for different channels effects.

Using a Pentium 2.80GHz with 1GB RAM memory, the *training* of the **HMM/CFG language model** with all features included, has an approximate duration of **two hours** for 45 iterations. However, the stabilization of the probabilities already takes place around the 30th iteration. The *decoding* of the test-set has a duration of about **twelve hours** using the prototype implementation of the UnifiedLM. This decoding module will be optimized in the future to reach a reasonable working speed.

4.3 Evaluation

The test results of the test are evaluated in terms of perplexity (PPL), word error rate (WER) and task error rate (TER) and compared to a stand-alone n-gram language model.

[5] Rich Transcription 2004 Spring Meeting Recognition Evaluation conducted by NIST in the USA

4.3.1 Definition of the evaluation measures

The *perplexity* of a probabilistic model (rule-based models cannot be evaluated with PPL) can be interpreted as the "predictability" of this model towards a test set of sentences. It is usually a good indicator for the quality of the model, though it is not always correlated with other parameters such as WER or with language understanding rates. In Chapter 2, equation 2.5, can be found the mathematical definition of the perplexity.

The *word error rate* is based on the minimal edit distance between hypothesis and reference sentence, this means it is based on the minimal number of substitutions s, insertions i and deletions d necessary to transform the hypothesis into the reference. Being n the number of reference words, the WER, in percent, is computed as follows:

$$WER = \frac{s+i+d}{n} * 100\% \tag{4.1}$$

The *task error rate* is defined as the percentage of sentences that were not aligned with the same task as the reference annotation:

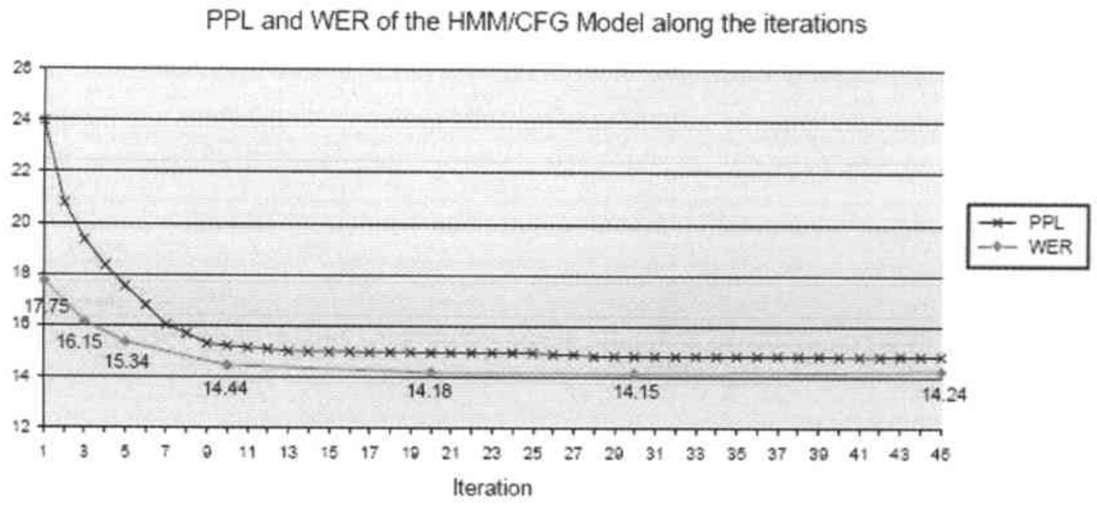
$$TER = \frac{\text{\# task recognition errors}}{\text{\# tasks}} * 100\% \tag{4.2}$$

4.3.2 Baseline model description

The baseline is a 3-gram model trained with the same corpus used to train the HMM/CFG, i.e. with a part of the BTEC consisting of **5,667 sentences** from the hotel-reservation subdomain which has a vocabulary of **1,593 words**. The smoothing technique is Witten-Bell, the same used for the n-grams of the non-fixed nodes of our system. The tool SRILM was used to compute probabilities and perplexity. A language model weight lz = 32 and a word transition penalty lp = 10 was always used for decoding the baseline and all other models.

4.3.3 Perplexity and Word Error Rate

The training of the HMM/CFG produced 45 language models, one for each iteration. The parameters of the model are re-adjusted in each iteration and thus, an evaluation of intermediate models shows the evolution of the model's quality. The mathematical convergence analysis of the EM-algorithm in [DLR77] shows that it always converges to a local maximum. If we plot the evolution of the perplexity along the iterations and compute the WER for some of them we obtain the following graph:



The curve of the PPL is a clearly descending asymptote with a limit about **PPL = 14.8**. The curve of the WER is also descending with a limit slightly above **14.1 %**. However, a small increase can bee seen between the 30th and 45th iterations. In general, both PPL and WER show a considerable correlation.

Taking some single values from the iterations and comparing them to the baseline 3-gram LM, we obtain the following table:

	Iteration	PPL	WER
HMM / CFG	1	23.97	17.75 %
	3	19.36	16.15 %
	5	17.53	15.34 %
	10	15.20	14.44 %
	20	14.96	14.18 %
	30	14.83	14.15 %
	45	14.82	14.24 %
3-gram LM		11.06	14.16 %

Although the PPL of the HMM/CFG reaches a value more than three points above the baseline, the WER values after the 20th iteration are nearly the same for both models.

Another model has been trained, setting the CFG rule probabilities with *uniform distribution*, instead of training them from the data. The results from the 45th iteration clearly show a poorer performance, indicating that the training of the CFG probabilities makes the model adapt better to the corpus:

PPL	WER
22.13	15.78 %

Therefore, we take the model with trained CFG probabilities as our best system, since its performance is practically **the same** as the baseline LM if evaluated in **recognition accuracy**. In the next section, we examine the performance in terms of **understanding accuracy**.

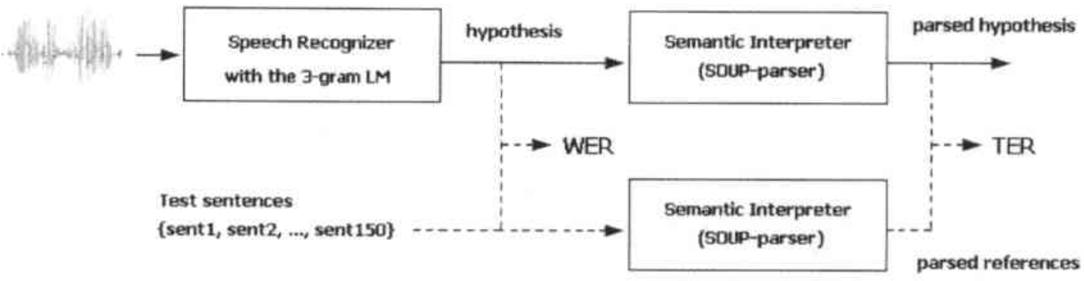
4.3.4 Task Error Rate

Since the structure of our HHM/CFG is semantic-based, the **path** of a hypothesis sentence through the model determines to what **task** the utterance has been aligned to. Thus, the output of the system is not only a sentence, but also a semantic interpretation of the sentence. Using equation 4.2 to calculate the task error rate, we obtain the following table:

Iteration	TER
1	16.61 %
3	15.03 %
5	13.11 %
10	13.46 %
20	12.94 %
30	13.28 %
45	13.66 %

The evolution of the TER is descending, though there are some oscillations in the last iterations. Therefore, the correlation between WER and TER is not very high.

The baseline is a 3-gram model, and thus has no semantic output. The only way to compare it with the HMM/CFG in understanding accuracy is creating a two-step system with the speech recognizer and a semantic interpreter after it:



Speech Recognizer with the 3-gram LM
hypothesis
Semantic Interpreter (SOUP-parser)
parsed hypothesis
WER
TER
Test sentences {sent1, sent2, ..., sent150}
Semantic Interpreter (SOUP-parser)
parsed references

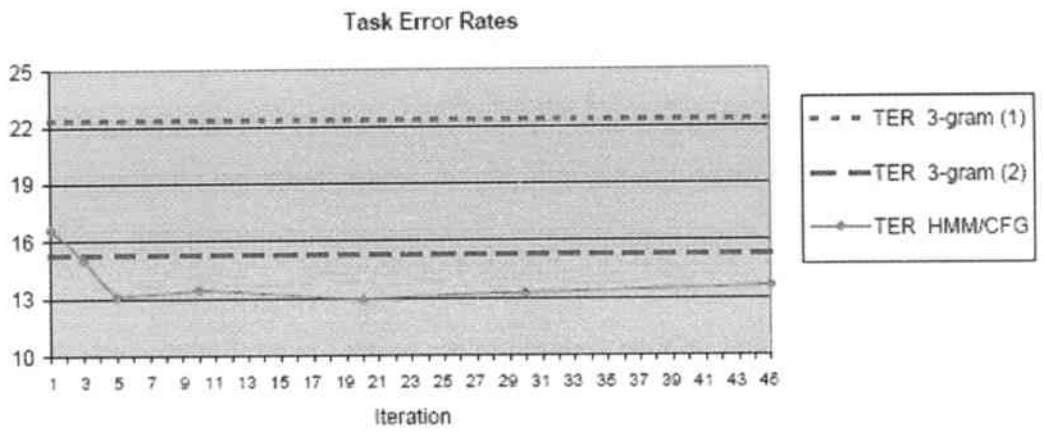
Figure 4.2: Baseline two-step system – decoding + parsing

As seen in Figure 5.1, the test sentences are parsed with the SOUP parser (using the travel grammars referred to in chapter 3.1), as well as the hypothesis sentences. The differences in the parsing results are counted as errors enabling the calculation of a TER. The top level rules of the SOUP grammar are equivalent (though not the same) to the tasks of our semantic schema definition and thus are used to compute the TER. However, there are several ways to determine the TER, because the parsing often assigns more than one top level rule to a single sentence. Therefore, two different TERs have been calculated:

- **TER (1) = 22.33%** (considering all top level rules assigned to a sentence as **"a whole"** and thus counting a *task error* each time any of the top level rules of a sentence has not been correctly recognized)

- **TER (2) = 15.27%** (considering all top level rules of all sentences **separately** and counting a *task error* each time one of them has not been correctly recognized)

The following picture shows the TER values for both the HMM/CFG and the baseline model in a single graph:



As it can be seen, both ways of computing the TER result in a poorer understanding accuracy than that of our HMM/CFG system. Considering the best of both values, TER (2), our one-step speech understanding system has reduced the TER by about **10%** over the baseline two-step system already after the 5th iteration. If we take the model in its best iteration (the 20th iteration) the relative TER reduction over the baseline is of **15.3%**.

This can be considered as a successful result, especially if we understand that the evaluation is somehow biased in favor of the baseline system because the test data annotation for the baseline was not manual, but parsed with the same grammar as the hypothesis sentences. Thus, when ambiguous parses exist, only the one selected by the SOUP parser is considered correct.

5 Conclusions and Perspectives

The last chapter has shown to what extent the system developed in this thesis outperforms the baseline 3-gram model in understanding accuracy. In this chapter we point out some interpretations of this result and suggest possible improvements to the system and further work in a similar research direction.

5.1 Interpretation of the results

The interpretation of the evaluation parameters of both models is simple: even if the HMM/CFG has been designed to optimize the **task** accuracy, it yields similar results as the 3-gram baseline in **word** accuracy. This finding supports to the thesis that: "optimizing a language model for understanding accuracy does not necessarily reduce the recognition accuracy".

It can be added that a semantic-based unified language model additionally outputs the semantic interpretation of the utterance at the same time as the decoding. In other words, a unified language model provides a unified output without impoverishing word accuracy. A conclusion to this is that a one-step system (integrating recognition and understanding in the decoding process) provides better results than a two-step system (separating recognition and understanding in concatenated processes).

Compared to other semantic-based or rule-based systems, the system developed in this thesis has clear advantages:

- it doesn't need very large corpora to be robustly trained
- the semantics are very simple to define
- there is an available tool to annotate the corpus against the semantic schema in a user friendly interface (see chapter 3.1.3 Semantic Tools)
- the output is in the standard Java Speech Grammar Format, though with special tags referring to the non-fixed nodes, having an associated n-gram file each
- spontaneous speech effects can be easily modeled with the HMM/CFG structure because of its data-driven nature
- parsing is done while decoding

In summary, this thesis has shown how a hybrid system that combines statistical and linguistic approaches to language modeling can produce enhanced results and even improve the understanding performance of a speech recognition engine based on purely statistical methods.

5.2 What could be improved?

Although the results of the HMM/CFG have already shown success, several issues remain as possible improvements.

Some of the 26 tasks defined for the hotel domain in this thesis intersect (semantically) with each other. For example, the task "request_information-object-property" can be interpreted as the task "request_information-room", being the instance of the OBJECT slot precisely a word from the ROOM slot. This produces annotation ambiguities and, consequently, interpretation errors. An attempt to define the tasks as totally disjunctive semantic classes could show better accuracy.

There are other ambiguities in the definition of the slots of the semantic schema. For example, the rule defined in the CFG Library for ROOM does include "room", "suite", etc. and the rule for ROOM-PROP includes expressions such as "shower", "double", "twin", "good view", etc. A more compact definition could include words that can form a compound noun into ROOM, e.g. "double room" or "twin room". This is, again, a matter of *guess and check*, though linguistic knowledge could direct these decisions.

The semantic output of our system is limited and rather simple. Only top-level concepts (tasks) and some more specific concepts (slots) are given as the semantic interpretation of each utterance. A more detailed semantic schema with elaborate CFG rules could be defined to obtain a more complete understanding system.

Another improvement to our system could be to use some smoothing technique to estimate the probabilities of the model; not only the transition and CFG rule probabilities, but also those computed in the E-Step of the training algorithm. In this last case, though, it is not clear if the convergence of the EM-algorithm could be preserved.

Some changes in the structure of the HMM/CFG can also be examined. For example, the pre- to post-node transition could be optionally deactivated forcing each sentence to have at least one of the concepts defined in its corresponding task.

The optimization of the search algorithm and of the model structure should also be investigated, making the system suitable for real-time applications. The actual decoding speed is still too slow: around 13 x real-time.

Additionally, an automatic annotation of the training corpus could be tried, to render the system even more independent of human assistance.

5.3 Further work

Six years ago, Rosenfeld wrote:

> "[…] it could be argued that attempts to integrate linguistic knowledge into our models have so far failed because we don't know how to appropriately encode such knowledge, namely, how to optimally combine it with data. Put yet another way, we haven't figured out how to simultaneously get the most out of both our knowledge and our data. Between knowledge without data and data without knowledge, apparently the latter (witness the n-gram) is more successful. But there is no inherent reason why we can't have both.“ [Ros00b]

Even though the language model implemented in this thesis is intended for a specific domain -while Rosenfeld's affirmations address principally large vocabulary language modeling- it is an initial contribution towards a unified view and implementation of language modeling. There is still a long way to go in this direction.

If researchers adopt this understanding-driven orientation of language modeling in further work, it may be necessary to define a unified parameter for evaluating both word accuracy and semantic interpretation simultaneously. Especially when the training of the model is designed to estimate the joint probability P(W, Syn, Sem), a joint error rate seems to be the fairest and most coherent evaluation measure.

Many experiments and new approaches can be attempted in the future: probabilistic LR grammars, link grammars, support vector machines for language modeling, dynamic Bayesian networks, named entity modeling, prior knowledge coding, etc. All these are fields and/or techniques that still can be investigated to improve the actual language modeling performances. In particular, dynamic structures that can self-organize themselves *on the fly* and acquire new domain knowledge are promising trends.

One can never forget that language modeling deals with language (sic). With this I mean that the *study of language* has to go to its place within the field of language modeling for being able to encode the linguistic knowledge and profit from it. However, there are many detractors of this view.

I would like this diploma thesis to provide a little encouragement to those who are skeptical about the role of prior linguistic knowledge in the future of speech and language technologies.

Appendix

How the Expectation Maximization algorithm works

The EM-algorithm is a general method of finding the maximum likelihood estimate of the parameters of an underlying distribution from a given data set when the data is **incomplete** or has **missing values**. Its application for Hidden Markov Models is known as the *Baum-Welch algorithm*.

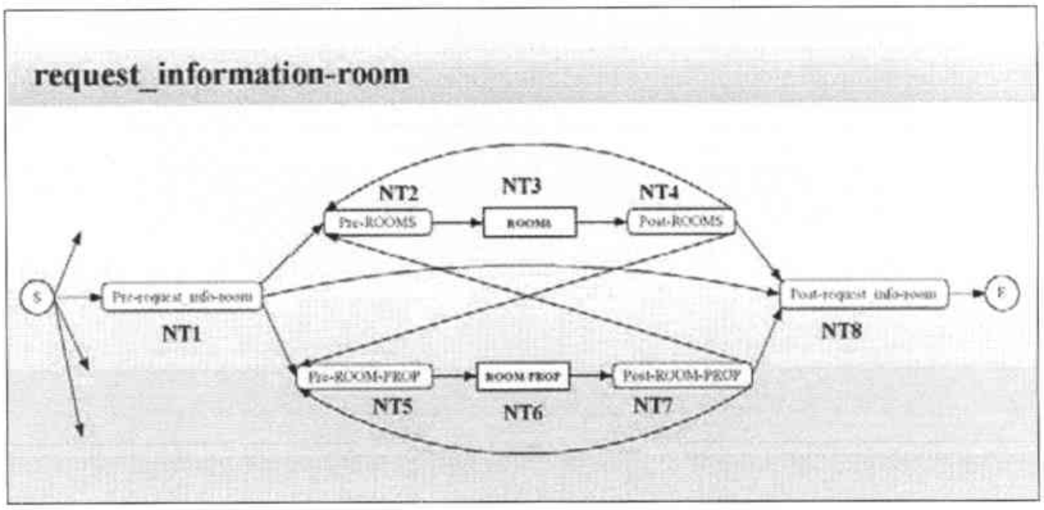
In this thesis the algorithm has been implemented to estimate the probabilities of the word sequences covered by each node of the model. The mathematical derivation of the formula can be found in [WA03a]. In this appendix we will only illustrate how the formula is applied with a simple example and briefly explain how the formula can be intuitively understood.

We consider a very small training set consisting of 5 annotated sentences:

```
<request_info-room id="S1" text="i want a room with a clean bath">
        <ROOMS text="room"/>
        <ROOM-PROPERTY text="clean bath"/>
</request_info-room>


<request_info-room  id="S2"  text="i  would  like  a  twin  room  with  bath
please">
        <ROOM-PROPERTY text="twin"/>
        <ROOMS text="room"/>
        <ROOM-PROPERTY text="bath"/>
</request_info-room>
```

```
<request_info-room id="S3" text="i would like a double room with a shower
please">
        <ROOM-PROPERTY text="double"/>
        <ROOMS text="room"/>
        <ROOM-PROPERTY text="shower"/>
</request_info-room>


<request_info-room id="S4" text="i would like a room with bath please ">
        <ROOMS text="room"/>
        <ROOM-PROPERTY text="bath"/>
</request_info-room>


<request_info-room id="S5" text="can i have a suite with bath please">
        <ROOMS text="suite"/>
        <ROOM-PROPERTY text="bath"/>
</request_info-room>
```

All 5 sentences are annotated with the task "request_information-room", which has the following structure:



As described in Section 3.2.3, the annotated sentences are first aligned with the model, finding a path for each of them. Besides of the words that get aligned with the slots, the remaining word sequences, aligned with pairs of NTs, are:

S1: NT1 NT2 → "i want a" NT4 NT5 → "with a" NT7 NT8 → "" (=epsilon) S2: NT1 NT5 → "i would like a" NT7 NT2 → "" (=epsilon) NT4 NT5 → "with" NT7 NT8 → "please" S3: NT1 NT5 → "i would like a" NT7 NT2 → "" (=epsilon) NT4 NT5 → "with a" NT7 NT8 → "please"	S4: NT1 NT2 → "i would like a" NT4 NT5 → "with" NT7 NT8 → "please" S5: NT1 NT2 → "can i have a" NT4 NT5 → "with" NT7 NT8 → "please"

This leads to the following counts (different from the expected counts!):

C(NT1 NT2 → "i want a") = 1
C(NT4 NT5 → "with a") = 2
C(NT7 NT8 → "") = 1
C(NT1 NT5 → "i would like a") = 2
C(NT7 NT2 → "") = 2
C(NT4 NT5 → "with") = 3
C(NT7 NT8 → "please") = 4
C(NT1 NT2 → "i would like a") = 1
C(NT1 NT2 → "can i have a") = 1

Now, all possible segmentations of each word sequence is done and the resulting sub-"word sequences" are added to the list of its corresponding node. Note that the segmentation of a word sequence, e.g. "i want a", does not produce the same sub-"word sequences" for both nodes, in this case NT1 and NT2. For this example, NT1 would get {epsilon, i, i want, i want a} and NT2 would get {i want a, want a, a, epsilon}.

As it can be seen, the nullstring (=epsilon) is added in all nodes to let each node the possibility of emitting nothing at all. The following boxes show the result of these segmentations:

	NT2 epsilon i want a want a a i would like a would like a like a can i have a i have a have a	NT3 [ROOMS] *No word sequences for EM-training, since the probabilities of the CFG rules are estimated with ML.*	NT4 epsilon with with a	
NT1 epsilon i i want i want a i would i would like i would like a can can i can i have can i have a				NT8 epsilon sir please sir please
	NT5 epsilon a with a like a would like a i would like a	NT6 [ROOM-PROP] *No word sequences for EM-training, since the probabilities of the CFG rules are estimated with ML.*	NT7 Epsilon please sir	

From now on, the sub-"word sequences" from the lists in each node will be denoted by an alpha. And so, P($NT_k \rightarrow \alpha$) is the probability that the node NT_k covers the words in α. All these probabilities are initialized with an uniform distribution before the EM algorithm begins, i.e. P($NT_k \rightarrow \alpha$) = 1 / N , where N is the number of α's in NT_k.

As described in Section 3.2.4, the EM-algorithm consists of two steps: the computation of the expected counts and the posterior ML estimation of the probabilities P($NT_k \rightarrow \alpha$), also called emission probabilities. These probabilities have been expressed with the different notation b(α | NT_k) in the third chapter of this thesis, but refer to exactly the same concept.

The computation of the expected count of a specific α whithin a specific node, is made through an iterative process expressed by the following formula:

$$\hat{C}_{EM}(NT_k \rightarrow \alpha) = \sum_{N,w} C(N,w) \frac{P(NT_k \rightarrow \alpha)}{\lambda_1^m(1,n+1)} \sum_{ij:\alpha=w_i \ldots w_j} \lambda_1^{k-1}(1,i)\lambda_{k+1}^m(j+1,n+1) \qquad (7.1)$$

where $\lambda_s^t(p,q)$ is the probability that the NT-sequence covers the word sequence $w_p \ldots w_{q-1}$.

These λs are calculated via dynamic programming (DP) with the expressions:

$$\begin{aligned} \lambda_s^t(p,q) &= \sum_{p \le r \le q} \lambda_s^{t-1}(p,r)\lambda_t^t(r,q) \\ \lambda_s^s(p,q) &= \begin{cases} P(NT_s \rightarrow w_p, \ldots, w_{q-1}) & \text{if } p < q \\ P(NT_s \rightarrow epsilon) & \text{if } p = q \end{cases} \end{aligned} \qquad (7.2)$$

In our example, the expected counts C(NT_1→ ""), C(NT_1→ "i"), C(NT_1→ "i want"), C(NT_1→ "i want a"), ... C(NT_1→ "can i have a"), C(NT_2→ ""), C(NT_2→ "a"), C(NT_2→ "want a"), etc. would be computed. If we apply the formula, say for node NT_1 and α = "i", we obtain the following expression:

$$\begin{aligned} \hat{C}_{EM}(NT_1 \rightarrow \text{"i"}) = C(\text{NT}_1\ \text{NT}_2 \rightarrow \text{"i want a"}) & \frac{P(\text{NT}_1 \rightarrow \text{"i"})}{\lambda_1^2(1,4)} \lambda_2^2(2,4) \quad + \\ C(\text{NT}_1\ \text{NT}_5 \rightarrow \text{"i would like a"}) & \frac{P(\text{NT}_1 \rightarrow \text{"i"})}{\lambda_1^5(1,5)} \lambda_5^5(2,5) \quad + \\ C(\text{NT}_1\ \text{NT}_2 \rightarrow \text{"i would like a"}) & \frac{P(\text{NT}_1 \rightarrow \text{"i"})}{\lambda_1^2(1,5)} \lambda_2^2(2,5) \end{aligned} \qquad (7.3)$$

The expected count can be interpreted as a **weighted sum** of the counts C(NT_1 NT_2 → "i want a"), C(NT_1 NT_5 → "i would like a") and C(NT_1 NT_2 → "i would like a"). These are precisely the counts of the pairs of NTs and word sequences where NT_1 and "i" (in the beginning of the sequence) appear.

In generally, each addend of this sum is multiplied by the following weight:

$$\frac{P(NT_k \to \alpha) \sum_{ij:\alpha=w_i \ldots w_j} \lambda_1^{k-1}(1,i)\lambda_{k+1}^{m}(j+1,n+1)}{\lambda_1^{m}(1,n+1)} \leq 1 \tag{7.4}$$

This weight being a probability by itself, and therefore smaller or equal to one, is composed of three terms:

- P(NT_k→ α): the probability that NT_k covers the word sequence α. In the first iteration this probability is 1 / N, where N is the number of α's in NT_k (in our example this term would be 1/11 for NT_k and "i" in the first iteration). In subsequent iterations this probability will be different, since it will be re-estimated with the expected counts of the previous iteration. All addends of the sum are multiplied by this term.

- $\sum_{ij:\alpha=w_i \ldots w_j} \lambda_1^{k-1}(1,i)\lambda_{k+1}^{m}(j+1,n+1)$: the probability that the **remaining** NTs, i.e. all NTs of the NT-sequence but NT_k (in our example NT_1) cover the **remaining** words of the word sequence, i.e. all words, but those in α (in our example α = "i"). All addends of the sum are multiplied by this term. With the model structure defined in this thesis, there only exist NT-sequences of length 2; thus this term simplifies to only one lambda. In our example, the lambdas $\lambda_2^2(2,4)$, $\lambda_5^5(2,5)$ and $\lambda_2^2(2,5)$ are used, respectively.

- $\lambda_1^m(1, n+1)$: the probability that the NT-sequence taken into consideration for this addend covers the complete word sequence (that is why the indexes go from 1 to n+1). In our example, "complete word sequence" includes "i want", "i would like a" and "i would like a". All addends of the sum are divided by this term.

As it can be seen, the three terms of this weight form a fraction. The **numerator** of this fraction is the probability that the segmentation of the actual word sequence is as follows:

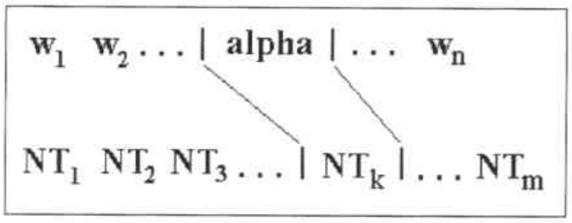

w_1 w_2 . . . | alpha | . . . w_n
NT_1 NT_2 NT_3 . . . | NT_k | . . . NT_m

Figure A.1: Segmentation considering $NT_k \rightarrow$ alpha

This picture is a general representation for NT-sequences of any length. In this thesis only appear NT-sequences of two NTs.

The **denominator** of the fraction is the sum of all probabilities of all possible segmentations (with the segmentation of the picture included). With this interpretation, the ratio (7.4) can be considered as **the probability of the segmentation represented in figure A.1 over all possible segmentations.**

As a conclusion, it can be said that each weight discriminates between the addends and favors those who belong to a word sequence, whose corresponding segmentation is more probable.

References

[Can87] Francisco Canals Vidal, "Sobre la esencia del conocimiento", Promociones y Publicaciones Universitarias, S. A., Barcelona, Spain, 1987.

[CG96] Stanley F. Chen and Joshua Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling" in Proceedings of the 34th Annual Meeting of the ACL, pp. 310-318, Santa Cruz, California, USA, June 1996.

[Cho57] Noam Chomsky, "Syntactic Structures", The Hague: Mouton & Co., 1957.

[Dah94] D. Dahl et al, "Expanding the scope of the ATIS Task: the ATIS-3 Corpus" in Human Language Technology Workshop, 1994.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm" in Journal of the Royal Statistical Society, Series B, Vol. 39, No1, pp. 1-38, 1977.

[FGH$^+$97] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries and M. Westphal, "The Karlsruhe-Verbmobil Speech Recognition Engine" in Proceedings of the ICASSP, München, Germany, 1997.

[Fle04] Falk Fleischer, "Einbettung von Grammatikregeln in n-gram-Sprachmodelle", Studienarbeit, Universität Karlsruhe, Germany, 2004.

[FMS04] Christian Fügen, Florian Metze and Hagen Soltau, "JRTk and JANUS – The Ibis-Gang (IBIS V5.0 P013)", internal documentation of the Janus Recognition Toolkit, Karlsruhe, Germany, 2004.

[FSS$^+$03] Christian Fügen, Sebastian Stücker, Hagen Soltau, Florian Metze, and Tanja Schultz, "Efficient Handling of Multilingual Language Models" in Proceedings of the ASRU, St. Thomas, US Virgin Islands, 2003.

[Füg04] Christian Fügen, "Language Modeling experiences collected over the past evaluations (SWB, MT)", talk given at the Interactive Systems Labs, Pittsburgh, USA, August 2004.

[Gav00a] Marsal Gavaldà, "SOUP: A Parser for real-world Spontaneous Speech" in Proceedings of the International Workshop on Parsing Technologies, Trento, Italy, 2000.

[Gav00b] Marsal Gavaldà, "Growing semantic grammars", Ph.D. Thesis, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2000.

[HHP01] Timothy J. Hazen, I. Lee Hetherington, and Alex Park, "FST-based recognition techniques for multi-lingual and multi-domain spontaneous speech" in Proceedings of Eurospeech, Aalborg, Denmark, 2001.

[Hua01] Xuedong Huang et al, "MiPad: A Multimodel Interaction prototype" in Proceedings of the ICASSP, Salt Lake City, Utah, USA, 2001.

[Jel89] Fred Jelinek, "Self-organized Language Modeling for Speech Recognition" in Readings in Speech Recognition, Morgan Kaufmann, edited by K. F. Lee and A. Waibel, 1989.

[Jel95] Fred Jelinek, closing remarks in the Language Modeling Summer Workshop at Johns Hopkins University, Baltimore, Maryland, USA, 1995.

[Jon06] Rebecca Jonson, "Generating statistical language models from interpretation grammars in dialogue systems" in Proceedings of the EACL, Trento, Italy, April 2006.

[LGW$^+$03] L. Levin, D. Gates, D. Wallace, K. Peterson, E. Pianta and N. Mana, "The NESPOLE! Interchange Format" in the final version of the NESPOLE! IF report, http://www.is.cs.cmu.edu/nespole/, February 2003.

[Lee04] Lillian Lee, "'I'm sorry Dave, I'm afraid I can't do that': Linguistics, Statistics, and natural language processing circa 2001" in Computer Science: Reflections on the Field, Reflections from the Field, pp. 111-118, 2004.

[LWL+97] Alon Lavie, Alex Waibel, Lori Levin, Michael Finke, Donna Gates, Marsal Gavaldà, Torsten Zeppenfeld and Piming Zhan, "JANUS III: Speech-to-Speech translation in multiple languages" in Proceedings of the ICASSP, Munich, Germany, 1997.

[MBIS04] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz, "Hidden Understanding Models of natural language" in Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, New Mexico State University, USA, 1994.

[MJF+04] Florian Metze, Qin Jin, Christian Fügen, Yue Pan, Kornel Laskowski, and Tanja Schultz, "Issues in Meeting Transcription – The ISL Meeting Transcription System" in Proceedings of the ICSLP, Jeju Island, Korea, October 2004.

[Mor98] Antonio Moreno Sandoval, "Lingüística computacional", Editorial Síntesis, Madrid, Spain, 1998.

[Min75] Marvin Minsky, "A framework for representing knowledge" in Artificial Intelligence Memo No. 306, Massachusetts Institute of Technology, A.I. Laboratory, June 1975.

[Per00] Fernando Pereira, "Formal grammar and information theory: together again?" in Philosophical Transactions of the Royal Society, April 2000.

[Rab89] Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition" in Proceedings of the IEEE, Volume 77, No. 2, February 1989.

[RJ93] Lawrence R. Rabiner and Biing-Hwang Juang, "Fundamentals of speech recognition", Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[Ros00a] Ronald Rosenfeld, "Two Decades of Statistical Language Modeling: Where do we go from here?" in Proceedings of the IEEE, Volume 88, Issue 8, August 2000.

[Ros00b] Ronald Rosenfeld, "Incorporating linguistic structure into statistical language models" in Philosophical Transactions of the Royal Society of London A 358, 2000.

[RR95] W. Reichl and G. Ruske, "Discriminative training for continuous speech recognition" in Proceedings of Eurospeech, Volume 1, pp. 537-740, 1995.

[SBVW06] Tanja Schultz, Alan W. Black, Stephan Vogel, and Monika Woszczyna, "Flexible Speech Translation Systems" in IEEE Transactions on Audio, Speech and Language Processing, Vol 14(2), March 2006.

[Sha48] Claude E. Shannon, "A mathematical theory of communication" in The Bell System Technical Journal, Volume 27, pp. 379-423, 623-656, July, October, 1948.

[SMF$^+$01] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel, "A One-pass Decoder based on Polymorphic Linguistic Context Assignment" in Proceedings of the ASRU, Madonna di Campiglio Trento, Italy, December 2001.

[Sto02] Andreas Stolcke, "SRILM – An extensible language modeling toolkit" in Proceedings of the ICSLP, Denver, Colorado, USA, 2002.

[SO04] Andreas Stolcke and Stephen M. Omohundro, "Best-first model merging for Hidden Markov Model induction" in TR-94-003, International Computer Science Institute, Berkeley, California, USA, April 1994.

[TSK02] Toshiyuki Takezawa, Fumiaki Sugaya, and Genichiro Kikui, "Using bilingual conversational expressions in Speech Translation" in Linguistics and Phonetics, Urayasu, Japan, 2002.

[VOW$^+$97] V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young, "Mmie training of large vocabulary recognition systems" in Speech Communication, pp. 303-314, September 1997.

[WA01] Ye-Yi Wang and Alex Acero, "Grammar Learning for Spoken Language Understanding" in IEEE Workshop on Automatic Speech Recognition and Understanding, Madonna di Campiglio, Italy, 2001.

[WA02] Ye-Yi Wang and Alex Acero, "Evaluation of Spoken Language Grammar Learning in ATIS Domain" in Proceedings of ICASSP, Orlando, Florida, USA, 2002.

[WA03a] Ye-Yi Wang and Alex Acero, "Concept Acquisition in Example-Based Grammar Authoring" in Proceedings of ICASSP, Hong Kong, China, 2003.

[WA03b] Ye-Yi Wang and Alex Acero, "Combination of CFG and N-gram Modeling in Semantic Grammar Learning" in Proceedings of the Eurospeech 2003, Geneva, Switzerland, 2003.

[WA03c] Ye-Yi Wang, Alex Acero and Ciprian Chelba, "Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy" in Proceedings of the ASRU, St. Thomas, US Virgin Islands, 2003.

[WM01] Chin-Chung Wong and Helen Meng, "Improvements on a semi-automatic grammar induction framework" in Proceedings of the ASRU, Madonna di Campiglio, Italy, 2001.

[WTSW02] Zhirong Wang, Umut Topkara, Tanja Schultz, and Alex Waibel, "Towards Universal Speech Recognition" in Proceeedings of the ICMI, Pittsburgh, PA, USA, 2002.