Spoken Language Understanding without Transcriptions in a Call Center Scenario

Submitted to the

Technische Fakultät der
Universität Erlangen–Nürnberg

in partial fulfillment of the requirements for
the degree of

DOKTOR–INGENIEUR

of

Michael Levit

Erlangen — 2004

As dissertation accepted by the
Technische Fakultät der
Universität Erlangen–Nürnberg

Date of submission:	October 18th, 2004
Date of doctorate:	
Dean:	Prof. Dr. rer. nat. A. Winnacker
Reviewer:	Prof. Dr.-Ing. H. Niemann
	Dr. A. Gorin
	Prof. Dr.-Ing. J. Hornegger

Acknowledgments

I would like to express my gratitude to my mentors and colleagues without whom this work wouldn't have been possible.

First and foremost, I would like to thank Professor Heinrich Niemann, the head of the Chair for Pattern Recognition at the University of Erlangen, for teaching me the basics of the field and waking my interest in it, but also for giving me the opportunity to earn a doctorate at his chair.

I owe special thanks to Dr. Elmar Nöth, head of the Speech Processing Group at the Chair for Pattern Recognition, whose sometimes critical and yet unfailingly fair opinion guided my progress throughout these years. It is Elmar from whom I have the strong conviction that science is not only about creativity and innovation but is also a great deal of fun. The role of Elmar's benevolent participation in my career is really hard to underestimate.

Among people who contributed the most to this work, I would like to particularly distinguish Dr. Allen L. Gorin. It was my honor and privilege to work in his team at the AT&T Labs. While being leader of the HMIHY project and later as Director of the Knowledge Discovery Department, Al's professional expertise in the scientific matters and his genuine support of a friend in everyday situations remained a steady and irreplaceable source of confidence for me that I will always appreciate.

For countless and very informative discussions that gradually shaped this dissertation I would also like to pay tribute to my former colleagues from the AT&T Labs and the University of Erlangen, and in particular to Jerry Wright, Patrick Haffner, Hiyan Alshawi, John Grothendieck, Gökhan and Dilek Tur, Vincent Goffin, Viktor Zeissler and many many others. All of them showed keen interest in my work and I am deeply grateful to them for sharing with me their vision of the field.

Abstract

This dissertation addresses the possibilities of Spoken Language Understanding (SLU) in a call center scenario. It is widely agreed upon that human understanding involves complicated cognitive structures the machine replication of which is not tangible today. However, many practically important applications in the automated SLU don't have to rely on computer cognition, while adopting the "behaviorist" approach to understanding instead. This approach states that understanding is present wherever the action the machine takes upon receiving an input message, is perceived as intuitively correct. This action can be formally encoded in terms of its *semantic function* (that determines a rough category of the action) and *semantic attributes*, parameters that this function takes to become well-defined. Thus, the goal of the understanding becomes to extract these elements from the input signals wherever possible. So, calltype COLLECT_CALL is semantic function and named entity *12345* is its parameter in the utterance *"I'd like to make a collect call to number 12345"* taken from one call center scenario. In general, calltypes as semantic functions and named entities as semantic attributes are characteristic for many call center applications; in this work we show how spoken utterances can be handled with respect to these information types. We extract calltypes and consider three categories of named entity processing tasks: detection, localization and value extraction of named entities.

One distinctiveness of our experiments consists in not relying on the availability of manually created word-level annotations for training corpora from the target domain. To retain acceptable word accuracy in the ASR-output, we suggest using the mechanism of unsupervised language model adaptation. Similarly, we avoid the need to manually annotate instances of named entities in the training data by using generic application-independent grammars for their modeling. For those hard cases where not even an off-the-shelf language model is available to bootstrap the speech recognizer, we show how our algorithms can be ported onto the phone level.

In the context of the last task, we also discuss the academic problem of word lexicon extraction from a continuous phone stream. We use special semantic and syntactic qualities of words, to infer phone subsequences that replicate them.

All experiments we report on in this thesis were conducted on the *"How May I Help You?"* speech corpus of over-the-phone interactions of AT&T customers with the company's partly automated call center.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 How Difficult is Automatic Spoken Language Understanding?

It is a fantastic, tempting vision that has been planted in our minds during the past five decades: computers of the future will be able to do everything. Today's science fiction writers already foresee the day when huge spaceships have the abilities to dodge hefty meteorite showers in a matter of milliseconds and to automatically exploit space-time singularities to beam themselves in an arbitrary point of the universe. The Hollywood movies are jammed with gross constructions of motors, springs and light diodes looking like real people and even acting in a way natural to humans. Even penetrating our minds and dwell there without our permission, appears to be anything but impossible for a self-aware computer community of the late twenty first century. While some of those visions may be less unrealistic than the others, they all have at least one point in common: the aspect of communication. The computers of the future interact; among themselves and also with us, people. In the latter case, they adopt our natural communication endowment and talk. And when we talk to them, they listen. And understand. Faultless communication is certainly a crucial pre-condition for allowing computers of the posterior to do all these mind-boggling tricks, they obviously will be capable of. In fact, what's the point of authorizing the on-board computer with the steering powers, if each time the captain orders an urgent direction change, he risks to get baffled with something like: *"Sorry Captain, I didn't understand your order. Please repeat it slowly in a calm voice. I recommend that you use simple sentences and make sure that other crew members remain silent while you are speaking"*. Even though everybody silently assumes that situations like this will never happen in the years to come, reliable voice interaction with computers in the future should not be taken for granted, but rather considered a challenge for generations of research. The one reason being that, given the technology of today, making computers use language like we do, is a dream as distant as flying to galaxies far, far away. Or is it?

Even though we are painfully aware of the fact that the automatic speech processing is not yet capable of replicating human ear and mouth [Gol00], it is worthwhile to differentiate its subtopics according to how close to a successful solution they are. First of all, a decision on the nomenclature must be made and explanation given to what we understand under successful solution of a task in the research field of automatic speech processing. The common assumption followed in virtually all speech processing literature is that the performance of an average native language speaker can be taken as a standard the automated systems should aim at. A task is thus said to be solved successfully by the system, if humans, when presented with the same task, achieve results that are not significantly better than the ones this system produced. The scope of possible speech processing tasks is very wide. It includes speech synthesis, embraces a wide range of recognition topics stretching from isolated digit recognition, for which a first solution was proposed as early as in the 50ies [Dav52], to the difficult task of spoken language recognition under hard acoustic conditions. During the past decade, spoken language understanding (SLU) started drawing a lot of attention, acting as a cornerstone of numerous academic and practical problems. Applications like knowledge retrieval, call center or machine translation, when provided with audio interface, have SLU as an indispensable part in their algorithmic endowment. While speech synthesis research already delivered a number of mature commercial products with the speech quality comparable to the one of humans [Beu99, Rut00] (see also a comparison in [VA02]), the performance of automatic speech recognition systems (ASR) still remains an order of magnitude behind what the humans can do even on the simplest tasks of isolated digit recognition [Lip97].

This discrepancy is yet much more striking in the field of automatic language and speech understanding, where the definition itself of what it means for a machine to understand speech is still a subject for taxonomic discussions [Fod75, Win80, Gor95]. Some authors even expressed doubts if computer understanding of natural language is tractable at all [Dre92]. Being traditionally thought as a processing unit downstream from the speech recognition stage, speech understanding is also bound to suffer from the shortcomings of the latter, especially since it is still unclear if the improvement of recognition rates over time obeys the promising Moore's law, like many other performance criteria in computer science [Chu03]. Nonetheless, some authors predict that over the next decade several breakthroughs in the speech understanding technology can be expected, including continuous speech understanding with standard dictionaries [Ber00]. In fact, there is already a number of developed understanding systems that specialize on conversations with tightly restricted subset of practical goals. Thus, narrowing the domain of discourse to allow for only a finite (usually rather low) number of permitted dialog objectives, made it possible to take a successive approach to the concept of understanding and start with its rudimentary approximations such as topic classification. Certainly, by peeling down the targeted application field in this way, machine understanding is denied the deep psychological groundings of the understanding as we used to think about it with regard to human beings, however it turns out that often even this approximation can fulfill many of our practical needs.

There are several factors that stem from the comparative hardness of the spoken language recognition task and single out spoken language understanding as one of the most challenging tasks in the language understanding domain:

- Compared to the understanding of written texts, SLU is much more difficult because of the imperfectness of the speech recognizer it is based upon. Instead of *"Hi, I'm John Stone. I got my last AT&T bill for 50$ on October 16th"* the word recognizer might spit out something like: *"hi I'm John stone I hot last of t and t bill oh fifteen dollars and on October sixteen"*. In this example several types of errors occurred: some words were misrecognized (*got* became *hot*, *my* was deleted and *and* inserted), capitalization was disregarded in a proper name (*stone* instead of *Stone*), abbreviations and notations adherent to special conventions lost their consistency after being spelled out (*AT&T* and *50$*). There are many factors that lead to misrecognition errors: such as not matching acoustic conditions (noise, echo etc.), not suitable language models, speech variations (dialects, sociolects and argots) and others [Fur00, Hua01]. In any case, all of the recognition artifacts above have to be allowed for and dealt with when trying to recover the meaning of the spoken message after it has been recognized by an ASR-system.

- Even if the word recognizer were perfect, there would be another complication concerning natural (spontaneous) speech. Natural spoken language can hardly be expressed in terms of the mandatory grammars describing the polished "correct" language. Since spoken language is closely related to the thinking processes and the later are often considered nondeterministic for their complexity, it is only a logical consequence that the language we speak is rich on discontinuities in the speech flow, which include such phenomena as word and phrase breaks, fillers, repairs, repetitions etc. [Fur03]. Besides, we can speak continuously and yet use erroneous syntax (*"He they decided to go"* or *"he do a nice job"*), because we might have changed our intentions in the middle of the utterance, had a slip of tongue or used an alternative (e.g. sociolectal) language form. In any case the understanding mechanism will encounter more difficulties parsing this "unclean" language version than a written, well-rounded one.

- Finally, SLU not only has to struggle with the speech discontinuities described above, it also has to put up with the lack of other useful information available in the written text. Apart from the issues afore mentioned (like for instance spelled out abbreviations), a further interesting example is punctuation, an important part of the syntactic parsing processes [Hab94] which helps to disambiguate otherwise ambiguous utterances. One well known compensation for this loss is using prosody which some authors consider as the origin and foundation for punctuation [Sch90]. However, prosodic analysis is not a part of recognition process, but rather augments it.

1.2 Overview of Existing Spoken Language Understanding Systems

Despite all these difficulties, there has been a large body of research on spoken language understanding. In the following section we will briefly describe some of the most popular practical approaches, arranging them roughly in two categories, linguistic or statistic, according to the predominant motivational principle behind them.

Linguistically Guided SLU-systems

Starting in the late seventies the Chair for Pattern Recognition at University Erlangen-Nuremberg and its industrial and academic partners invested a lot of research effort in developing of an automated natural language dialog system [Hei79]. After a number of intermediate successful results, the EVAR[1]-system was launched in 1993 as the worldwide first information system for public use over telephone line [Gal98]. The goal of EVAR was to provide on-demand information about InterCity railway connections between German cities.

The understanding component of EVAR performs linguistic analysis of the recognized utterances based on the knowledge representation with *Unification Categorial Grammars (UCG)* [Zee88]. Unification combines feature structures (*signs*) that reflect morphologic, syntactic and

[1]EVAR is a German abbreviation for: *Recognize, Understand, Answer, Ask back (Erkennen, Verstehen, Antworten, Rückfragen)*

semantic characteristics of the categories they encode, in a new more specific structure. Given two argument feature structures, unification produces the most general feature structure that is conforming to the descriptions of both arguments. One example of unification is given below:

$$\begin{bmatrix} cat: & NP \\ num: & singular \end{bmatrix} \sqcup \begin{bmatrix} cat: & NP \\ gen: & masculine \end{bmatrix} = \begin{bmatrix} cat: & NP \\ num: & singular \\ gen: & masculine \end{bmatrix} \tag{1.1}$$

The grammar employed in EVAR utilizes unification mechanism by modifying it in such way that contiguous neighbor word sequences can be merged if they are compliant to each other. To select optimal linguistic representation of an utterance, robust *island parsing* is employed.

Another example of a spoken language understanding system which employs linguistical knowledge to perform complete linguistic analysis of the utterances is the GEMINI by SRI [Dow93]. This system uses domain-independent typed unification grammar for English to extract constituents based on a domain-specific lexicon defining word base forms, morphological rules and such. One peculiarity of this system is its ability to detect and correct disfluencies in the spoken speech such as repairs. This is carried out by means of a semantic fallback where no acceptable semantic interpretation of the complete utterance is possible [Dow93].

Other examples of linguistically guided automated understanding systems include original PHOENIX system of CMU [War91] and MIT's TINA [Sen92], where probabilities are introduced in the context-free rules of the linguistic constraints.

Statistically Motivated SLU-analysis

Interpretation of meaning as a combination of several application-dependent semantic units, has been a popular approach taken on by many research groups. Some 15 years ago, ARPA Spoken Language Systems community first presented the ATIS (Air Travel Information Service) task [Pri90]. The anticipated goal was to motivate participants to build natural language understanding systems capable of answering users' queries about numerous aspects of airline trips, such as source and destination cities, flight fares, served meals and many others. Several research groups answered the call by designing new (or tuning their existent) SLU-systems to the ATIS-benchmark.

The CHRONUS system by AT&T [Pie92, Pie95] was one of the participants in the contest. Its understanding module relied on the representation of the sentence meaning as a sequence of *semantic concepts* (basic units of application-dependent meaning, like origin of a flight, destination, meal etc.) expressed as attribute-value pairs. The semantic decoding in CHRONUS presupposed a sequential correspondence between semantic and acoustic units, so that each semantic concept from the conceptual segmentation had to be accounted for by a contiguous acoustic segment. This allowed for a simple stochastic modeling with the sequence of semantic concepts and word emission probabilities within each concept, both governed by n-gram stochastic processes. If we denote the observed acoustic signal as A, the recognized word sequence as $W = w_1 \dots w_{T_w}$ and the corresponding concept sequence as $C = c_1 \dots c_{T_c}$, then the formal decoding rule is to find such sequence C^* (and, in the case of integrated speech recognition, also W^*) that would maximize the joint posterior probability:

$$W^*, C^* = \underset{W,C}{\operatorname{argmax}} P(W, C|A) = \underset{W,C}{\operatorname{argmax}} P(A|W)P(W|C)P(C) \tag{1.2}$$

with "semantic" and "syntactic" probabilities $P(C)$ and $P(W|C)$ approximated as first order Markov chains:

$$\begin{aligned} P(W|C) &\approx P(w_1) \prod_{t=2} P(w_t|w_{t-1}, C) && (1.3) \\ P(C) &\approx P(c_1) \prod_{t=2} P(c_t|c_{t-1}). && (1.4) \end{aligned}$$

Additionally, to increase the robustness of the concept-dependent language models, non-content words are excluded from them and content words are aggregated in the so-called *superwords* (e.g. *"san-francisco-international-airport"*) by a lexical parser [Pie95].

The concepts in the resulting segmentation are provided with values extracted from the corresponding word sequences by means of a *template generator* and assembled in one structure representing the meaning of the utterance. This structure is then used by the dialog manager to continue the dialog.

There is also a statistic analysis opportunity in EVAR. As a domain-dependent alternative for the generic deep linguistic analysis in EVAR, the shallow linguistic analysis was suggested and implemented by Haas [Haa00]. As in CHRONUS, the meaning of an utterance here was also reduced to a sequence of one or several attribute-value pairs, however no assumptions about contiguity of words representing one semantic concept was made; besides, the requirement that each word had to belong to at least one meaning-bearing semantic concept was dropped as well. As a result, localization and extraction of attributes and their values was guided by a probabilistic *assignment model* which led to a description formalism for the word generation process similar to Hidden Markov Models of higher statistical orders [Haa00, page 85].

Among other projects rooting in the stochastic models of sentence meaning, we would like to mention the understanding approach by IBM borrowed from the field of *automated translation* where natural language sentences are translated into a special formal language, whose semantics can be interpreted straightforwardly [Eps96], and a deployed AT&T system HMIHY [Gor97], which will be explained in detail and referred to repeatedly throughout this thesis.

Summarizing, we can say that today there are two fundamental directions for the evolution of spoken language understanding systems: the one following the rationalistic view of a language and leaning on a framework of meticulously elaborated linguistic rules, and the other sticking to the empirical model of statistical rule learning. There is also a growing number of tentative hybrid approaches integrating elements of statistics in the spoken language understanding based on linguistic rules.

1.3 Field of Application

In psycholinguistics, to understand the meaning of a sentence means to infer its underlying propositional structure and to correlate it with the pragmatic context of the entire interaction and with the world knowledge of the listener (see also discussion in [Ger90, Rie94]). Does this definition require a computer to maintain a cognitive model of the topic that can be disentangled from a particular realization or transfered to different applications? Maybe if we want a conversation partner, but for practical purposes all we aspire are computerized systems that are capable of serving our needs, or speaking more technically, providing adequate reactions to human requests expressed as spoken utterances. For that matter, we can look at an automatic understanding system as a black box which, provided with input stimuli, *somehow* produces correct output action. A simple *behavioristic model* where machine is *trained* to deliver an appropriate action for each stimulus, is absolutely sufficient in this case. The purpose of such training is to establish and reinforce expected connections between possible inputs and supported outputs of the systems. There are many ways of how these connections can be realized in practice. One example are *connectionist networks* [And89], an instrument for stochastic modeling that has been successfully employed in several information retrieval experiments [Gor94b, Mil93].

A similar notion of *operational understanding* can be found in the literature. If the goal of communication is to effect a certain kind of correspondence between the (mental) states of the speaker and the hearer [Fod75], then the operational interpretation of understanding is an instance or act of a desired change in the state of the recipient [Gor91]. In the case of machines, this change is accomplished by the actions they take upon receiving a request, whereby the space of actions is defined by the specific requirements of the application and can include primitive actions like saying "hello", or giving a beep as well as rather elaborate combinations thereof, like finding a red ball on the table, grabbing it and putting it on the top of a red cube.

The research field this thesis is located in, relates to the class of practical tasks that can be solved based on this behavioristic model of understanding which we formalize in the following definition:

Definition 1.1 (Understanding) Understanding *means extracting application-relevant information from one or several input channels and organizing (interpreting) it in a way that comprehensively defines an appropriate (re)action of the system.*

The role of contextual and world knowledge for understanding is not explicitely stated in this definition even though it is tacitly assumed within the interpretation stage. Moreover, the emphasis of this work is on the methods of information retrieval, so that in the context of this thesis, we will be using the general assumption of a trivial interpretation procedure with minimal influence of context, and thus refer to the task of extraction of the relevant information from the input channels as the *understanding task* as well.

In a typical scenario of the person-to-machine communication, the user initiates a dialog with the goal of receiving some information or getting some operation done. In any case, upon receiving the request, the computer is expected to execute one or several actions, while the entire space of the supported actions can be split in a number of subsets, delimited by the action's type. As an example, consider the *"How May I Help You?"* task [Gor97]. This is a typical *call center scenario* with non-expert users calling in and making various service requests over the phone in form of natural language utterances. The automated dialog system developed at the AT&T Laboratories was designed to infer appropriate machine actions from these requests that were elicited by an open-end prompt *"How May I Help You?"*. Suppose, one caller said: *"Hi, I don't recognize several numbers on my bill"*; then, the type of the action that is expected from the system is CLARIFY_UNRECOGNIZED_NUMBER. If, however, the request was: *"I would like to make a credit card call"*, then it should lead to an action of the type CREDITCARD_CALL. Finding out an appropriate type of action based on customers' requests submitted over the telephone lines is usually referred to as the *calltype classification task.*

Most of the time knowing only the type of the expected action the request is supposed to elicit, will not be sufficient to complete it. In the credit card example above, the system also has to know where the call must be going to. Of course, we can define a separate action type for collect calls to each valid phone number, but this will blow up the size of the task, and make it totally intractable in cases where proper names or geographic locations are part of the request. This is why the more elegant solution is to maintain a separate space of additional information units which will account for the choice of the object or other auxiliary information. These information units are generally called *semantic attributes*. The action type extracted via calltype classification will be augmented by the appropriate semantic attributes and thus form a ready-to-process instruction, so that one could say that the system *understood* the request.

If now caller's request is *"Hi, I would like to make a credit card call to Georgia, my card number is 1234567890"*, it will be represented as:

- *action type:* CREDITCARD_CALL
- *destination:* Georgia
- *cardnumber:* 1234567890.

One special kind of semantic attributes especially important for the call center scenario are *named entities*. Named entities have been originally introduced as a placeholder for proper names in the language [NIS], but their definition was then expanded to embrace numeric expressions and time information as well [Chi97]. The definition for named entity that we use in this work is given below:

Definition 1.2 (Named entity) *In the context of this work, the notation* named entity *is used to refer to semantic attributes with large definition domains.*

In the example above *"Georgia"* is a named entity and so is the card number *"1234567890"*. Here are some other examples of named entities: *"AT&T Corporation"*, *"quarter to six"* (time expression), *"area code 123 number 4567890"* (US phone number).

In the *spoken language understanding task*, requests to machines are communicated by means of spontaneous, naturally spoken utterances representing an additional difficulty for the understanding. In fact, the tight connection between the quality of speech recognition and the understanding rates (rates of correctly inferred actions) makes the role of a good recognizer at hand crucial for the practicality of each spoken language understanding system. The contribution of language modeling, i.e. modeling of statistical dependencies among recognition units (e.g. words) in the language, to ASR- and, through it, to SLU-performance, is considered particularly important in the framework of this thesis.

It is a well known fact that language models vary strongly depending on the application domain. The probability of the word *"stocks"* in the Wall Street Journal corpus is much higher than the probability of the same word in the ATIS database. Thus, for each practical application it is important to have a language model optimized for this application. Usually, producing an accurate language model, is preceded by manual transliteration of large quantities of dialog samples that pertain to the domain in question, so that the estimation of the language model can be conducted on the resultant noise-free transcriptions. However, obtaining these transcriptions is costly and time consuming. This is why less expensive techniques have become increasingly popular in the recent years. One of them suggests estimating a general language model (of English) and adjusting it appropriately for whichever domain is considered, with a minimal amount of manually transcribed data collected from this very domain. This approach is called *language model adaptation* and has been addressed in several publications [Ros95, Woo98, Ljo00]. However, one could go even further and attempt the *unsupervised* language model adaptation that relies only on a sufficient amount of audio data from the domain of interest. This topic will be one important issue that we decided to include in the agenda for this thesis.

1.4 Contribution of this Work

In this work we consider the understanding from the behavioristic positions as an intuitively comprehensible causal link between receiving input stimuli and consequential performing of an action.

Stemming from Definition 1.1, we formulate one possible model that can be used to approximate understanding in a restricted subset of practical applications involving person-to-machine communication. The understanding task that is stipulated by this model consists in extracting of the following two kinds of information from the input:

- type of the action this input is supposed to elicit. The boundaries between different action types are drawn according to the application needs and the intuition of the designer. This allows us to talk about these boundaries as defined in terms of differences in *meaning* or *semantics*, and to refer to the action types themselves as *semantic functions of the actions*. In the calltype classification task, semantic functions are realized by the calltypes;

- parameters that semantic functions must take to become well-defined; in particular, we focus on the named entities introduced in Definition 1.2 which represent a subclass of the semantic attributes.

For the task of calltype classification we will show how large margin classifiers, and in particular *support vector machines*, can be employed to attain high classification rates when using word n-grams from the ASR-output and system prompt information as classification features. As far as processing of named entities is concerned, we will suggest a categorization of the relevant applications in a number of subtasks and present solution proposals for each of them. It will be demonstrated how named entities can be successfully detected in the ASR-output using that same large margin classifiers, and how the exact positions and meaning of all of their instances can be further determined by means of a maximum-likelihood parsing with generic handcrafted named entity grammars encoded as finite state machines.

We also will demonstrate that our algorithms remain practical even when employing unsupervised language model adaptation instead of using transcription-based language model in the recognizer. For the task of named entity localization this practicality will be guaranteed by the so-called “approximate matching” that allows error-tolerant instantiation of the named entities in the ASR-output.

Additionally, we will explore the special situation with no access to word-level transcriptions of the language samples whatsoever, and show that most of our algorithms can be successfully ported to the output of a phone recognizer. Furthermore, based on the definition of words through their semantic and syntactic properties, we will explore the possibility of the linguistically unsupervised extraction of word-like units from spoken utterances represented as continuous phone streams.

1.5 Organization of the Next Chapters

The contents of the following chapters are organized as follows: in Chapter 2 we present an overview of contemporary ideas for automatic language understanding; we will define and justify two understanding tasks: calltype classification and semantic attribute processing (in particular: named entity processing) which are typical for many practical problems of automated spoken language understanding. These tasks will be dealt with exhaustively in the subsequent Chapters 3 and 4 respectively. The extent to which the inferencing of word-like units is possible from continuous speech signal paired with primitive semantics will be studied in Chapter 5. A series of experiments conducted for each of the major topics addressed in the theoretical part of this thesis, will be described and analyzed in Chapter 6. This dissertation will be concluded by an outline of potential directions for future research in Chapter 7 and summarization of the most important ideas and results of this work in Chapter 8.

Chapter 2

Ontological Basis for Spoken Language Understanding in a Call Center Scenario

2.1 What is Language Understanding

It is unquestionable that language holds a tremendously important position in the human society. Most communication is being facilitated by its means, in fact, many researchers distinguish it as a central aspect of human intelligence [Who56, Wit68, Joh87]. As all communication channels, language implicates sender and receiver ends. The machinery of language would be totally useless if people were endowed to produce speech signals without being able to listen to them and understand the meaning encoded in the acoustic sound waves (a similar statement certainly holds for the written language as well). Thus, the success of language as a communication instrument is largely due to our ability to retrieve useful information from the oscillating air pressure, decode it into mental representations of forms and actions and fit them into the framework of our expectations, based on the shared world knowledge.

How humans accomplish this task has been the subject for numerous physiological, linguistic, psycholinguistic, psychological and philosophical studies. As far as the latter three science fields are concerned, the majority of scientists today believe that understanding is best described in terms of the underlying cognitive concepts in the mind of the hearer [Ger90, Rie94]. The nature of these concepts is extremely complicated and still not well understood; yet it is clear that if we were to model human understanding properly, and tried to recreate the conceptual thinking and understanding in machines, we would have to resort to rather crude approximations. James Allen writes: "*The present state of knowledge about natural language processing is so preliminary that attempting to build a cognitively correct model is not feasible*" [All95, page 3].

2.1.1 Behaviorist View of Language Understanding

The question is, however: do we need cognitively correct models? And, if not, how can we carry out the transfer of language understanding mechanisms by humans onto computers then? The solution is to realize that we don't really care what is going on inside the machine as long as it does exactly what it's been told. If we ask a robot to move one particular object and it executes the action, we are satisfied even though the robot hasn't acquired a cognitive representation of the object or space or time. If a person feels like talking to a psychiatrist, and ELIZA [Wei66] could pass the Turing test [Tur92] and deliver a realistic impression of one, why not (putting aside the ethic and moral issues for a minute) deploy it just because it lacks the ethereal human cognition?

In fact, even if we were determined to create an automatic system which had an understanding machinery as much human-like as possible, we still would have to assess its understanding success by asking questions, making requests and judging the appropriateness of machine responses and actions it takes upon these requests.

John Bennett formulated this idea (also for human interactions) in the following behavioristic assertion [Ben67]:

> *"Unless the recipient makes a response which demonstrates his grasp – or lack of it – of the author's intention, the latter has no means of telling whether his intention has been put over".*

And, as we noticed before, the correct response is just what we are trying to achieve in the case of person-to-machine communication. In the computer science literature considering the appropriateness of the machine actions as a measure of its understanding defines *"operational understanding"* [Bob68, Gor91]. Gorin et al. formalize this term in the following definition [Gor91]:

> *"For any particular task, the goal is to map input messages into meaningful action. The set of all possible input messages we call language, and the mapping we call understanding".*

This definition can be easily formalized into a *functional model of understanding*, where the input message and the taken action are tied together by a functional dependency:

$$\text{action} = \psi(\text{message}). \tag{2.1}$$

In reality, people don't base their actions solely on the preceding information input. An enormous role in the process of decision making and action taking plays the knowledge of the person about the situation, such as his world knowledge or the information obtained during previous dialog turns. Subsuming all these factors under the notion of "stimuli" we can derive a more accurate and general dependency that will also be in line with our remarks to Definition 1.1:

$$\text{action} = \psi(\text{stimuli}). \tag{2.2}$$

Strictly speaking, we can only call a relation between stimuli and action a function if there is no such stimuli that can cause an ambiguous reaction. Contrarily to the initial impulse of denying this property to the systems operating on the language input, we argue that the ambiguity of the language doesn't imply ambiguity of stipulated actions. It is certainly not difficult to come up with a sentence that can be interpreted in several alternative ways (consider, for instance, this rather famous example: *"visiting relatives can be annoying"* with the inherent ambiguity of the subject). However, given a sentence like this, the system should be able to identify the problem in the input and initiate its disambiguation. Thus, the (uniquely defined) action taken by the system is an attempt to resolve the conflict, or at least to indicate that this conflict occurred (alternatively, contextual knowledge can also be used for disambiguation). Hence, the relation is a function indeed. Moreover, in case of natural language, the function is also surjective by construction (for each supported action there is at least one input that leads to its execution).

As the nature and complexity of supported actions may vary dramatically even within one system, the image of the function becomes very complicated and hard to define in one succinct formula. In the call routing task, for instance, it can include the actions as diverse as transferring the call to an appropriate destination and taking over initiative to obtain the missing bits of information. This is why it is easier to allow for a set of several application-dependent functions to perform mapping from the space of stimuli onto the space of actions:

$$\begin{cases} \text{stimuli} & \mapsto & \psi(\cdot), \quad \text{where } \psi(\cdot) \in \{\psi^1(\cdot), \psi^2(\cdot), \ldots, \psi^M(\cdot)\}; \\ \text{action} & = & \psi(\text{stimuli}) \end{cases} \tag{2.3}$$

Each of these functions is uniquely identified by a specific action it leads to. In fact, one can think of the set of these functions as a set of distinct semantic classes into which all actions supported by the system can be organized. In Section1.4 we chose to call them *semantic functions of the actions*. Depending on the application, the actions (and their semantic functions) can be very complex and possess rich descriptive or procedural semantics. For example, in robotics applications, like those described in [Win73] and [Roy03], one action can comprise some others like weighing several objects before ”moving the heaviest” when demanded so by the operator. The result is a hierarchy of semantic functions, in which arbitrary complex relations can be established between explicitly supported action types and also new action types can be assembled at runtime.

In Ludwig Wittgenstein’s main work *“Tractatus Logico- Philosophicus”* the idea of the language as a reflection of the “real world” was suggested [Wit21]. This conception of language views each proposition to be a faithful picture of reality, whereby complex propositions are assembled from the simple ones in a way similar to how complex circumstances of reality consist of the primitive facts. For instance, the sentence *“*[*pick up the red cube*] *and* [*give it to me*]*”* corresponds to a succession of two simpler actions, the first being picking up the red cube, and the second handing it out to the operator. In the framework of functional understanding, (and especially in the master-slave scenario, where the system has to obey users’ orders), this purported connection stipulates that the hierarchy of semantic functions of the supported machine actions must induce a meaning-based hierarchy on the space of the natural language sentences potentially interpretable by the system. Thus, we can group these sentences (or their corresponding spoken utterances) in what we call *semantic categories* in such a way that all utterances from one semantic category would lead to actions that share one semantic function. In other words, there will be a simple correspondence between semantic categories of the spoken utterances and semantic functions of the actions these utterances are supposed to elicit. Except for the ambiguity resolution case mentioned above, this correspondence allows us to use both characteristics synonymically, so that found semantic category of an utterance means a successfully retrieved semantic function of the action it must lead to.

Like semantic concepts (e.g. in CHRONUS) semantic categories are also induced by meaning; but in their case, this meaning must relate to an entire utterance as opposed to its individual constituents. What happens, however, if the semantics of the utterance is rather elaborated, e.g. *“I want to pay my bill and then make a collect call”*? Here, we can either maintain a large set of very specific semantic categories (like PAY_BILL-AND-COLLECT_CALL) or allow for several general semantic categories for one utterance at the same time. Essentially, both methods are equivalent, but in practice, dealing with several alternative semantic categories in parallel is more convenient than handling complex conjunctions thereof. With this remark in mind, for the sake of simplicity, we will keep talking of yet *the* semantic category of an utterance and *the* semantic function of the intended action.

With the increasing degree of complexity of function hierarchy (and, thus also, with the increasingly complex space of semantic categories of the corresponding requests), more elaborate

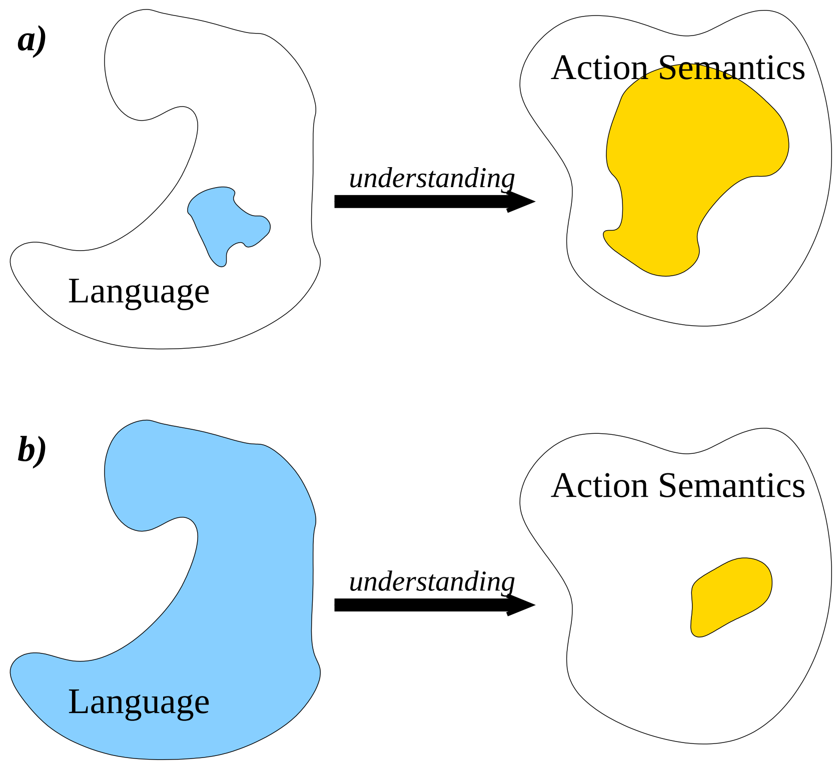


Figure 2.1: Two alternative approaches to SLU-system design: a) understanding of elaborate semantics is only feasible with rigorous language restrictions b) spontaneous natural language necessitates a limited scope of action semantics; after [Gor91].

syntactic and semantic parsing processes will be required. When dealing with natural language, these processes, among other things, will have to handle various contextual language phenomena like, for instance, ellipses. Also, because of the fragility of fine semantic structures in the natural language, the system would become less robust against such spontaneous speech effects as self-corrections and repetitions. Finally, when accomplishing the transition to spoken language as the operating medium, another weak point becomes evident: danger of recognition errors. In general, the designer of an SLU-system should keep in mind that there is always a compromise between its modeling potential and the limits of its practical application. One could try to account for very complicated semantic structures but – for practical usability reasons – would have to bind the speaker to grammatically correct utterance formulations only (and would also need an excellent ASR); the other alternative would be to allow for unconstrained speech and support only limited complexity of action semantics (see Figure 2.1).

2.1.2 Understanding = Semantic Function + Parameters

How to establish a trade-off between the complexity of the hierarchy of semantic functions of the actions (or simply: functions) and the feasible degree of allowed "naturalness" of the language in each particular case? Usually the application domain itself gives us a clue as to how to set priorities. In robotics applications, for example, it is desirable to allow the operator to express composed commands with pronominal references in one sentence: *"Find the blue pyramid and put it on top of the large cube"*, whereby also the assumption of the cooperativeness of the operator seems reasonable. In the applications like call center, however, one has to deal with customers who sometimes are not even aware that they are talking to machines, but whose requests can be generally grouped in a moderate number of semantic categories.

The emphasis of this dissertation is on spoken language understanding, this is why we have chosen the second view on the automatic language understanding and conducted our experiments in a call center scenario. The space of semantic functions in this case degrades to a countable and even finite set, chosen so as to represent different request topics.

Even though we emphatically do not claim that this approach is capable of understanding semantically complex utterances, it should be noticed that, embedded in a dialog scenario, it can solve also semantically complex problems. On the other hand, it is clear that even if semantic categories of two requests are the same, the mechanical actions taken by the system may still be different. Imagine that our system is supposed to answer customer questions about products sold in a store. Two sentences:

1. How much is *the red Nike jacket, size XXL*?

2. What are the costs of *a kitchen knife set*?

– essentially belong to the same semantic category (price inquiry) but refer to different objects. Furthermore, the number of possible objects can be very high. How is it possible then that with a small number of semantic categories the systems like this are capable of handling so many different requests? Apparently, the knowledge of the semantic category alone is here not enough to complete the transaction, but some additional information is required. This additional information can be easily incorporated in the functional model of understanding (2.3) through an explicit specification of the parameters that functions $\psi^m(\cdot)$ take:

$$\begin{cases} \text{stimuli} & \mapsto & (\ \psi(\cdot)\ ,\ \boldsymbol{\nu}\ ), \quad \text{where } \psi(\cdot) \in \{\psi^1(\cdot), \psi^2(\cdot), \ldots, \psi^M(\cdot)\} \\ \text{action} & = & \psi(\ \boldsymbol{\nu}\ ) \qquad\qquad\qquad\qquad\qquad\qquad \text{and } \boldsymbol{\nu} \in \Lambda^{\psi}; \end{cases} \tag{2.4}$$

In this formula the role of stimuli is two-fold. First of all, they determine the semantics of the action by means of an appropriate function $\psi(\cdot)$. Along with the function choice goes the selection of the parameter domain $\mathit{\Lambda}^{\psi}$ this function is working on. Secondly, they provide actual values for the parameter vector $\boldsymbol{\nu} \in \mathit{\Lambda}^{\psi}$. In Section 1.4 we named these parameters *semantic attributes*.

Let us now come back to the example on page 9. Here, we can build the following formula based on information extracted from the request:

$$\begin{aligned} \text{action} = \textsc{CreditCard\_Call}(\quad & \text{parameters.destination=}\textit{Georgia}, \\ & \text{parameters.cardnumber=}\textit{1234567890}). \end{aligned}$$

In many of today's approaches to spoken language understanding (see Section 1.2) semantic function and its parameters are viewed both as semantic attributes and handled in the same way. The meaning of an utterance is thus the result of a composition of several attributes and their values. In our strategy we single out semantic function though. This is done for the reasons given at the end of Section 2.1.1, and also because of the argument that in the practical applications like call center (especially call center first-level support) the space of supported semantic functions is limited, so that simpler algorithms can be used for their identification. For instance, instead of localizing an instance of one semantic concept responsible for the action to take, we might conclude on this action by collecting many weak indicators from all over the sentence. We will see in Chapter 3 how this can be done. Therefore, in the rest of this thesis we will use notation "semantic attributes" (or simply "attributes") only with respect to function parameters, unless explicitely stated otherwise.

Often the boundary where the function ends and its parameters start is rather vague. For example, in the request above we might as well introduce the function "make a call" and view "credit card" as a parameter defining how this call should be made. The concrete decision about role separation is to be made in each case separately, depending on the specifics of the task. However, in many cases, like phone numbers or proper names the choice is obvious, and the information should be treated as parameters. In the message understanding community these kinds of parameters are frequently referred as *named entities*. While most authors define the named entities by listing the types of semantic attributes they consider to belong into this class, we provided their tentative formal definition in Section 1.4. Extraction of named entities from text and speech is another major topic in the natural language understanding [Chi97] which we will address later in this thesis.

2.2 Calltype Classification

One task that recently became very popular in the language understanding community is *topic classification*. Topic classification has the objective of assignment of one or several topic-labels from a pre-defined set to spoken or written documents based on their contents. Despite being a clear simplification of the most general understanding paradigm, topic classification was recognized as tremendously useful for practical applications. One of its most gainful realizations can be found in the *call center scenario*, where users call in a communication hub over telephone lines with the intention to solve some problems or obtain some information concerning one subject or a limited number thereof. Since using human resources to provide this kind of service is extremely expensive for the companies, the need for automatization is obvious. Fortunately, at least for the first-level support call center applications, the communication domain is limited and can be categorized straightforwardly, so that the use of topic classification technology is manifest. Using semantic categories $\{c^1, \ldots, c^M\}$ (that correspond to the semantic functions $\psi^1(\cdot), \ldots, \psi^M(\cdot)$ of the supported actions) as the searched-for topics in this scenario turns topic classification into *calltype classification*.

2.2.1 SLU-model in a Dialog Framework

In the modern autonomous systems developed for the call center scenario, *understanding* is much more than just semantic categorization of spoken utterances. In Section 2.1.2 we already pointed out the necessity to augment the semantic category of an utterance by auxiliary information, semantic attributes and their values. Here we focus in short on another understanding mechanism which is one of the reasons why deployment of automatic speech understanding systems in call centers became so successful: *dialog*.

When two people are talking to each other, they rarely share all the information they plan to exchange at once. Instead, the total amount of useful information is distributed over the turns of the dialog. Imagine one person asking a passer-by for directions to a store:

— *…Excuse me, I'm looking for a store...*
— *Yes? What kind of store?*
— *A clothing store.*
— *There are several in the mall, which one are you thinking of?*
— *It's called "Joe's Rags".*
— *Aha, sounds familiar… What do they sell?*

— *They sell night gowns.*
— *Right! You have to take the next right, then...*

The reason for spreading information over several turns is that the questioner wants to make sure the addressed person will understand him, and in order to achieve this he does two things:

- gradually prepares the other to acquire the (request) information he wants to convey;
- provides more and more specific details after getting confirmation that previously conveyed information was received and correctly interpreted.

Henry Gleitman formulated this circumstance in the following way: *"To speak to another, one has to have an idea... that the other lives in the same, mutually perceived, world"* [Gle91, page 360].

The practical experience indicates that when interacting with computers, people often apply the same principles of communication as when communicating with each other [Ree96]. In particular, the dialog communication paradigm transfers onto the person-to-machine communication, and sound analogies in how humans and machines use dialogs to exchange information, can be established.

Up until now we didn't distinguish among different kinds of information (we called them "stimuli") that affect human understanding. However, their nature can be very versatile and require processing by separate mental mechanisms. In speech understanding one usually distinguishes three groups of input data that affect spoken dialog: *local (audio) information*, *dialog context* and *world knowledge*. Let's consider them separately and see how they are dealt with by speech understanding endowment of humans and its computer replica.

- **Local (audio) information.** We have already seen that functional model of language understanding doesn't build upon cognition, but rather suggests an approximation of utterance understanding as extraction of semantic function of the intended action along with its parameters. All deep groundings in the real world (like, for instance, the meaning of "today" or "I") can be factored out here if a learnable correspondence between (annotated) speech and intended actions is presumed.

  This approximation also tacitly relies on the assumption that the semantic function and all its parameters *can* be extracted from the given utterance. In fact, in many real life situations, restricting the input sources to the local (audio) information will be adequate: everybody reacts more or less the same when exposed to a sudden scream: "Watch out!"

(probably something like turning around the head and compulsive ducking). The same holds for practical automatic understanding: if we are lucky, all the information needed to conduct the action expected by the user will be present in just one utterance. According to what we are searching for in a call center scenario, two processes will be initiated for each new caller utterance: calltype classification and attribute (e.g. named entity) extraction. Sometimes it is these very two tasks that the notation "spoken language understanding" is used for.

- **Dialog context.** What happens, however, if the useful information is spread over several turns as demonstrated in our example above? In general, people interpret each new sentence in the context of their situational knowledge. This knowledge gives rise to a framework of their expectations, each new bit of information must fit in [All95, pages 465–467]. Situational competence is supported by memory, and the latter can be effectively divided in two parts: *short term*- and *long term memory* [Gle91, page 247]. From the dialog point of view, short term memory is responsible for the person's ability to maintain orientation in the course of the dialog, connecting its different turns into one meaningful conversation. For instance, it is the short term memory that physiologically supports elliptical structures in the language. By the automated dialog-based person-to-machine transactions, this mechanism is modeled by *dialog states* which define what relevant information has already been provided, what information is still missing and how to interpret the information to come based on the data already received. In a successful dialog, it is possible to close the gap between one semantic function and its parameters that define the intended action on the one side, and semantic categories and semantic attributes extracted from each individual utterance of this dialog on the other hand. For example, if the system prompted the caller for his home phone number and in the next utterance he provided a sequence of digits, there should be no doubts whether this sequence is the phone number of the caller or the amount he is trying to get a billing credit (credit for a misplaced call) for.

- **World knowledge.** Long term memory in humans is the reason why we can achieve most of our communication goals at all. In the dialog above, which is similar in spirit to the call center scenario conversations from our agenda, the fact that the hearer can relate the description *"clothing store that sells nightgowns"* to the shop he had actually seen or stopped by before, is because he has got a commemoration of the facility, stored in his long term memory. In the same manner, he knows that correct response to the *"I'm looking for…"*-question is to show the directions, etc. The computer realization of

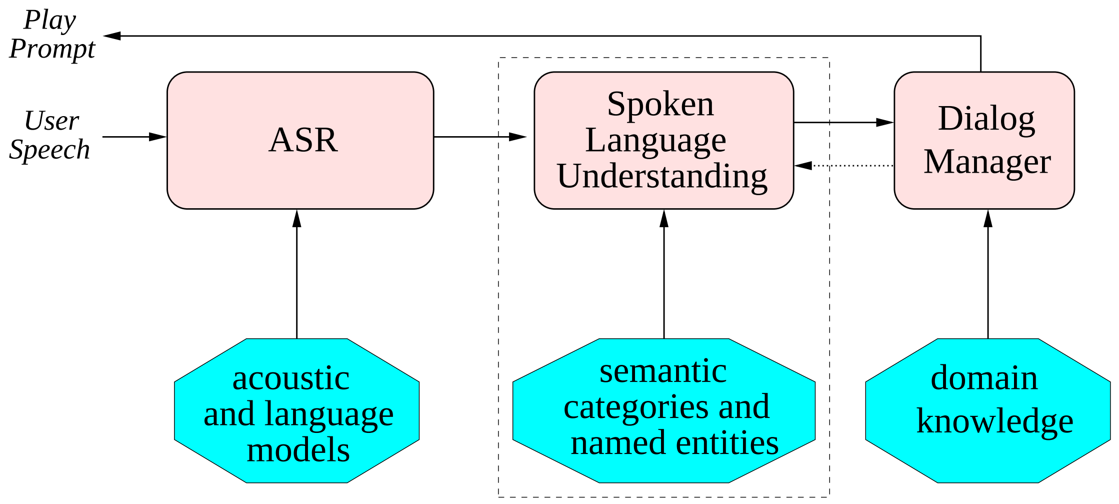

Play Prompt
User Speech
ASR
Spoken Language Understanding
Dialog Manager
acoustic and language models
semantic categories and named entities
domain knowledge

Figure 2.2: Spoken Language Understanding mechanism embedded in a dialog framework.

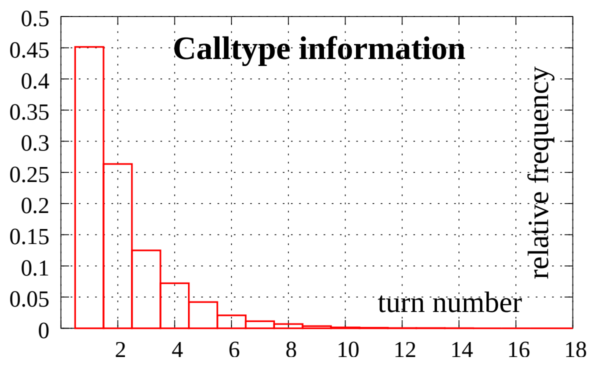
long term memory is highly purpose-oriented. It is a store of declarative and procedural knowledge (database) the understanding system is build as an interface to. Suppose that customer doesn't recognize a call on his bill. To help him, the system has to know who this customer is and also have this call in his database.

Looking back at the Definition 1.1, we note once again that retrieval of new information concerns only the local audio information, while its interpretation is influenced by the dialog context and the world knowledge. As a consequence, one could separate retrieval and interpretation parts in the semantic function $\psi(\cdot)$, as well as in the semantic attributes $\boldsymbol{\nu}$ that go along with it.

Dialog managers are programs responsible for dialog progress on the machine part [Den97, Abe99]. They model humans *reasoning abilities* and have access to all the aforementioned kinds of information: SLU classification results, dialog states hierarchy and systems internal database. A simplified representation of a natural language understanding system designed to sustain the analogy principles described above is shown in Figure 2.2. It also demonstrates how an SLU-mechanism can be embedded in a goal-oriented dialog framework.

Spoken language understanding systems that root in this design can engage their users in dialogs resembling person-to-person interaction. Here is a typical (but contrived) dialog from HMIHY [Gor97] for AT&T operator services:

— *This is AT&T. How may I help you?*
— *Hi, I would like to make a credit card call.*
— *What number would you like to call?*



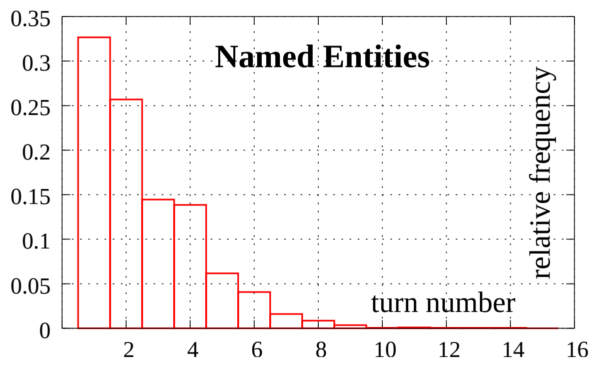


Figure 2.3: Relative frequencies of turn numbers containing calltype information and named entities in *HMIHY Customer Care* corpus.

— *The number is 1234567890.*

— *What is your credit card number?*

— *1234xxxxxxxxxxxx.*

In this example of person-to-machine interaction, the credit card call, customer asked for, must be facilitated. To complete this (or another similar) action, different pieces of relevant information from several utterances the customer makes during the dialog must be put together. The system's objective here is to manage the dialog in such a way that the overall interaction time till the request is successfully fulfilled is minimized [Haf03b]. Effectively, the functional model of utterance understanding as it appeared in (2.4) needs to be invoked each time when a new utterance arrives. Depending on the dialog state, one can expect it to return the semantic function of the action or its semantic attributes (e.g. named entities) or both. The strategic framework we adhere to in our experiments suggests performing the information extraction from the input messages only, and then (if needed) interpreting the retrieved information in the current dialog context. While the details of the interpretation step are not described in this work, our research is centered on the algorithms for extraction of semantic category of an utterance (calltype classification) and semantic attributes (named entity processing).

Despite the fact that all turns of the dialog can contain relevant information, not all of them are equally likely to do so. From the plot in Figure 2.3 we see that among all turns containing calltype information 45% are the first turns (compare this number with the prior percentage of the first turns in the entire data collection which is merely 25%), followed by second turns (26%), after which the percentage keeps on decreasing monotonically.

Similar skewed dependency holds also for the turns containing named entities, albeit with a much flatter slope. There are many reasons that explain this statistic pattern which we will not consider any closer in this work. However, it is important to be aware of this phenomena. This is why in the calltype classification experiments described below, we will distinguish between two cases:

1. all turns;
2. all turns following the system's initial prompt or a re-prompt (when no calltype information could be retrieved from the previous turns).

We will see later that the difficulty of these two tasks is not the same.

2.2.2 Evolution of Calltype Classifi cation Systems

Calltype classification task has come into existence as a compromise between cost-conscious economic policies of the companies that must support intensive telephone contact with their clients and an obligatory user friendliness of this service. It is certainly way too expensive to educate and maintain troops of qualified staff just to have them spend most of their time responding to rather simple queries; on the other hand, exposing customers to the cumbersome contest with the touch-tone IVR- (Interactive Voice Response) service would inevitably lead to their frustration and eventually end up in the company's losing them to competitors.

One reasonable trade-off here is to make machines understand customers' natural speech and carry out autonomously all response actions they can be trusted with, while falling back on human operators only in the cases where there are problems that can not be solved by automatic means. This strategy shifts the burden from the callers to the machines. While previously the callers were forced to comply with rather stringent and often incomprehensible rules imposed on the form of their requests by the IVR-system, now they are allowed to talk freely and it is the task of the machine to understand the pragmatic intentions concealed in their spontaneous speech.

Up to the recent past, most main stream approaches to the automated phone call handling were designed to recreate the complex semantic structure of customer's utterances, so that the most important information (the actual action) was not considered differently from the attributes and other auxiliary information. For instance, in the *frames*-based semantic models of meaning representation like the one of the MIT natural language understanding system for airline travel information ATIS [Zue92], the kind of request (calltype) is just another attribute frame *"clause"* which is formally treated in the same way as the local concepts [Sen92].

Nowadays, another (rather simplistic but therefore more robust) strategy of calltype classification became a viable alternative in many practical call center tasks. Calltype classification systems are expected to tag the input data with one or more semantic categories, which usually come from some predefined set with only a limited number of elements in it. There is an abundant supply of scientific literature on topic classification (see for instance [McD94, Joa98, Big01a]), and most of the theoretical results obtained there also hold for calltype classification task, although one certainly gets to struggle with such an inconvenience as a low quality speech signal (fed from telephone lines) leading to an increased level of noise in the input data.

After a series of preliminary experiments on practical goal-oriented automated language acquisition by Gorin et. all [Mil93, San93, Gor94a], AT&T Labs devised an understanding *"How May I Help You?"* (HMIHY) system that provided automated services to customers calling about their telephone business with the company [Gor97]. In this system, calltypes and other auxiliary information were extracted from the first user utterance, or from the subsequent utterances after the user had been involved in a clarifying dialog. The calltype classification was done by a naïve Bayes classifier based on *salient grammar fragments* [Wri97]: words or word phrases with strong associations with one or several calltypes, extracted from the training corpus.

While in the first application of the HMIHY-system, the *Operator Services* task, the number of rival calltypes was limited to only fifteen (including one open-end class OTHER), the successive extention *Customer Care* task counted already more than fifty supported semantic categories. In both cases, the prior calltype distributions were significantly skewed with the entropy in the later task estimated at about 3.9.

During its deployment and in the course of further development, the HMIHY-system underwent many strategical changes concerning the choice of classifier [Haf03b], dialog management [Abe99] and others.

The commercial success of HMIHY inspired other research groups to employ similar approaches to language understanding in a call center scenario (see for example [Gol99, Tan03]). From calltype identification experiments at BBN Technologies an automated call-routing system CALL DIRECTOR has emerged [Nat02]. This system essentially adheres to the same classification principles as HMIHY. The recognition and classification stages are separated, the results of recognition of the training corpus are used to select semantically relevant *keywords* by means of information theoretic measures. The keywords are then looked for in the test utterances and employed for the classification of the latter.

In the recent past, a new (and empirically better) approach to calltype classification emerged: the pre-selection of salient grammar fragments (or keywords) was given up altogether, instead the type of classifier (typically large margin classifier) was employed that could be trusted with the feature selection when provided with the ASR-output [Sch00, Haf03b]. We use this type of classifiers in our calltype classification experiments.

2.3 Extraction of Semantic Attributes

In Section 2.1.2 we pointed out that, in order to make the behavioristic approach feasible, we have to allow the semantic function of the request-elicited action to have parameters defining objects of this action as well as its characteristics. We called these parameters *semantic attributes*. One emblematic but also hard example of such parameters is given when the domain they come from is considerably large. While parameters can take various values across this domain, the changes in the meaning remain local and don't affect the semantic function itself.

Consider a number of examples where users ask for collect calls to different numbers: *"I'd like to make a collect call to number 1111111 / 2222222 / . . . "*. Whatever the concrete digits, the semantic category of all these utterances remains the same (request for collect call). Compare this with the following negative example: *"I'd like to make a collect / credit card call"*, where also procedural aspects of the required action change when replacing the description *collect* by *credit card*. As a result, in the second case we might consider COLLECT_CALL and CREDIT-CARD_CALL as two different calltypes, whereas different phone numbers from the first example are unquestionable attributes. In this thesis we will concentrate on the algorithms working with these "difficult but clear" attributes.

There are two factors that can contribute to the large size of a parameter domain: high number of out-of-vocabulary words (e.g. proper names) and/or composite parameters, consisting of several words. In the latter case the cardinal number of the parameter set is high due to the combinatorial complexity (e.g. numerical expressions). An important property of the composite attributes is that because of their size they are capable of making up large chunks of the utterance, and thus require elaborate explicite modeling. In the later chapters we will occupy ourselves predominantly with composite attributes, however both groups share the same principle role in spoken language understanding in a call center scenario, and we will not distinguish between them in the rest of this chapter where they are referred to as *named entities*, borrowing the notation commonly used in the message understanding community.

2.3.1 Role of Named Entities in Natural Language Processing

With the rapid growth of language understanding applications, named entity processing has been granted a lot of attention by several speech research groups [App99, Hua02, Bik99, Béc02]. For several years now identification of named entities has been subject for many academic but also practically oriented research projects.

In Section 2.1.2 we introduced named entities as a special case of parameters for functions representing action semantics. One distinctiveness of named entities as semantic attributes consists in the fact that *only a very small fraction of their possible values can be actually observed in the training corpus*. There exist much more possible names and much more possible telephone numbers than any reasonable, naturally collected training set can contain. Yet, just this kind of parameters is very typical for call centers. For example, when a customer reaches a representative, one of the first items to communicate is his phone number, which is a named entity.

The special role this subset of semantic attributes plays in natural language understanding is also voiced in simpler mechanisms for handling errors and ambiguities in cases when the calltype has been understood, but not the value of the accompanying named entities. Consider again the example from page 9. If the system understood that the user wanted to make a credit card call, but suspects a misrecognized destination number, following re-prompt will be played: *"What was the number again that you would like to call?"*. Compare this with the absurd re-prompt in the reversed case, where, for instance, the location was confidently recognized, but not the semantic category of the request: *"What was this thing about Georgia?"*.

The importance of the problem led to the need for standardization. The etymological roots of the notation "named entity" stem from the original definition given by the *Message Understanding Conference (MUC)* which says that named entity is *"... a named object of interest such as a person, organization, or location"* [NIS], so that only proper names like *Socrates*, *Erlangen, Germany* or *AT&T* are covered by it. However, some of the latest proposals distinguish among up to 150 different bottom level NE-types [Sek02]. As yet, MUC-7 provides definitions for three basic types of named entities: ENAMEX (proper names, acronyms, etc), TIMEX (temporal expression) and NUMEX (numerical expressions, monetary expressions and percentages) [Chi97]. On the whole, a single NE-type has the quality of a possibly infinite community of words or idiomatic word phrases capable of sharing the same semantic and syntactic roles in the sentence. By many NE-types (like proper names or phone numbers) these words and phrases are not likely to be found in a general purpose dictionary of the language.

An important aspect of working with named entities that has been stressed in recent publications concerns dealing with imperfect data [Kub98, Pal99, Béc04]. Unlike text documents with preserved orthography, capitalization and other spelling-specific clues for named entities, speech signal does not maintain any other NE-witnesses but their audio characteristics, so that algorithms successfully working with spelling features on text (see e.g. [Col99]) can not be applied here. Besides, speech data is prone to noise and this requires integrating some flexibility in the identification process. As a consequence, handcrafted NE-grammars successfully used on text, were replaced in many systems by trainable stochastic models which resulted in better identification rates [Bik99, Kub98] but made value extraction and normalization more difficult. An approach presented in [Béc04] brought about good value extraction results by paralleling handcrafted and statistically trained grammars. The value extraction method we will present in this thesis offers an alternative approach to combine corpus statistics and our prior knowledge about named entities that leads to improved results and renders the success independent from the detailed annotation of NE-instances in the training corpus.

2.3.2 Rule-based and Statistic Approaches

The battle between empiricism and rationalism in the spoken language processing has been fought for many decades. Which is the best way to model natural language: should the models rely on strict grammars postulating which is a correct sentence in the language and which is not, or on stochastic dependencies signalizing the strength of the sentence's affiliation with the language by means of confidence scores?

There are quite a many advocates of both theories among members of linguistic and computer-linguistic communities [Chu03]. On the one side the empiricists with Shannon's noisy-channel model of communication introducing probability into speech recognition [Sha48], and the n-gram theory as an empirical pendant to Firth's *"You shall know a word by the company it keeps"* [Fir68]. On the other side adherents of Chomsky's rationalistic point of view culminating in *competence models* [Cho57], a system of principles, determining the grammaticality of a sentence.

Since both doctrines have a number of positive and negative factors, there have been attempts to unite them into one theory allowing the grammar of a natural language to have an innate component as well as an empirical component [Pin99]. For automatic spoken language processing this translates into a system that would incorporate statistical and rule-based elements. While several practical SLU-systems adhering to this combined approach were developed in the recent years (see e.g. [Eck96, Gil98]), in the named entity processing, it is just starting to win support [Béc04], and the majority of todays strategies is as yet either rule-based or entirely statistical.

Rule-based and statistical approaches both rely on NE-models to look for in the test data. The distinction between them is that rule-based algorithms make use of handcrafted grammars, usually created by linguists, whereas statistical methods are able to automatically acquire stochastic representations of these grammars from the training corpus.

While handcrafted grammars lead to superior results on clean data, such as written text, in the cases where no spelling information is available or noise is present (speech) statistical approaches turned out to outperform rule-based systems [Bik99, Pal99]. This is mostly because of the misrecognition errors made by the ASR. On the other hand, training stochastic models requires large manually transcribed training corpora. If the goal of a stochastic grammar is not only to recognize named entities in the sentences but also to extract their values by means of a transduction, the amount of labeled data needed for training can become a serious problem. In any case, the stochastic model has to have a certain generalization power to account for the overwhelming majority of those instances not seen in training.

The question arises anyway: why dismiss our knowledge about the ways people commonly use to express named entities, encoded in handcrafted grammars? Observe that a grammar designer doesn't have to come up with new NE-grammars for each application. Many of the named entity types are generic and can be transfered from one domain to another with (almost) no corrections. Indeed, *"September fifteenth"* means the same thing in a call center scenario as well as in the machine translation task, although the application-dependent interpretation of the date can differ.

One way to incorporate this knowledge into statistical approach is to consider manually created and statistically trained NE-grammars in parallel [Béc04]. This however has a disadvantage of possible redundancy in both parts, whereas the motivation behind statistical training is to account for the case where manually created grammars are doomed to fail. A different solution seems to be more appropriate, which takes handcrafted grammars as a basis and enhances them by looking at NE-instances actually observed during the training. While this might be a good way to teach the machine what alternative formulations for a named entity it can expect from the speaker, this approach would also require a sufficient amount of training data with labeled instances of named entities. However, the major premise of this work is that no word transcriptions for the domain of interest are available, which also implicates that NE-delimiting markers are not present in the training data either. Our solution to this dilemma will be described in Section 4.3, where we will explain how independently acquired misrecognition statistics can be collected from a different domain and used for approximate matching of named entity grammar fragments in the corpus.

2.3.3 Named Entity Extraction Today

In this place we describe two innovative approaches that for the past decade have been defining the main streams in the exploration of the named entity extraction task.

About ten years ago SRI Artificial Intelligence Center started working on the rule-based concept to serve various information extracting tasks. Their FASTUS-system [Hob96] was designed to be easily adjustable to a number of information extraction tasks, by separating general linguistic knowledge from domain-specific features of a particular task. Applied to the MUC-6 named entity task, FASTUS also produced good results for named entity extraction from text [App93]. The system is based on a series of cascaded finite state transducers with the most important stages listed below [Hob96]:

- text tokenization: complex words (like multiwords and proper names) are extracted from text;
- on the basis of system's domain-independent knowledge about the language, small linguistic units such as noun groups, verb groups, and other phrases are recognized;
- the resulting representation is searched for patterns of interest (application-dependent)
- if referring to the same event, the found patterns are merged in merging incidents.

The final output of the system had representation conforming to the domain specification.

The successor system TEXTPRO [App99] was designed to meet the requirements of information extraction from ASR-transcripts. Its performance on Hub4 named entity identification task led the authors to believe that the major bottleneck when using rule-based systems for NE-extraction is the word accuracy of the recognizer and suggested that the next major improvement should come from the integration of named entity models in the latter.

As in the earlier nineties availability of large speech corpora stopped being a problem, Mercer's *"More data are better data"* [Chu93] could be put into practice and stochastic methods in speech recognition gained in popularity in virtually all fields of speech and language processing.

In the named entity processing one of the first products that adhered to this principle was BBN's IDENTIFINDER [Bik99]. Similar to FASTUS, IDENTIFINDER's first application was to discover named entities in written text, which it did with the success rate competitive to the systems operating on a rule-based principles. Still relying, among other things, on written word features such as capitalization or numbers, it was yet capable of showing superior performance on the mixed case texts, or when digits were spelled out (so-called *speech form* of text) comparatively to the rule-based algorithms. This was an optimistic result with regard to algorithm's application to speech.

The learning system of BBN considers named entity extraction problem as the one of parsing. When handling K NE-types, the text has to be segmented in word groups, each group tagged as one of the $K+1$ classes: one class for each NE-type and one for "not a named entity".

The parsing is done by means of Viterbi decoding with a quasi ergodic HMMs whose states represent one of these $K+1$ classes each and have unique statistical word bigram language models attributed to them: yet another Markov process that governs this state's word output [Bik97, Bik99]. Additionally, the state transition probabilities of the HMM are conditioned on the last word of the source state. The robustness of the algorithm can be further increased by incorporating back-off and smoothing techniques in the parsing process.

In [Kub98, Mil99] the behavior of IDENTIFINDER on ASR-output was studied. It turned out that even though the NE-extraction performance did drop with increasing word error rates, the dependency of its degradation on word error rate was not more than linear.

Several authors have adopted the ideology of IDENTIFINDER for their named entity extraction tools. The tagged language modeling approach presented in [Got99] also locally constrains stochastic language grammar based on the NE-class of the words. In [Hua02] the baseline algorithm of IDENTIFINDER has been extended by the possibility of continuous adaptation of the global stochastic model on the local profile of a particular discourse domain. The algorithm for named entity extraction described in [Béc02] models not just the named entities per se, but also the syntactic phrases adjacent to them, thus facilitating extraction of context-specific named entities which change their type depending on what syntactic and semantic context they are used in; besides the stochastic models are learned from ASR-transcriptions of the training data which allows to compensate for recognizer's errors.

The illustrativeness of named entity extraction task permits for easy online demonstration programs, especially by extraction from texts. One such example for NE-demo on the web is offered by the SAIL Labs for free evaluation [Med].

2.4 Moving Away from Manual Transcriptions

Several studies have shown that for the understanding applications there is a remarkable correlation between understanding rates and recognitions accuracy, whereby improving the recognition quality of the underlying ASR also leads to a better performance on the understanding task [Bor96, Bag99, BV03]. These observations suggest the importance of a fine-tuned speech recognizer employed in an understanding system.

2.4.1 Language Model Adaptation

What factors contribute to a good ASR-performance? The answer to this question lies in the basic formula of language decoding. Given an acoustic observation A, the goal is to find a word sequence W such that:

$$W^* = \underset{W}{\operatorname{argmax}} P(A|W)P(W). \tag{2.5}$$

This formula is based on the Bayes decision theory and minimizes the classification error rate [Nie, page 281],[Hua01, page 137]. It has two parts, each responsible for acoustic ($P(A|W)$) and language ($P(W)$) modeling respectively. The obvious intention is to estimate the probabilities in (2.5) as precisely as possible. While we are not going to address the issue of building acoustic models in this dissertation and in our experiments simply rely on the available ones, we regard the correct estimation of the language model (LM) as a relevant topic.

Let Σ^L be the word lexicon of language L. Then, for each word sequence from the set of combinatorially possible sequences $W \in (\Sigma^L)^*$, the language model determines its membership in the language. In the case of stochastic language modeling, it also provides a quantitative measure (likelihood) of W's affiliation with L:

$$P(W) \longrightarrow [0,1] \text{ with } \sum_{W \in (\Sigma^L)^*} P(W) = 1.0. \tag{2.6}$$

In other words, a stochastic language model induces a probability distribution on the set of all word sequences. This definition is valid for formal as well as natural languages. Even though one can talk about the language model of English in general, it is clear that the only practical way to provide a data-driven estimate for this kind of language model is to consider a representative sample of English. In practice, it is uncommonly difficult to create such a sample. What is tractable instead, is obtaining samples pertaining to specific domains of the language, like business communications or technical discussions. The result is a language model trained on and optimized for a particular language domain. Fortunately, most of the time this is also the aspired goal of the training, for any of today's practical applications concern with one particular facet of our communicational activities, like issuing a broker order or trying to remedy a network outage over the telephone.

On the other hand, since the number of life situations where an effective intervention of the person-to-machine communication technologies can be carried out is rapidly growing, one would want to have a certain language model flexibility, which would permit a fast and inexpensive re-tuning of an existing more general language model with as little data as possible and as fast as it gets. This strategy would be much more efficient than collecting large amounts of language material for a completely new language model each time a new conversation domain has to be served. The need for adaptation becomes even more critical when the domain of interest doesn't have any or has only limited resources of textual training material, but there is another reliably estimated “out-of-domain” language model available [Ros95, Iye98]. Here again, we would like the existing model to be trained on a corpus at least partially similar (in lexicon and in stochastic word dependencies) to the domain in question. This is usually provided for by a language model trained for the same language and covering a broader spectrum of topics. Such general language model is called *background language model*. Adaptation can come in handy also within one thematically and/or circumstantially restricted language domain. The discourse domain and speaking style can vary with time, so that periodic dynamic adjustments of the language model can improve the recognition [Kuh90, DP92, Kne93].

2.4.2 Unsupervised Language Model Adaptation

So far, the assumption was that one does have a corpus of manually created in-domain word transcriptions and, thus, can estimate a new language model (or at least adjust the existing one) based on the stochastic dependencies observed in this corpus. However, producing such transcriptions is very expensive since it involves engaging significant human resources. It is also extremely time consuming: for telephone speech, the estimates of real-time factor of high quality word transcribing process range from 20 to 50 [Bar01, Hol01] and the growth is super-linear in utterance length. If a new application is to be brought up in a short time, this kind of delay can be critical for the system’s rapid beginning of operation in the field.

Concerns like this led to a recent surge of interest in the unsupervised LM-adaptation [Bac03, Yok03]. By unsupervised LM-adaptation we have a robust background language model at our disposal and also an audio corpus for the domain of interest. This audio corpus is recognized using the background language model and, based on the ASR-output and the background language model itself, a new adapted language model is constructed.

At this point a simple but effective approach for unsupervised LM-adaptation that also leads to improved understanding rates is described. The adaptation scheme (see Figure 2.4) consists of two alternating steps:

1. recognize the in-domain audio corpus with the current language model (background model at the beginning of iterations);

2. create a new stochastic language model by counting word n-grams only in this ASR-output.

Since sufficient amounts of spoken material is available for the adaptation, the background language model is used only as a starting point for iterations and is neglected after the first pass.

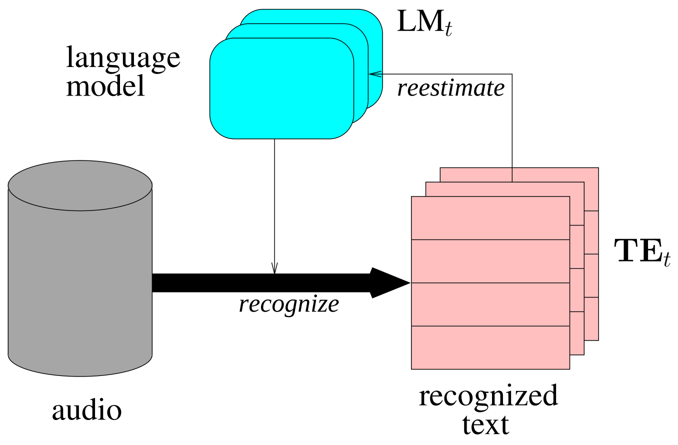


Figure 2.4: Iterative algorithm for unsupervised language model adaptation.

This downsizes drastically the adapted model, while preserving important content words, whose correct recognition is crucial for good understanding results [Bor96]. Before each recognition step some parameter adjustments must be done, in order to strengthen the relative contribution of the language model in the basic recognition formula (2.5). In the experiments we report on in this thesis a language model weight κ_i related to word insertion penalty [Hua01, page 610] was integrated in formula (2.5) and gradually increased from 8 to 16:

$$W^* = \operatorname*{argmax}_W P(A|W)P^{\kappa_i}(W). \tag{2.7}$$

2.4.3 Unsupervised Training of Phone Language Models

What happens if the background language model isn't available either? We would find ourselves in situations like this, for example, when dealing with languages without established written form, like Taiwanese [Klö03], or when the lexicon is extremely domain specific and the nature of the discussions is very different from the traditional language. These restrictions mean that the available word statistics will not help recognition or understanding but, on the contrary, can be misleading.

However, topic classification tasks, unlike the problems where deep semantic analysis is required, have no obligatory assumptions about word-level data representation [War97, Gor99, Lev01, Als03]. In [Lev01] we presented an efficient way to acquire salient phone sequences, which are then combined in *acoustic morphemes* (AM) and used for calltype classification [Lev02]. The idea behind acoustic morphemes is to replicate syntactic and semantic role words play in the language. We will address the process of AM-creation and use in Chapter 5. Alshawi in [Als03] achieved good calltype classification results by using boosting as the classification vehicle. In his experiments the classifier was operating directly on phone recognition output.

By changing the choice of basic recognition units from words to subword units like phones, we don't eliminate the need for language model estimation, but to some extent aggravate it, for now the language model for phones must be created from scratch. This, however, can be done in a way very similar to the one in Figure 2.4. Starting with a phone-loop (zerogram language model) one can alternately recognize the corpus and re-estimate the language model based on the resulting ASR-output. In [Als03] on each consecutive iteration a stochastic language model of higher order was created, which made each new model more explicitly defined than its predecessor. Due to the small number of phones in the vocabulary (usually about 50), estimating n-gram probabilities for phones is feasible for n up to 5 and even higher. We adopted this approach for phone-based calltype classification and named entity processing and report on the achieved results in the next chapters.

Chapter 3

Calltype Classification without Transcriptions

Spoken language understanding is a research field within automated language processing dedicated to the extraction of meaning from naturally spoken speech. Unlike Shannon's idea of communication where the objective is to assure signal transmission with minimal loss, and the meaning conveyed by this signal is regarded as not important [Sha48], in the task of language understanding meaning is the ultimate goal and error free recognition is only one of the possible steps towards reliable meaning extraction. According to the behavioristic interpretation of understanding suggested in Chapter 2, meaning is approximated by the semantic function of the aspired machine action along with the values of parameters this function takes. We also have seen how the semantic functions of the actions are reflected in the semantic categories of the utterances that are suppose to elicit them and pointed out the enormous potential complexity of the space of possible semantic categories of speech utterances. For many real-life applications, such as call center, however, it is usually sufficient to restrict the scope of supported categories to a rather small number. The understanding problem restricted in this way will possess entropy much lower than the entropy of the language itself. In his famous guessing experiment with human subjects, Shannon estimated the entropy of written English at 0.6 to 1.3 bit per letter [Sha51]. Given the average word length of 5.5 letters per word (including separating spaces) [Jel90], we arrive at the entropy estimate of about 5.5 bit/word. Assuming that an average utterance contains about 10 words, the entropy of the language is about 55 bit/utterance. On the other hand, the understanding complexity of the call center task with, say, 100 semantic categories, leads in the worst case of uniformly distributed and mutually independent semantic categories to the entropy values of ca. 6.5 bit/utterance.

This crucial difference leads to a reasonable presumption of redundancy in the language (and in the speech signal in particular) with respect to the understanding task, which implies that not every part of the signal is important for its correct understanding. Given that, the exact recognition of all words in the speech signal appears not to be necessary anymore. Instead, we may restrict the search space to the parts of the signal which are truly essential for conveying meaning (we call them *salient*), whereby it is the application itself that defines which words are salient and which ones are not. For example, in the HMIHY task, the utterance *"Well, hello, I wanted to ask you for a hm... a billing credit"* only contains two salient words *"billing"* and *"credit"*; these salient words can be used to derive semantic category of the request.

3.1 Positioning of the Task

In Section 2.2 we have looked at calltype classification as one practical application of topic classification with individual topics optimized according to the needs of call centers employing it. Whichever theoretical results hold for topic classification, they automatically apply to calltype classification as well.

Except for the possible explicit assumption of noise in the input data, the topic classification task itself pertains to a broader family of *text categorization* problems. In text categorization, there is a number of categories formed by some application-dependent criterion; for each of these categories several document examples are provided. While an abstract definition of the criterion may be specified (e.g. agenda, intended action, author characteristics), the challenge of the task is to automatically find out its manifestation in wording of the documents, enabling thus a consistent classification of the previously unknown examples.

The problem of text categorization has been repeatedly addressed in the literature with a broad spectrum of agendas including news categorization [Hay91], user profiling [Lan95], calltype classification [Sch00] and many others. In the case of calltype classification, the classification criterion is the semantic category of the utterance, more specifically: its calltype.

Even though, we assume that salient words are important for text categorization, finding these is by far not enough to solve the problem successfully. At best, these words can act as features for the actual classification problem, as opposed to a well studied *word spotting* task [Hua94, Kni96] where finding keywords in speech or text is a goal in itself.

Before we explain this distinction in detail, let's state another major difference between (key)word spotting and text categorization using the calltype classification example. In word spotting the choice of the keyword(s) to seek for is usually determined by the user (or by the requirements of a higher level application utilizing word spotting mechanism to achieve its goals). On the contrary, the challenge of calltype classification is to *determine* the keywords that are best associated with target calltypes. Hereby it is important that the choice of a particular keyword is affected not only by one calltype it is supposed to indicate, but by the community of all supported calltypes, and therefore should be determined in a discriminative way. So, *"credit"* would make a good representative keyword for calltype CREDITCARD_CALL with the only possible alternative COLLECT_CALL, but would be a poor choice for distinction between CREDITCARD_CALL and BILLING_CREDIT. In other words, the degree of usefulness of a keyword for recognition of a semantic category is determined by the relative strength of the semantic associations (salience) between this keyword and this semantic category.

Salience of a word w has many mathematical incarnations. The simplest is PMAX, the maximum of conditional semantic probabilities of utterances that contain w:

$$\text{PMAX}(w) = \max_m P(c^m|w), \tag{3.1}$$

where $\{c^1, \ldots, c^M\}$ are the semantic categories.

A slightly more complicated measure comes from Information Theory and is based on the *Kullback-Leibler distance* between two distributions:

$$\text{KL}(w) = \sum_{m=1}^{M} P(c^m|w) \log \frac{P(c^m|w)}{P(c^m)}. \tag{3.2}$$

The second measure is truly discriminative since it compares conditional probabilities for all semantic categories, whereas PMAX focuses only on the most probable one. Besides, KL salience only marks a keyword if it significantly affects probabilities of semantic categories relative to their prior distribution. In a way both measures complement each other: one signalizing that the word can be salient at all and the other specifying the semantic category for which it is salient (if any). There are also other popular salience measures (see, for instance, an informative comparison of various selection criteria for selecting classification features in text categorization in [Yan97]).

In many cases, a cheap way to increase the salience of a word is to grow it by one or several adjacent words. In the example above *"credit card"* becomes a reliable indicator of the utterance calltype even in the second task setup. Thus, another specific characteristic of text categorization (and, in particular, of calltype classification) is that one considers not just single words but rather sequences thereof. Henceforth, we will call linear sequences of words (or phones) that we automatically extract and use in our algorithms, *strings*. Using strings makes the strategy portable to the phone-base case (Section 3.3.2) where utterances are represented as phone sequences (or lattices). Obviously, the notion of salience and its practical realizations also hold for arbitrary string s of adjacent words or phones.

Along the same line of thought, we observe that it is necessary to take into consideration all encountered keyword instances in the utterances to make a prediction about its semantic category. Say, we found string *"credit card"* in the utterance, but also word *"misplaced"* at some other point. Now, the plausible guess is that the caller tried to make a credit card call but the call was misplaced and the caller is now asking for a credit for it.

Summarizing, we see that calltype classification is indeed a more elaborate task than word spotting, but it can use the results of the latter by, first, devising the list of relevant keywords from training data and then employing the detected instances along with their scores as classification features for the final classification of the test utterances. The classification itself can be performed by various external classifiers (see, for instance, [Wri97, Mye00]).

The question arises, however, whether the preselection based on the simple salience measures above, in fact generates classification features which are optimal with respect to the classification task? Observe that when this preselection is done, feature-strings are picked out based on their salience estimated for each string independently. But for classification, the totality of all used features is considered, which relativizes the individual salience values. For example, both strings *"credit card call"* and *"credit card call to"* are probably equally salient for category CREDIT-CARD_CALL, but discrimination power of the second string in the presence of the first one is insignificant[1].

On the other hand, experiments suggest that there are only few irrelevant word-features in text categorization. In [Joa98] the results of Bayes-classification with alternative non-overlapping feature groups are presented. Each group contained words ranked similarly by the *mutual information gain* criterion. The experiments showed that even the group of the features ranked lowest still led to classification much better than purely by chance, albeit that otherwise the rank of the group correlated positively with the classification performance. This suggests that ideally, relatively few pruning effort at the stage of feature generation should be made. However, not doing feature selection at all would usually lead to a number of features which for most classifiers is prohibitively high.

In the recent years much research has been done in the direction of the *large margin classifiers*. Because of their internal selectivity, this type of classifiers can handle a large number of features. We will talk about these classifiers and their application to calltype classification in the

[1]General rules of feature selection for classification are described in detail in [Nie].

following section.

3.2 Large Margin Classifiers

In this section we present classification algorithms we chose to employ for the calltype classification task and named entity detection task that will be discussed in Section 4.2. We decided to use large margin classifiers because of their outstanding generalization performance. Large margin classifiers belong to the category of distribution-free classifiers which model boundaries between classes, instead of approximating the distribution densities of classes themselves [Nie, page 346], [Dud01, page 215]. They strive for separating classes by hyperplanes that have the highest possible distance to their critical points.

Suppose, we are to separate two classes in an d-dimensional space $\mathbb{R}^d$. The training data consists of pairs

$$(\boldsymbol{x}_i, y_i) \;\; i = 1 \dots I, \quad \text{where } \; \boldsymbol{x}_i \in \mathbb{R}^d \text{ and } \;\; y_i \in \{-1, +1\}. \tag{3.3}$$

This problem can be formulated as a search for decision function

$$g(\boldsymbol{x}) : \mathbb{R}^d \mapsto \{-1, +1\}, \tag{3.4}$$

whereby $g(\boldsymbol{x})$ is often considered as *sign*-threshold of a real-valued linear function

$$G(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{w} + b, \quad \text{with} \quad \boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^d \text{ and } \;\; b \in \mathbb{R}. \tag{3.5}$$

The $d-1$-dimensional hyperplane defined by equation $G(\boldsymbol{x}) = 0$ determines *decision boundary*: the unknown points will be classified to one class or the other, depending on what side of this boundary they lie on. If the problem is linearly separable, i.e. there exist at least one hyperplane that enables error-free classification of the training data, then there is an infinite number of such hyperplanes (Figure 3.1). In fact, for a training sample of size I, a valid solution is any hypothesis function $G(\boldsymbol{x})$ for which holds:

$$\rho_G := \min_{i=1 \dots I} \rho(\boldsymbol{x}_i, y_i) = \min_{i=1 \dots I} y_i G(\boldsymbol{x}_i) > 0. \tag{3.6}$$

It can easily be seen that the quantity $\rho(\boldsymbol{x}_i, y_i)$ has a geometric interpretation as the distance between point $\boldsymbol{x}_i$ and hyperplane $G(\boldsymbol{x}) = 0$ (negative distance means misclassification); therefore it is called a *margin* of $\boldsymbol{x}_i$. From here it is intuitively clear that the higher the minimum margin ρ_G

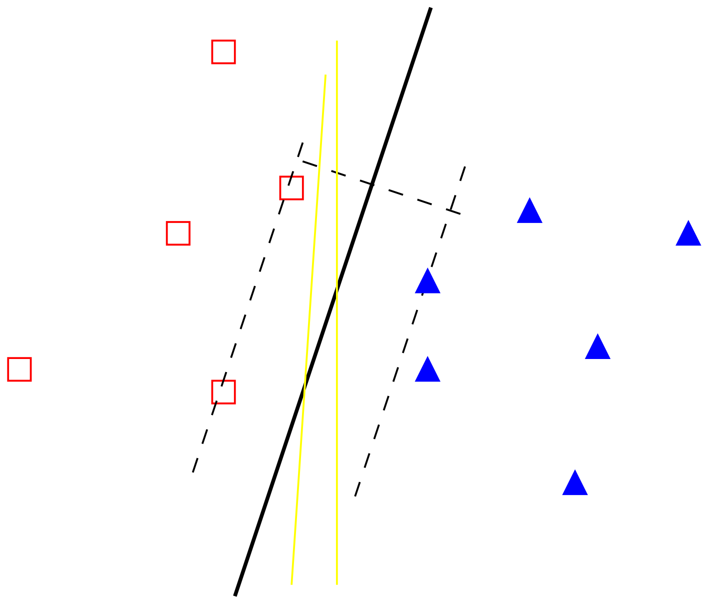


Figure 3.1: Separating two classes by a hyperplane; the "best" separation is provided by the hyperplane with the largest margin to both classes.

over all training points, the more robust generalization in patterns can be expected from the corresponding decision boundary [Smo00]. Thus, the objective of training a large margin classifier is to find the hyperplane G^* with parameters $\boldsymbol{w}^*$ and b^* such that the margin ρ_{G^*} is maximal:

$$\boldsymbol{w}^*, b^* \;:=\; \underset{\boldsymbol{w},b}{\operatorname{argmax}}\, \rho_G. \tag{3.7}$$

With no constraints on $G(\boldsymbol{x})$ this maximum doesn't exist though. Therefore some precautions must be made: one can impose a length restriction on $\boldsymbol{w}$: $||\,\boldsymbol{w}\,|| \equiv 1$ or work with the normalized decision functions [Smo00]:

$$G(\boldsymbol{x}) = \frac{\boldsymbol{w}\cdot\boldsymbol{x} + b}{||\,\boldsymbol{w}\,||}. \tag{3.8}$$

It is also possible to formulate the optimization problem in such a way that a risk minimizing solution is possible, even in cases where errors by hyperplane separations are inevitable [Vap95, Cor95].

Practical realizations of large margin classifiers like support vector machines or boosting (see below) usually have extensions which allow application of these techniques to the cases where discrimination among several (more than two) classes is needed.

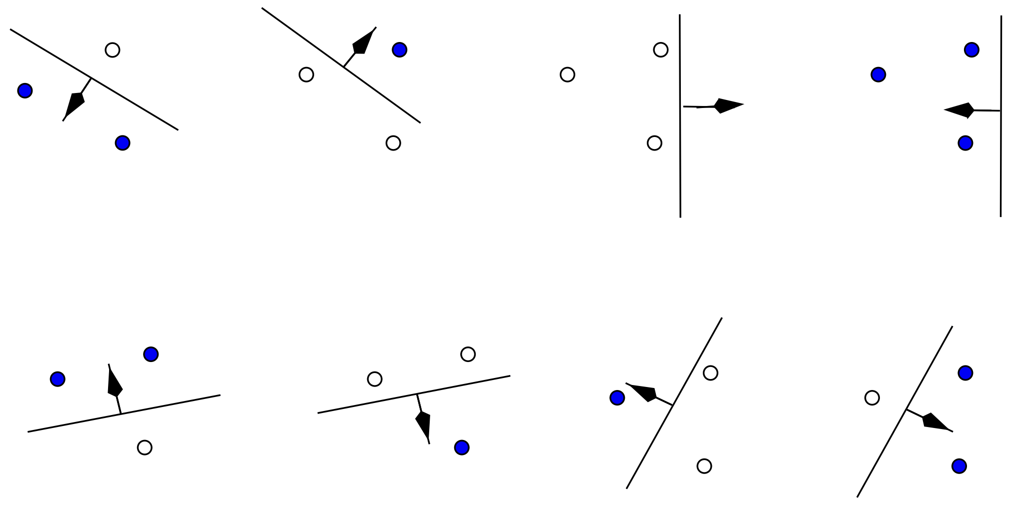


Figure 3.2: Shattering three points in $\mathbb{R}^2$ by the set of oriented straight lines; from [Bur98].

3.2.1 Support Vector Machines

Theoretical Foundations of Support Vector Machines

Support Vector Machines (SVMs) [Vap95, Bur98] are probably the most famous and commonly acknowledged member of the family of large margin classifiers today. Rooted in Vapnik's *Structural Risk Minimization (SRM)* theory [Vap82] they strive for finding a compromise between the actual error achieved on the given training set and the intrinsic complexity of the classifier, its discrimination "capacity" characterized numerically by the *Vapnik-Chervonenkis- (VC-) dimension*.

Let $\mathcal{G}$ be the set of decision functions (3.4), and let $\bar{I}$ be the highest number such that there exist a set of $\bar{I}$ points $\boldsymbol{x}_i \in \mathbb{R}^d$ that can be *shattered*, i.e. labeled in all possible $2^{\bar{I}}$ ways by functions from $\mathcal{G}$. Then $\bar{I}$ is the VC-dimension of the set $\mathcal{G}$.

In Figure 3.2 the constellation of three points in $\mathbb{R}^2$ is shown to be shattered by the set of oriented straight lines. However, there is no four points in $\mathbb{R}^2$ that can be shattered by the same set. Thus, the VC-dimension of straight lines in $\mathbb{R}^2$ is equal three[2].

The higher VC-dimension of a function set the better the chance of finding a decision function from this set that will label the training set error-free. At the same time the risk of overfitting increases too. In [Vap95] it has been shown that the estimator of the misclassification rate on unseen test data (*expected risk*) can be expressed as a sum of the observed error rate on the training data (*empirical risk*) and another term which depends linearly on the VC-dimension of

[2]In general it can be shown that the VC-dimension of a hyperplane-classifier in $\mathbb{R}^d$ is equal to $d + 1$.

the chosen set of decision functions.

Stemming from this theoretical result, the SRM suggests hierarchical split of the set of decision functions into a number of concentric subsets with monotonically decreasing VC-dimensions. By trying out these subsets on an independent validation set and comparing the estimates of the expected risk, the final set of decision functions is established [Bur98].

To train an SVM-classifier for a linearly separable task, following assumption about linear decision function (3.5) is made:

$$G(\boldsymbol{x}_i) \begin{cases} \geq 1, & \text{if } y_i = +1 \\ \leq -1, & \text{if } y_i = -1, \end{cases} \tag{3.9}$$

so that the separating hyperplane will lie exactly in the middle between two parallel "bounding" hyperplanes: $G(\boldsymbol{x}) = -1$ and $G(\boldsymbol{x}) = +1$. With $\boldsymbol{w}$ being the normal to the separating hyperplane $G(\boldsymbol{x}) = 0$ as it was defined in (3.5), it can be easily shown that the margin ρ_G in this case is equal $1/||\,\boldsymbol{w}\,||$. From this it follows that margin maximization is equivalent to minimization of $||\,\boldsymbol{w}\,||^2$, so that (3.7) becomes:

$$\begin{cases} \boldsymbol{w}^*, b^* \;:=\; \operatorname{argmin} ||\,\boldsymbol{w}\,||^2 \\ \text{where} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) \geq 1, \; i = 1 \dots I. \end{cases} \tag{3.10}$$

Vapnik has shown in [Vap82] that if all training examples $\boldsymbol{x}$ are contained in a ball of radius ϱ, the VC-dimension of the hyperplane decision functions (3.5) on which condition (3.9) is imposed, is bounded by:

$$\text{VC} \leq \min((\varrho^2, ||\,\boldsymbol{w}\,||^2), I) + 1. \tag{3.11}$$

This relation demonstrates how support vector machines are in fact adherent to the SRM-ideology.

Finding a numerical solution for the optimization problem (3.10) is difficult, therefore one usually considers its Lagrangian formulation and switches over to a corresponding dual problem [Fle87, Bur98]:

$$\begin{cases} \lambda_i^* \;:=\; \operatorname{argmax} \sum\limits_i \lambda_i - \frac{1}{2} \sum\limits_{i,j} \lambda_i \lambda_j y_i y_j \, \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \text{subject to constraints:} \;\; \sum\limits_i \lambda_i y_i = 0 \;\text{ and }\; \lambda_i \geq 0 \end{cases} \quad i = 1 \dots I, \tag{3.12}$$

where λ_i's are Lagrange multipliers and the hyperplane normal $\boldsymbol{w}$ can be then computed as:

$$\boldsymbol{w} = \sum_i \lambda_i y_i \, \boldsymbol{x}_i \,. \tag{3.13}$$

To determine b^* one can consider any of the points $\boldsymbol{x}_i$ for which equality $y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) = 1$ holds, and solve this equality with respect to b[3]. Now, to classify an unknown point $\boldsymbol{x}$, compute:

$$\text{sign}(\boldsymbol{w}^* \cdot \boldsymbol{x} + b^*) = \text{sign}\left(\sum_{i=0}^{I} \lambda_i^* y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^*\right) = \text{sign}\left(\sum_{i \in \text{SV}} \lambda_i^* y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^*\right), \quad (3.14)$$

with SV as an index subset for those vectors for which $\lambda_i^* > 0$. Thus, only those training examples participate in the computations for which the corresponding Lagrange multiplier is positive. It is these vectors, also called *support vectors*, that define the two hyperplanes $G(\boldsymbol{x}) = \pm 1$, the space between which is cut in the middle by the decision hyperplane.

The analysis above was designed for the case when linear separation of the classes is possible. Yet in many practical problems this is not so. Support vector machines deal with this either by allowing for (but penalizing) misclassifications [Cor95] or/and by transferring the feature vectors in a higher-dimensional space, where linear separation would become feasible [Bos92]. For the latter case, the so called *kernel trick* is used [Aiz64] which takes advantage of the fact that in the optimization problem (3.12) and in the solution (3.14) feature vectors occur only pairwise and always in a dot product form.

Suppose we found a transformation of vectors $\boldsymbol{x}$ in the original feature space $\mathbb{R}^d$ in a higher-dimensional (usually Hilbert-) space $\mathbb{H}$, where linear separation is possible:

$$\Psi : \mathbb{R}^d \mapsto \mathbb{H}. \quad (3.15)$$

Then formulae (3.12) and (3.14) will contain dot products of the form $\Psi(\boldsymbol{x}) \cdot \Psi(\boldsymbol{y}),\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. If we now find such a kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ that:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \Psi(\boldsymbol{x}) \cdot \Psi(\boldsymbol{y}), \quad (3.16)$$

neither the transformation $\Psi(\boldsymbol{x})$ nor the dot product will have to be computed explicitly, saving a great deal of computational complexity. The choice of a particular kernel transformation is usually made empirically, however there are several well studied traditional kernel functions. Two of them are:

1. *polynomial kernels*:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^{\text{deg}} \quad (3.17)$$

[3]In practice, to increase robustness one usually takes several such points on both sides of the separating hyperplane and sets b to the average computed value.

with polynomial degree deg usually equal 1 or 3;

2. *radial-basis kernels* (note that the dimension of space $\mathbb{H}$ here is infinite):

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{2\sigma^2}}. \tag{3.18}$$

Despite their succinct form, the kernels can correspond to very complicated mappings (3.15). For instance, the polynomial kernel mentioned above corresponds to a mapping into a high-dimensional space $\mathbb{H}$ where each element of the new feature vector is computed as a product of up to deg elements of the original vector $\boldsymbol{x} \in \mathbb{R}^d$ [Smo00]. The effect of the kernel trick is that while still using a linear hyperplane to separate high-dimensional feature vectors in $\mathbb{H}$, mapping back into the original feature space $\mathbb{R}^d$ will produce a non-linear separation plane. In our experiments for calltype classification (and later for named entity detection) we will use polynomial cubic kernels. Theoretical properties of different kernel functions as well as the conditions for their existence are discussed in detail in [Bur98].

Llama: SVM-toolkit by AT&T

The success of support vector machines in various classification tasks [Cor95, Sch96, Pra04] (and their proven usefulness for text categorization in particular [Joa98]) caused a considerable surge of demand for well documented all-purpose software for SVM training and classification. Several computer science groups offered their products for public use. Among them are SVM^{light} from the University of Dortmund [SVM] and LIBSVM from the National Taiwan University [LIB].

We have chosen the SVM-toolkit LLAMA developed by Patrick Haffner from AT&T Laboratories [Lla], because of its good performance, versatile user interface, and the opportunity of direct support by the author. The toolkit allows for flexible kernel parameter settings and encourages user's initiative in the empirical search for many other training and classification parameters, while also suggesting robust default configurations.

For text categorization, LLAMA recommends taking all n-grams present in the training sentences. Those n-grams with the expected number of occurrences in one sentence over some pre-specified threshold should be pre-selected as classification features. This mechanism can be extended to deal with the most general kind of ASR-output: weighted word (or phone) lattices. For an audio signal, *weighted lattice* is a compact way to encode its several decoding alternatives with their respective probabilities as one finite state automaton (see Chapter 4). In the most general case of a corpus consisting of I weighted lattices $\mathcal{L}_i$, the expected number of occurrences for some n-gram s is estimated as:

$$\bar{\#}s = \frac{1}{I}\sum_{i=1}^{I}\sum_{S\in\mathcal{L}_i} \#_{(S)}sP_{\mathcal{L}_i}(S), \tag{3.19}$$

where the second summation runs over all decoding alternatives S encoded in lattice $\mathcal{L}_i$, $\#_{(S)}s$ denotes the number of occurrences of s in S and $P_{\mathcal{L}_i}(S)$ stands for the probability of S in this lattice.

Since n-grams extraction (in our experiments we considered all n-grams with $n \leq 5$) can also be implemented as a composition of two finite state transducers[4], employing n-grams as classification features is a special case of *rational kernels* on lattices [Cor03]. Rational kernels express the kernel function $K(\cdot,\cdot)$ of two weighted lattices as a function of the transduction costs between the individual paths through them [Cor03]. These kernels can further be combined with other families of kernels, like, for instance, polynomial kernels of the form (3.17), and thus become efficiently incorporated in the SVM training and classification procedures as described earlier in this section.

The multiclass classification option in LLAMA is realized through the "one-against-others" mechanism [Haf03b]. Suppose, there are M semantic categories $\{c^1,\ldots,c^M\}$ to classify among. Then M simple binary classifiers are considered separately, each responsible for taking apart $c^m,\ m\in 1\ldots M$ and all other categories pooled together. The vector of these classification outputs is then used to make a multiclass decision, while the *class independence* assumption is made. This assumption implies that affiliations of one example $\boldsymbol{x}$ with two different classes are two independent events:

$$P(c^{m_1},c^{m_2}|\,\boldsymbol{x}) = P(c^{m_1}|\,\boldsymbol{x})P(c^{m_2}|\,\boldsymbol{x}), \tag{3.20}$$

an approximation which is more than welcome in the framework of calltype classification where several calltypes can occur simultaneously.

3.2.2 Boosting

Theoretical Foundations of Boosting

Together with other combining machine-learning classification techniques such as *random subspace method* and *bagging*, *boosting* is a method of organizing numerous "weak" classifiers of

[4]Finite state automata and, in particular, finite state transducers will be discussed in Chapter 4.

low reliability with the goal of producing one "strong" classifier of high prediction accuracy [Sch02].

Often it is rather straightforward to create a weak classifier. In the example from Section 3.1, string *"credit card"* salient for calltype CREDITCARD_CALL delivers one such classifier, for its presence in an utterance would indicate this calltype with probability significantly higher than the prior. Obviously, it shouldn't be difficult to come up with many weak classifiers like this (consider, for instance, all strings with PMAX > 0.5). How to aggregate the power of all these weak classifiers?

Let's look at our classification problem again. Given the training corpus (3.3) we were looking for a decision function (3.4) that separates two classes in $\mathbb{R}^d$. The boosting algorithm ADABOOST[5] [Fre97] suggests classification by means of a convex combination of a number of weak classifiers. These classifiers are selected during an iterative procedure with the following alternate steps on each iteration:

1. modify the distribution of training patterns so as to emphasize the patterns which were misclassified on the previous iteration (initially uniform distribution is assumed);

2. from the pool of available weak classifiers select the one most useful under the assumption of the modified distribution.

Technical details of the training and classification processes are provided in Figure 3.3.

In [Sch98] the upper bound for the empirical loss of ADABOOST-classifier was proven:

$$\bar{\text{Loss}} = \prod_{t=1}^{T} |Z_t|, \tag{3.24}$$

where Z_t are the normalization coefficients from (3.22). This bound prescribes the selection procedure for the best weak classifier on each iteration: *select the weak classifier minimizing upper loss bound (3.24)*. The same rule also guides the choice of weight coefficients μ_t.

From (3.21) we see that, like support vector machines, boosting concentrates on the training examples that are hardest to classify. It then selects the one classifier that is most helpful by the classification of these examples. In [Sch98] it was also proven that *AdaBoost* is at the same time a large margin classifier.

BoosTexter

BOOSTEXTER is a software package written at AT&T Laboratories specifically optimized for

[5]ADABOOST is a short name for *Ada*ptive *Boost*ing

```
GIVEN: $(\boldsymbol{x}_i, y_i) \in (\mathbb{R}^d \times \{-1, +1\});\quad i = 1 \ldots I$
INITIALIZE: distribution $\mathcal{D}_1(i) = 1/I$
FOR $t = 1, \ldots, T$
    get best weak classifier $h_t : \mathbb{R}^d \mapsto \mathbb{R}$ for $\mathcal{D}_t(i)$
    choose coefficient: $\mu_t \in \mathbb{R}$
    update distribution:
        $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i) \exp\left(-\mu_t y_i h_t(\boldsymbol{x}_i)\right)}{Z_t}$    (3.21)
    with normalization factor $Z_t$:
        $Z_t = \sum_i \mathcal{D}_t(i) \exp\left(-\mu_t y_i h_t(\boldsymbol{x}_i)\right)$    (3.22)
OUTPUT: final classifier
        $g(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \mu_t h_t(\boldsymbol{x})\right)$    (3.23)
```

Figure 3.3: Training and classification with ADABOOST (after [Sch02]).

multiclass multi-label text categorization problems [Sch00], that is, it allows for more than two classes in the formulation of the classification problem, and also can handle patterns with several labels attached to them.

To each pattern $\boldsymbol{x} \in \mathbb{R}^d$, in this formulation, there is a corresponding subset of labels y from the set of the known labels $\mathcal{Y}$, so that the weak hypotheses used by the BOOSTEXTER are all of the form:

$$h(\boldsymbol{x}, y) : \mathbb{R}^d \times \mathcal{Y} \longrightarrow \mathbb{R}, \tag{3.25}$$

i.e. for each pattern/label pair they provide a real-valued prediction probability of their co-occurrence. In our experiments we used the simple hypothesis space, suggested in [Sch00]: one-level decision trees checking presence or absence of a word (string) s in the utterance:

$$h(\boldsymbol{x}, y) = \begin{cases} \text{const}_{1y} & \text{if } s \text{ is present in the utterance;} \\ \text{const}_{0y} & \text{if } s \text{ is not present in the utterance.} \end{cases} \tag{3.26}$$

In fact, we decided to select strings s from the pool of sparse word (or phone) n-grams with $n \leq 5$, so that each feature vector $\boldsymbol{x}$ contains 1's for those n-grams from the training set that are also present in the given utterance and 0's for all the others. For each s, we set $\mu_t = 1$ and select const_{1y} or const_{0y} so as to minimize the normalization coefficient Z_t which (if this s is selected) would correspond to it, and is a multiclass analog to the normalization coefficient in (3.22). Then, the hypothesis based on the string s with the minimum Z_t is selected as the next weak classifier [Sch00].

Optimizing Z_t corresponds to the minimization of the *Hamming loss*: the fraction of pairs of the examples $\boldsymbol{x}_i$ and labels y_i for which classification produced wrong results. While other objective functions are possible, we will stick in our experiments to this version, since, according to the original paper [Sch00], it was shown to produce the best text categorization results.

3.3 Using Large Margin Classifiers for Calltype Classification

In the previous sections we presented algorithms that we will use for calltype classification in the experimental part of this thesis. The baseline training and classification processes can be carried out according to the following straightforward schema:
Training:

1. manually create word transcriptions for a large corpus of audio data from the domain of interest;
2. based on these transcriptions, create word lexicon and estimate a language model;
3. recognize audio data from training corpus using this language model;
4. manually provide each transcribed utterance of the training corpus with one or (where necessary) several calltype labels (e.g. CREDITCARD_CALL or PAY_BILL);
5. train a large margin classifier (in multiclass mode) to predict calltypes; use the n-grams extracted from the textual representation of the training utterances as classification features.

Classification:

1. recognize audio representation of the utterance using the same language model;
2. use the trained classifier to predict utterance calltype.

It should be noticed that the most human effort is spent at step 1 of the training procedure. The challenge of this work is to eliminate the need for manual word annotation of the data. Assuming that we know what language is being spoken and have an off-the-shelf background language model for this language, we could try and use it for recognition and hope that the misrecognition errors will remain consistent across different calltypes not hurting classification much [Sey97].

3.3.1 Unsupervised Adaptation of Background Word Language Model

Another promising solution has been outlined in Section 2.4.2. It uses unsupervised language model adaptation to refine statistic dependencies contained in the background language model with respect to the audio data from the domain of interest. This refinement typically results in a leaner lexicon (the unseen words will not participate in the language model creation on the consecutive iterations), which has a substantial speed-up at the classification stage as a welcome side effect.

Note that the idea of using topic identification for language model adaptation has been around for quite a while now (see for instance [Sey97, Big01b]). We, however, pursue a reverse strategy of employing the technique of unsupervised language model adaptation in order to improve the classification results. We suspect that both adaptation processes take place in parallel during communication. Knowing the potential conversation domain certainly helps word recognition by shifting the very generic background language model into a specific direction which, in its turn, contributes to a better understanding (semantic interpretation) of individual utterances.

Thus, in the case of missing manual transcription, steps 1 and 2 of the training algorithm above get replaced by:

1^*. starting with the background language model, perform its unsupervised adaptation to the domain of interest as described in Section 2.4.2.

Of course this same language model will also be used for recognition of unknown utterances at the classification stage.

3.3.2 Phone-based Calltype Classifi cation

As we already pointed out in Section 2.4.3 it is possible to solve calltype classification problems even when there is no information about the language used. The reason is the independency of the training and classification algorithms from the nature of the text stream.

Note that in the sections of this chapter dedicated to the classifier choice we never insisted on words being the only adequate basis for selecting n-grams to use as classification features. In

n	$\geq$3-grams	$\geq$4-grams	$\geq$5-grams
proportion	90%	65%	40%

Table 3.1: Percentage of phone n-grams classification features selected by boosting for different values of n.

fact, similar classification features could be extracted from the ASR-output expressed in terms of syllables, phones or data-driven subword units as well. Certainly, the advantage of words consists in their innate individual meaningfulness, often articulated as salience in the classification tasks. This leads to a considerable proportion of unigrams among selected classification features: in our boosting experiments this proportion was about 50%. To acquire a satisfying classification potential with phones, unigrams are by far not sufficient, for the associations between individual phones and calltypes are very weak. Table 3.1 shows average observed n-gram percentages among selected phone strings in the phone-based boosting experiments.

It is remarkable that about 50% of selected features are longer than the average length of English words (4.5 symbols), leading to results consistent with the single word proportion in the word-based classification mentioned earlier.

We will compare the results of word- and phone-based calltype classification in absence of manual transcriptions in Section 6.3.

Chapter 4

Processing of Semantic Attributes without Transcriptions

In this chapter we will talk about processing of semantic attributes in the framework of a behavioristic approach to automated spoken language understanding as it was presented in Chapter 2. Having introduced semantic attributes as parameters of semantic functions of the intended actions, we focused on one prototypical class of such parameters that is particularly important for the practical applications from the call center scenario: semantic attributes with a large number of possible values. Typically these attributes are either realized by words that are not in the ASR-vocabulary (e.g. proper names) or consist of more than one word (e.g. phone numbers). In any case, only a small fraction of all possible values can be accounted for in any training set of a reasonable size. In Chapter 1 we decided to call these attributes *named entities* (NE), leaning on the traditional notation from the literature.

At this point, we will present our view at the processing of named entities. While the algorithms that will be described were optimized to handle named entities, and some of them even work only with those named entities that can be modeled with grammars, the conceptual analysis of the NE-processing problem will retain its validity when applied to any kind of semantic attributes.

Similar to calltype classification, many of the algorithms from this chapter will remain unchanged (or require minimal adjustments) no matter what the representation level of the underlying data: words or phones. Therefore, our algorithms will be discussed in the most general task setup. Explicit remarks will be made wherever there is a need to underline the differences caused by a certain representation level or due to the absent manual transcriptions.

4.1 What We Want to Know about Named Entities

In Section 2.3 we explained the role of semantic attributes in the mapping of natural language requests into the space of machine actions, and justified our focusing on named entities. Along with proper names, numerical and time expressions are most commonly considered to be named entities [Chi97]. In the end, there will be a number of *named entity types* (generic, like, for instance, proper names, time expressions etc. or application-dependent such as departure times, client numbers and so on) that one will be working with.

While extensive work on named entity definitions is being done, we believe there is a substantial lack in the procedural aspect of named entity processing. For instance, little attention was paid to the extraction and normalization of values of named entities, once those have been found. In this section we suggest a categorization of named entity related tasks, explaining differences but also underpinning interconnections among them. Later in this chapter we will also suggest methods for solving each of these subtasks.

We distinguish three major subtasks in the named entity processing [Lev04]:

4.1.1 Detection Task

The purpose of this task is to decide whether a given utterance contains named entities of the specified type. A simple "yes/no" answer (or the probability of the presence) is sufficient. This particular subtask of named entity processing seems to have been largely neglected up to now, while the most attention was focused on the extraction task. Nonetheless, named entity detection has an importance of its own. One example is exploiting presence of named entities of certain types in an utterance to facilitate calltype classification (see previous chapter) of this utterance. In [Béc04] strong co-dependencies between occurrences of named entities and some calltype labels are reported. For instance, the probability of calltype UNRECOGNIZED_NUMBER in the presence of at least one instance of named entity ITEM_PLACE (geographic place the call was made to) in a sentence was 53.5%, and the probability of calltype RATE_CALLING_PLANS was 10%, while their prior probabilities were mere 4% and 1.5% respectively. Another possible implication of named entity detection results is the modification of the dialog manager behavior. Consider the situation where the system acts on the assumption that the caller is talking about some phone call he allegedly made. Besides, let its confidence of having detected a named entity in the utterance be rather high, but the overall confidence score of the recognized text very low. In this case, the computer can re-prompt relating to the already given information: *"Please, repeat the date of this call once again"*, rather than conceding the understanding failure with something like

“Sorry, I didn’t understand you”, followed by a repetition of the previous prompt. Finally, in the framework of active learning [Ric03] named entity detection could be employed to distinguish and extract utterances most informative with respect to named entities to retrain their acoustic and language models.

4.1.2 Localization Task

The goal of this task is to find out how many instances of the specified named entity type are present in the utterance and determine their exact locations. Applications like SCANMAIL [Whi02] are designed to provide high-end interface to voice mail, simplifying and speeding up user’s interaction with the core system. For this software it is essential to find the exact location of important information in the ASR-transcription of the utterance and draw user’s attention to it by highlighting the region or recovering its audio representation. There is no need for the system to actually *understand* the meaning of the found named entity; just localizing all its instances in the signal will suffice. Most of the research in the named entity processing so far was concerned with this application [Ben97, App99, Kub98].

4.1.3 Value Extraction Task

For each encountered instance of the specified named entity type, the task is to extract its value. If we want to design an autonomous system capable of conducting dialog with the user by its own means, information extraction will become its indispensable component. Take, for instance, the HMIHY domain described in [Gor97]. Obviously, there are numerous ways for the callers to express the same information. *“May six of year two thousand”* and *“the sixth of May two thousand”* certainly mean the same (at least in the application context), albeit the user employs different means to convey this meaning. In order for the system to react adequately to user’s request, it must prescind from a particular wording and concentrate on the valuable bits of information. In other words, a *normalization* has to be carried out. In our example, both expressions will be transformed into something like “05.06.2000”.

These three subtasks often depend on one another. So, before we start looking for starting and ending positions of putative named entities in the utterance, it is worth turning to detection mechanism first for an accurate prediction as to whether the utterance in question contains any instances of this named entity type at all. Given that issuing such a prediction is often computationally inexpensive, and the percentage of utterances containing named entities can be fairly

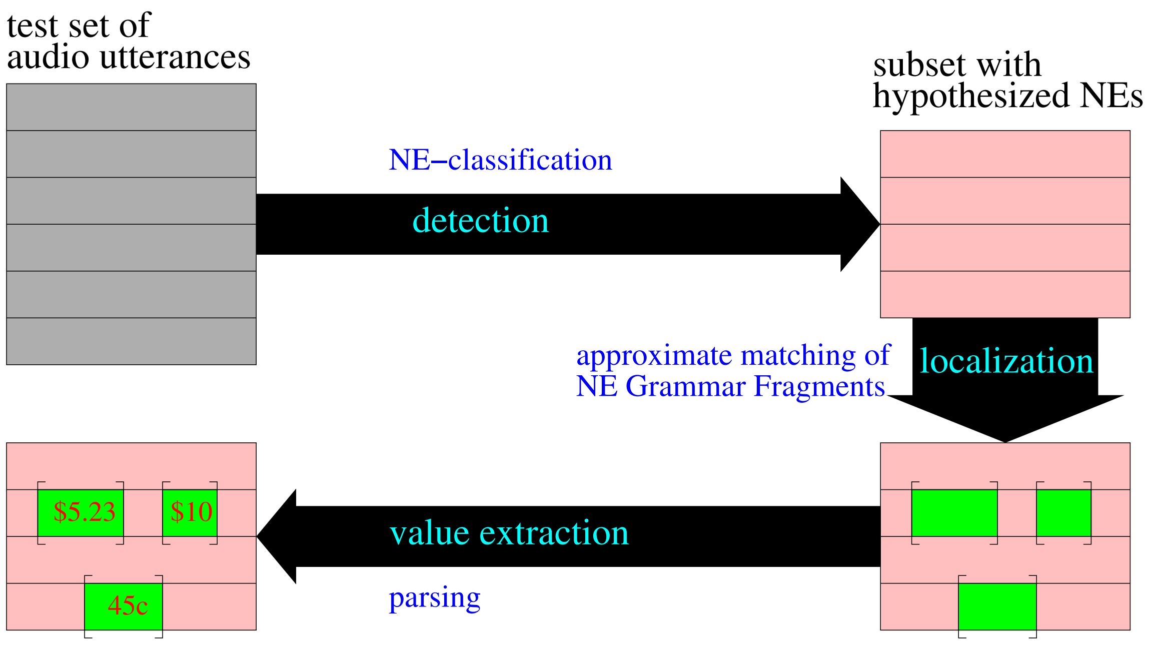


Figure 4.1: Three stages in named entity processing: detection: find utterances likely to contain named entities; localization: find positions of all named entity instances; value extraction: determine normalized values of encountered named entities.

low [Béc04], the amount of saved computational power is considerable. We will show later that also localization quality can be improved by the preceding detection step. Similarly, it is needful to localize the interval with a named entity instance, before trying to extract its value.

A schematic dependency of the subtasks (in the batch mode) is illustrated in Figure 4.1. We see how the named entity detection stage first reduces the corpus with suspected named entities down to only those utterances in which named entities are likely to occur. During the next step, the localization procedure determines where to look for named entities in these selected utterances. Finally, all intervals enclosed between corresponding starting and ending positions will be passed to the parser to extract and normalize the named entity values.

4.2 Named Entity Detection

To detect named entity of a specified type in an utterance means to make a decision about presence of one or more instances of this type there. Formally speaking, this can be described in the following manner:
Let there be K supported named entity types $\nu^1, \nu^2 \ldots, \nu^K$. Then, the detection task is to find K

binary functions $A^k(\cdot)$ (one for each named entity type) that for each utterance $W \in \Sigma^*$ make predictions as to the presence of the corresponding named entities in it:

$$A^k(W) : \Sigma^* \mapsto \{-1, +1\}, \tag{4.1}$$

where "-1" means absence and "+1" presence of ν^k in W.

Observe that knowledge of the innate nature of a named entity (e.g. its grammar) is helpful but not necessary to make a guess about its presence in the utterance. In other words, if we managed to localize an instance of some named entity type, we implicitly answered the detection question positively, whereas the opposite is not true: spotting a named entity in the utterance alone would usually provide no information about its exact location in it. Besides, the localization failure doesn't necessarily mean that the named entity is not there at all, but can also be ascribed to deficiencies of the localization method, which is usually more sophisticated and requires more specific domain knowledge than a detection algorithm. To detect a named entity we don't have to model it, but to collect indicators for (or against) its presence from the utterance.

The nature of these indicators can be manifold. The most plausible source of information is certainly the utterance itself. As in the case of calltype classification, we prefer to work with the recognition results instead of audio signal, for the latter contain loads of redundant information (like time information, speech frequency, loudness and others), whereas the textual representation is largely content-relevant. The *textual indicators* of named entities are words or word strings whose presence in the utterances correlates with the presence of a given named entity type.

For instance, the presence of the word *"called"* in the text increases the probability of at least one named entity of type PHONE_NUMBER from its original prior 3% to 14%. And the word *"number"* boosts it to 25%. Each indicator by itself represents a weak classifier, yet, as we have seen in Chapter 3, they can be combined to obtain one strong classifier.

In general, textual indicators can:

- be part of named entity itself (e.g. lie in its *core*);
- belong to its direct context;
- be located anywhere in the utterance while correlating with named entity occurrences.

Intuitively, words and word strings that constitute named entities themselves are the best evidence for their presence. Encountering *"twenty dollars"* in text obviously increases the probability of money-related named entities there. The next group of textual indicators are word strings that are formally not part of named entity but pertain to its immediate enclosure. Consider the expression *"Last month I paid the amount of twenty* too *dollars"*. Here the misrecognized monetary named entity *"twenty two dollars"* is preceded by the word string *"paid the amount of"* which, being a strong indicator for a subsequent named entity itself, helps to detect (and even extract) it despite possible errors in the NE-core. The context is especially important for the so-called *context-specific* named entities [Béc04] whose interpretation and even recognition is subject to semantic and pragmatic requirements of a particular application. For example, the HMIHY labeling guide [Alv03] defines named entity ITEM_AMOUNT which is considered present if and only if the monetary amount it refers to, is printed anywhere on the customer bill[1]. According to this definition, sentence

> *You sent me a* **bill for** *twenty dollars.*

contains a named entity of the type ITEM_AMOUNT (underlined), but

> *I sent you a* **check for** *twenty dollars.*

does not. If we were to detect this named entity via its direct modeling, we might have to include the context *"bill for..."* in the model. However, we would encounter actual difficulties at the latest by trying to model context *"check for..."* as a witness for named entity absence rather than for its presence. We will see later that this is not that much of a problem in our approach where detection is considered as a classification task. The last group of textual indicators that contribute to named entity detection retains only loose areal dependency on the named entity core. The proximity either has a weak impact on detection:

> *I* **paid** *for this call more than two dollars.*

or is irrelevant at all (as opposed to the pure fact of concomitance):

> *Yes, I want to* **dispute the following charges** *on my last month bill: you made me pay twenty dollars for...*

In the latter case, observe that the exact wording of the textual indicator (here: *"dispute the following charges"*) can vary, while the really conclusive information about named entity presence is contained in the semantics of the utterance. Once it has become clear that the caller's intention is to dispute a charge, we can expect him to mention the amount he is talking about as well. This circumstantiates the idea of using also other indicators than those extracted from the (recognized) text of the utterance for named entity detection.

[1]For more information on the NE-definitions used in this work see Appendix C

In compliance with the conclusion from the previous paragraph, the first candidate for *non-textual indicators* is the calltype of the utterance. However, the calltype itself is extracted from the same textual representation, and thus, according to the *data processing inequality* [Cov91, page 32], should not improve classification, but rather introduce additional noise. At the same time, there is another source of *reliable* information available from the dialog manager. In the *computer-initiative* and *mixed-initiative* types of person-to-machine dialog [Lev00a] the user's utterances always follow computer prompts, even though the user is allowed to switch topics. In general, rules of traditional person-to-person conversation apply, and the user is expected to advert in some way to the prompt previously generated. For instance, if the last prompt was: *"Please, give me your phone number"*, the probability of finding a phone number in the next customer utterance will be much higher than after *"Is there anything else I can help you with?"* Thus, we can take the system prompt as an additional non-textual indicator for named entity detection. This goes in line with the importance of modeling of dialog acts and other system information for language model optimization in the speech recognition, noticed in several experiment studies [Eck96, Hac01, Béc03]. Besides, it accomplishes the first step in the direction of the dialog-step-dependent interpretation of the retrieved information, as outlined in Section 2.2.1.

So far, we have been talking about selection of indicators that can be used for named entity detection, but postponed the discussion of the procedural aspect. How can detection be carried out? Based on the similarities between equations (4.1) and (3.4), we suggest a rather intuitive way of *detection-by-classification*. The detection task, when looked at from the classification perspective, degrades to a number of simple utterance classification tasks with two induced classes: "contains NE" and "doesn't contain NE". Thus, we can employ an available classifier to solve the detection problem using the indicators above as classification features. As in the case of calltype classification, we suggest using large margin classifiers, namely boosting and support vector machines, and restrict the textual indicators to (gappy) n-grams again.

In many cases the detection task must be performed for $K > 1$ named entity types in parallel. For this kind of situations a substantial speed-up can be achieved by replacing K independent detection processes (i.e. K classifiers) by one classifier with $K + 1$ classes, one of them being rejection. This approach has an advantage of discriminative detection, which accommodates the fact that instances of different NE-types can compete for the same time interval in the signal. As an example, consider the following sentence:

My number is **one sixteen oh three** *X X X X X*.

with *X* as an arbitrary digit. Here the obvious named entity is a phone number 116-03XXXXX. However, the first four digits on their own can also account for a date expression (January, 16^{th},

2003) or for a monetary expression ($116.03). Only the fact that the phone number named entity fits much better (e.g. with respect to the coverage) into the entire utterance, will proclaim its clear victory, while slashing the scores of the others almost down to the level of the rejection class.

For each new utterance, the classification process will produce $K+1$ real-valued scores, each corresponding to one of K supported named entity types or to the rejection class:

$$(\nu^1, \nu^2 \ldots, \nu^K, (\nu^{K+1} = \nu^{\text{rej}})) \longrightarrow (p^1, p^2 \ldots\ldots p^K, (p^{K+1} = p^{\text{rej}})) \tag{4.2}$$

To make binary decisions in the spirit of (4.1), we compare the scores achieved for each named entity type with a threshold θ. We declare named entity type $\nu^k \neq \nu^{\text{rej}}$ for detected if and only if $p^k > \theta$.

Similarly to the calltype classification, the named entity detection can also be performed on ASR-lattices, provided the classification feature extraction mechanism can handle them. In this case one can also take advantage of the confidence scores each word in a lattice is provided with. Besides, the same detection mechanism can also be applied to the output of a phone recognizer instead of a word recognizer.

4.3 Named Entity Localization

One of the most widely accepted problems in the automated SLU is semantic attributes extraction from speech. While knowledge of the presence of an attribute in an utterance can come in handy for its understanding, it is at most just a means to finding out the exact value of one attribute and its interplay with the others. Yet before the value of an attribute can be extracted, the corresponding part of the speech signal must be pegged. For instance, if the recognized utterance was: *"I got this bill for too dollars just yesterday"*, the system must convert it into something like *"I got this bill for* <NE-start> *too dollars* <NE-end> *just yesterday"*, following the convention that the part between the markers <NE-start> and <NE-end> must be a named entity. In other words, one has to *localize* an instance of the attribute before passing the putative hit over to a parser which will then educe its value. As in the previous section, we will avail ourselves of the named entities to illustrate attribute localization. Before we continue, let us stress that while named entity localization, that is finding their starting and ending positions in an utterance, is an important step towards extracting their values, this process can be important in itself, as in the SCANMAIL application mentioned above. In fact, most of the effort in NE-processing concentrated so far on this stage, which also became objective in the named entity task definition of MUC-7 where it is called *named entity identification*.

How can localization be performed? The task is essentially to separate the parts that account for the attributes from the rest of the signal. One way to attack this problem is by identifying boundaries of attribute instances. So, in [Bik97, Bik99] Hidden Markov Models (HMMs) [Rab89] were used to separate named entities from the rest of the text. Each word was labeled either NAME or NOT-NAME, and the HMM-states were designed to represent these classes[2], while generation of a particular word given the presence (or absence) of the named entity was ruled by a bigram language model.

Because of the *Markovian property* of HMMs and bigrams [Cov91], the decision about the NE-boundary between two consecutive words was made only based on the words themselves and their suspected named entity affiliations. This *locality* condition may work well for proper names but will fail in the cases where overview over the entire attribute is needed, e.g. because of its application-dependent definition. Consider, for example, the named entity representing phone numbers. About American phone numbers we know that they consist of seven (without area code), ten (with area code) or eleven (with area code and leading "1") digits. Thus, if we want to decide whether to put a boundary between *"is"* and *"one"* in the utterance fragment: *"... the number is one two three ... "*, we have to be able to look up at least six words ahead of *"one"* and check if they are all digits as well. This is impossible with the standard HMMs, but can be achieved when the whole named entity is modeled.

While reliable modeling of the entire named entity is extremely difficult for proper names in speech, because of their high out-of-vocabulary rate and elevated number of misrecognitions as a consequence, other named entities like, for instance, time or monetary expressions can be successfully modeled in this way [Béc02]. The pivotal decision about the nature of the model to use concerns the source of the knowledge it will be built according to. In Section 2.3.2 we already compared two polar approaches to named entity extraction, one stemming from the competence models and premising manual handcrafting of the grammars, and the other adhering to the statistical principles of learning. There, it was pointed out that while handcrafted attribute grammars allow for high coverage with least expense, they are not able to compensate for misrecognition errors.

But what if we could find a way to incorporate approximate pattern matching in the process of instantiation of manually created NE-grammars in the noisy ASR-output? From this approximate matching we would like to expect good predictions of the typical misrecognition errors of the

[2]The actual algorithm was capable of handling several named entity types simultaneously, so that one state was created for each NE-type.

employed ASR-system. In Section 4.3.4 we will describe the design of the approximate matching mechanism that plays a crucial role in our localization procedure.

Once the misrecognition problem is solved (or at least alleviated), the rule-based modeling of named entities regains our attention because of two major advantages that the statistical approaches don't possess:

- Speakers are willing to go for a great variety of different ways to express one and the same semantic attribute value. The number is skyrocketing with the increasing complexity of lengthy semantic attributes, such as many of named entity types are. One can say *"June first"*, *"first of June"*, *"June the first"*, *"the first of June"*, *"June one"* etc., and all this will mean one and the same day of the year. Given the high number of possible combinations of days of the week, days of the month, months and years, the variety is truly impressive, yet it can be encoded as a compact regular grammar by a linguist in a matter of hours. Even if some of the expressions were "forgotten" at some point, they could be easily added anytime later. Learning all these alternatives automatically from training corpus, however, either means an intricate generalization procedure or requires large amounts of training data, which contravenes the usually limited size of the available training corpora;

- On the other hand, a large training corpus means an increase in the annotating effort. Remember that automated learning of NE-grammars from a corpus presupposes availability of the instance-delimiting markers in the utterance transcriptions. Additionally in this method, the corresponding NE-values must be annotated as well, in order to train the automatic value extraction. Since the general premise of this thesis is to explore the possibilities of avoiding manual transcribing, automatic learning of NE-grammars must be ruled out, and we must try to utilize handcrafted generic named entity grammars instead, extending them by some approximate matching mechanisms.

4.3.1 Named Entity Fragments

At this point we describe grammars that we used to model named entities and the *named entity fragments*, finite state automata derived from these grammars to be employed in the localization mechanisms. We restrict the scope of our interest only to those named entity types that can be modeled by grammars at all, ignoring the rest (e.g. proper names with a high out-of-vocabulary rate). The good news is that for call centers, actual names are often not as important as customer identification numbers, such as account number, phone number etc., and these can in fact be modeled with grammars.

<START>	→	<NUM> dollars
<NUM>	→	<DG> | <TE> | <TN> | <TN><DG>
<DG>	→	one | . . . | nine
<TE>	→	ten | . . . | nineteen
<TN>	→	twenty | . . . | ninety

Figure 4.2: Grammar representing simple monetary expressions (e.g. *"twenty five dollars", "five dollars"* etc.; symbol "|" separates alternative expansion rules.

What is a grammar anyway? The common formalism used to describe grammars is a quadruple $(\mathcal{N}, \mathcal{T}, \mathcal{P}, N^s)$. Here $\mathcal{N}$ is a set of *nonterminal symbols (nonterminals)* and $\mathcal{T}$ is a set of *terminal symbols (terminals)*. Set $\mathcal{P}$ consists of *rewrite rules* of the form $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}$ which map sequences of symbols $\boldsymbol{\alpha} \in (\mathcal{N} \bigcup \mathcal{T})^*$ with at least one nonterminal into another symbol sequence $\boldsymbol{\beta} \in (\mathcal{N} \bigcup \mathcal{T})^*$. The expansion starts with a special *start nonterminal* $N^s \in \mathcal{N}$. A grammar example in Figure 4.2 represents simple monetary expressions from one to ninety nine dollars[3] and is a strongly simplified version of the actual grammar that we use to localize named entity ITEM_AMOUNT in our experiments. Consider, for instance, expression *"twenty five dollars"*. It can be *derived* using the following rewrite rules from this grammar: 1) <START>→<NUM> dollars 2) <NUM>→<TN><DG> 3) <TN>→ twenty 4) <DG>→ five. In a similar way sentences belonging to a natural language can be parsed, albeit the grammars suitable for their modeling must be much more powerful than the one shown above.

In general, the choice of a grammar is determined by the language it is supposed to model. James Allen imposes the following requirements on a grammar:

> *In constructing a grammar for a language, you are interested in* **generality**, *the range of sentences the grammar analyzes correctly;* **selectivity**, *the range of non-sentences it identifies as problematic; and* **understandability**, *the simplicity of the grammar itself* [All95, page 44].

While generality and selectivity are determined by the particular choice of rules, grammars simplicity is characterized by the nature of the allowed rules and is inversely related to its expressive power. While recommending to select always the simplest grammar of the adequate ones, Chomsky in [Cho59] proposed the following hierarchy:

[3]For the sake of simplicity we do not treat the special case of "one dollar" differently here.

Type 0 grammars (a.k.a. *unrestricted grammars*)
this is the most general grammar within the bounds of the definition above; no further restrictions on the constitution of α or β are imposed;

Type 1 grammars (a.k.a. *context-sensitive grammars*)
the only additional restriction on this grammar is that for each rewrite rule $\alpha \longrightarrow \beta$ the following holds: $|\,\alpha\,| \leq |\,\beta\,|$ or $|\,\beta\,| = 0$;

Type 2 grammars (a.k.a. *context-free grammars*)
this is a subtype of the Type 1 grammars, with the left hand side of the rewrite rules also satisfying the following condition: $|\,\alpha\,| = 1$, that is, α consists of only one nonterminal; the example grammar we presented in Figure 4.2 belongs to this class;

Type 3 grammars (a.k.a. *regular grammars*)
this is a subtype of the Type 2 grammars; these grammars permit at most one nonterminal in the sequence β. Depending on whether the nonterminal is allowed only after or only before the terminals, these grammars are subdivided in *right-regular* and *left-regular* grammars respectively.

Even though context-free grammars might sometimes be inadequate for language interpretation and understanding [Shi85], they (often with stochastic extensions) seem to have struck a good compromise between generative power and feasible complexity for human language recognition [Sto94, Che96]. In our case, the requirements are yet even more modest: we are not interested in modeling the entire language with its complicated linguistic schemas, but rather a small subset of it (phone numbers, date expressions etc.) that keeps the minimum of syntactic structurality.

Each rule in our grammars has one nonterminal on the left hand side and a regular expression of terminals or nonterminals in its body. This is a convenient way to compactly represent context-free grammars. Moreover, our grammars are acyclic, i.e. for each rule the following assertion holds: the nonterminal from the rule's head may not occur in the expansion of the nonterminals from the rule's body. This means that it is possible to establish a partial order on all nonterminals of the grammar, which makes this grammar a regular (Type 3) grammar[4]. About regular grammars we know that they are equivalent to *finite state automata* [Hop79], for which an efficient modeling tool was at our disposal [FSM]. Thus, we converted the named entity grammars into finite state *acceptors* (see their definition below), which were used in further processing. The finite automata that encode NE-grammars are called *named entity fragments*.

[4]For more information on the rewrite rules used in our experiments see Appendix C.

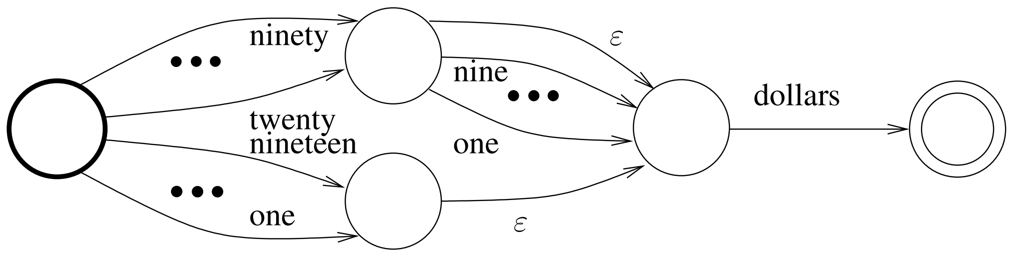


Figure 4.3: Finite state acceptor accepting the language modeled by the grammar in Figure 4.2; ε is an *empty symbol* (see page 71) and means that no symbols are needed to take the arc.

Modeling semantic attributes with finite state automata and their extensions is by no means an unexplored area of research in the spoken language understanding [Ehr90, Gil98]. In [Ehr90, pages 102ff], for instance, *Augmented Transition Networks (ATN)* were used to model complex time expressions in German. Along with other ATN-based models, these grammars were integrated in the *semantic network* structure of the knowledge-based pattern understanding system ERNEST [Nie90] with the overall goal of utterance meaning interpretation [Kum91].

Let us now disclose the formalism of acceptors. Finite state acceptors are finite state automata formally defined by a 5-tuple $(\mathcal{Q}, \Sigma^I, \delta, Q^s, \mathcal{Q}^F)$, where $\mathcal{Q}$ is a finite set of *states*, Σ^I is a finite *input alphabet*, $Q^s \in \mathcal{Q}$ is the *initial state* and $\mathcal{Q}^F \subseteq \mathcal{Q}$ is a subset of distinguished *final states*. The transition function $\delta : \mathcal{Q} \times \Sigma^I \longrightarrow \mathcal{Q}$ determines states that can be reached from a current state by absorbing an input symbol [Hop79]. The notation "acceptors" is used because each of these automata *accepts* (all and only those sentences of the) language defined by a regular grammar (or regular expression) equivalent to it. Our grammar example from Figure 4.2 can be modified into a right- (or left-)regular grammar and therefore represents a language that will be accepted by the finite state acceptor in Figure 4.3. Additionally, each arc of an acceptor and each final state can be provided with costs that it takes to move along this arc or terminate in this state respectively. These costs are usually taken from some *semiring* $\mathbb{K}$ and such automata are called *weighted* over $\mathbb{K}$ and can be used to model stochastic grammars.

In the same way, we encoded our named entity grammars in the named entity fragments for the purpose of their localization in the ASR-output. At this point, it needs to be said that the ability to handcraft these fragments certainly presupposes a certain degree of knowledge of the language in question. This is by no means a contradiction to our strategy of the unsupervised adaptation of the word-based language models, since some basic elements of the language (like, for instance, date expressions) remain the same throughout various thematically different domains. The second alternative of the phone-based algorithms that we consider in parallel, is in fact conceived for the case that our information about the language is not even sufficient to create a background model. However, we believe that building a local phone-based grammar for just a small cutout of the language which each named entity accounts for, requires effort and costs much lower than by creating a lexicon of the entire new language, even for the case of a restricted discourse domain.

4.3.2 Localization via Maximum-likelihood Parsing

The idea behind our localization strategy is to find such a parse of the ASR-output in terms of named entities that has the highest likelihood [Lev02]. For the sake of simplicity, let's consider the case where only instances of one named entity type are searched for (the case of several NE-types is a natural extension thereof). The lexicon $\varSigma^{\mathrm{NE}}$ for parsing consists of *fragments* ϕ^j that are of two types:

1. all words $\{w^j\}$ present in the ASR-dictionary $\varSigma$;
2. one special fragment ϕ^{NE} representing the named entity in question.

To compute the likelihood $P(A|\,\varSigma^{\mathrm{NE}})$ of some utterance submitted as the audio signal A, we need to introduce three hidden variables. First, there is a hidden sequence of fragments forming a particular parse of the produced ASR-output in terms of the elements from the lexicon $\varSigma^{\mathrm{NE}}$:

$$\varSigma^{\mathrm{NE}} = \{\phi^j\} = \{\phi^{\mathrm{NE}}, w^1, w^2, \dots, w^J\}. \tag{4.3}$$

Similar to the named entity fragments, the word-fragments w^j can also be thought as representing primitive grammars defining trivial one-word-languages. Each fragment in the parse is realized by one particular sentence in the language it accepts, which we call *(valid) path* q through the fragment ϕ. In the case of word-fragments these paths are the words themselves. Thus, with no loss of generality, the paths q can be treated as word strings (see the definition on page 39). The sequence of these paths is the second hidden variable. Because of the likely misrecognitions made by the ASR, we would like to make parsing error-tolerant and allow for a certain degree of mismatch between the instantiated fragment path q and the part s of the ASR-output it accounts for. The segmentation of the ASR-output in these parts is the third hidden variable.

Let $\varPhi$ be a sequence of fragments $\phi_t \in \varSigma^{\mathrm{NE}}$ which constitute some fragment-level representation of A, let Q be the corresponding sequence of instantiated paths (word strings) q_t through these fragments. Also, let S be a segmentation of the ASR-output of A induced by Q. Then, the likelihood of A becomes:

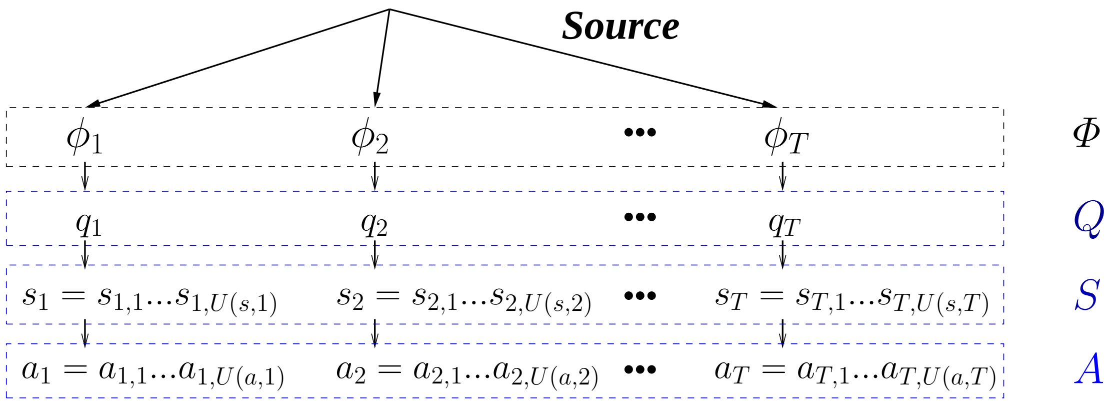

Source
ϕ_1
ϕ_2
ϕ_T
Φ
q_1
q_2
q_T
Q
$s_1 = s_{1,1}...s_{1,U(s,1)}$
$s_2 = s_{2,1}...s_{2,U(s,2)}$
$s_T = s_{T,1}...s_{T,U(s,T)}$
S
$a_1 = a_{1,1}...a_{1,U(a,1)}$
$a_2 = a_{2,1}...a_{2,U(a,2)}$
$a_T = a_{T,1}...a_{T,U(a,T)}$
A

Figure 4.4: Generative production mechanism for utterance audio representation.

$$P(A|\,\Sigma^{\mathrm{NE}}) = \sum_{\Phi,Q,S} P(\Phi, Q, S, A|\,\Sigma^{\mathrm{NE}}). \tag{4.4}$$

Maximum-likelihood parsing aims at finding such combination of fragment and path sequences and an ASR-output segmentation that has the maximum joint likelihood:

$$(\Phi^*, Q^*, S^*) = \operatorname*{argmax}_{\Phi,Q,S} P(\Phi, Q, S, A|\,\Sigma^{\mathrm{NE}}). \tag{4.5}$$

To solve this optimization task, we employ the *generative production model* from Figure 4.4 (also confer [Lev04] and [Hac01]): according to this sequential approach, we have a source emitting sequences of fragments ϕ_t. For each fragment there is an appropriate distribution of its valid paths, and each of the latter can be realized by various strings of words in the ASR-output that we call *surface forms*. In particular, if q_t is the instantiated path through ϕ_t, its realization in the ASR-output is a word string $s_t = s_{t,1} \ldots s_{t,U(s,t)}$, where each segment s_t also reflects an interval a_t of acoustic observations $a_{t,1} \ldots a_{t,U(a,t)}$ from the input audio signal A. Together these segments constitute a segmentation S of the ASR-output.

This model allows for the causality-based dependency restrictions and thus leads to the following chain-rule decomposition of (4.5) to obtain the maximum-likelihood path (Φ^*, Q^*, S^*)[5]:

$$(\Phi^*, Q^*, S^*) = \operatorname*{argmax}_{\Phi,Q,S} \prod_t \underbrace{P(\phi_t|\phi_{t-n_{\max}+1} \ldots \phi_{t-1})}_{\text{language model}} \times \underbrace{P(q_t|\phi_t)}_{\text{NE-grammar}} \times \underbrace{P(s_t|q_t)}_{\text{misrecognitions}} \times \underbrace{P(a_t|s_t)}_{\text{acoustic score}}. \tag{4.6}$$

[5]We will drop the conditioning on the fragment lexicon in the subsequent formulae for the sake of clarity.

In the style of [Pie95], this decomposition can be interpreted as a product of semantic (probability of meaning), primitive syntactic (probability of words given meaning) and acoustic (probability of acoustic observations given words) probabilistic components. In (4.6) we have a *parsing language model* of order $n_{\max}$ that governs the distribution of fragment sequences. In our experiments we tried out $n_{\max} = 1, 2, 3$ (i.e. uni- to trigrams). The exact description of how language models were estimated is given in Section 4.3.3. For a weighted NE-grammar represented by fragment ϕ, the probability of a particular terminal representation q is an intrinsic characteristic of that grammar and is derived from the costs of its rules that must be applied in order to obtain q. The rule cost in handcrafted grammars usually reflects the expert knowledge as to how typical are the terminal representations this rule leads to, for the given named entity. If the grammar is not weighted (as it is in our case), probabilities of all its legitimate sentences are assumed equal. The probability of distortion $P(s|q)$ due to misrecognitions describes characteristics of the employed recognizer. In Section 4.3.4 we will present one way of robust estimation of these probabilities. Finally, the acoustic likelihood $P(a|s)$ of a word string s reflecting how well this string models the portion a of the acoustic signal it is covering, can be obtained directly from the weighted ASR-output (best path or word lattice alike).

As an example, consider the recognized utterance from page 60 again. Applied to this utterance, decomposition (4.6) will contain (among other terms) the probability of the monetary-expression named entity following *"I got this bill for"*, the probability of *"two dollars"* as a realization of this named entity, the probability of *"too dollars"* recognized instead of this realization, and also acoustic score of the corresponding part of the signal.

Let us now take a detailed look at some of the terms in (4.6).

4.3.3 Estimating Language Model for Parsing

From (4.6) it could be seen that named entities can be embedded in parsing language models of higher order with other words. If training transcriptions are available, this language model can be estimated from them by replacing all occurrences of the named entity by a special word and counting of word n-grams in the resulting corpus. If, however, the only kind of information available for training are oracle indications as to the presence or absence of the named entities in each utterance, ad hoc alternative solutions must be found.

So, to estimate a simple unigram language model, a straightforward maximum-likelihood

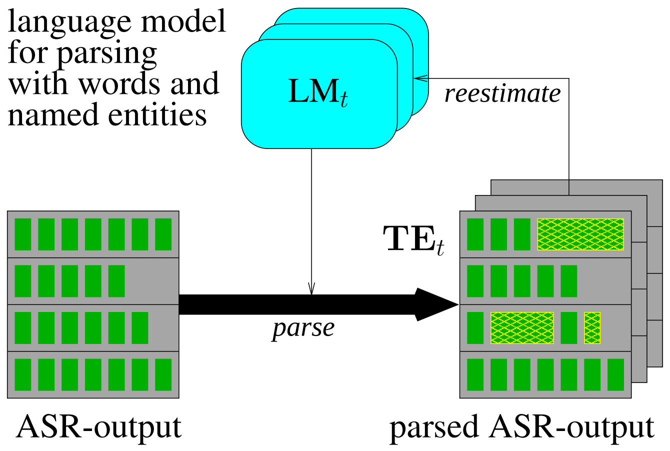


Figure 4.5: Iterative estimation of language model for parsing.

estimation can be used. For all fragments (words and named entities) from (4.3) we compute:

$$P(\phi) := \frac{\#_{\text{train}}\phi}{\sum_j \#_{\text{train}}\phi^j}, \tag{4.7}$$

with fragment counts $\#_{\text{train}}\phi^j$ obtained from the ASR-output of the training corpus and available NE-statistics.

It is also possible to increase the order of the language model, by performing a series of iterations, consisting of consecutive maximum-likelihood parsing (4.6) and estimation of a new parsing language model. This mechanism is graphically explained in Figure 4.5.

4.3.4 Approximate Matching for Maximum-likelihood Parsing

In this section we will focus on the mechanism of approximate matching which is the main reason why (4.6) is useful in practical applications. In short, *approximate matching* means that the probability term $P(s|q)$ from (4.6) is different from the Kronecker delta function which is 1 if $s = q$ and 0 otherwise. While we intend to apply the approximate matching to model the distortions that named entity fragment paths undergo on their way to become recognized NE-instances, for the derivation of the approximate matching algorithms the nature of q and s is irrelevant, and one can operate in terms of arbitrary "intended" and "recognized" word strings q and s. Also, since the algorithms described here are designed to work with words *or* subword units (such as phones) as basic recognition units, henceforth, we will be speaking of abstract *symbols* (and strings thereof).

Suppose now that the user had the symbol string q in mind. What is the probability that in the distorted ASR-output q will be observed as s? The brute force solution is to observe all occurrences of q in the transcriptions of some validation data set and see how they are represented in the corresponding ASR-output [Béc04]. For each q, this effectively implies estimating a distribution $P(s|q)$ over all possible strings s.

Two factors speak out against this methodology in our case:

1. remember that one characteristic property of the semantic attributes we are working with in this thesis is that only a small percentage of their possible values is actually present in the corpus, mostly because they consist of several words and have high entropy. This means that we can not count on a sufficient number of validation examples for each string q to estimate the corresponding distribution $P(s|q)$;

2. in Section 2.4 the main thrust of this thesis was directed at the spoken language understanding without transcribed data from the domain of interest. This means that the distribution $P(s|q)$ will have to be estimated from out-of-domain data. We can indeed expect a considerable portion of the lexicon of the domain of interest to be present in the background language model[6]. At the same time, the proportion of longer strings q that are also present in the out-of-domain data is rather humble, since longer strings tend to carry much more domain-specific information (an observation that holds for words and subword units alike) [Gor99].

Therefore, we decided not to work with holistic distributions of possible string distortions, but to devise a mechanism which would allow on-the-fly probability computation for some string s as a possible distortion of the given string q based on the confusion probabilities of individual symbols in these strings. In other words, we think of string distortions as derived from a symbol-level *noisy channel*, a model where statistical dependencies of the output on the input are uniquely determined by a conditional symbol-wise distribution $P(\mathrm{output}|\mathrm{input})$. This approach has a lot in common with the *string edit distance* paradigm [Bah75, Hal80, Ris98], where the distance (usually expressed as *cost* which is "-log"-related to probability) is computed as aggregation of costs of *elementary edit operations* (identities, insertions, deletions and substitutions). In our noisy channel model, we also consider four types of primitive symbol-wise distortions which we call *mappings*:

[6]In our experiments around 80% of all words that occurred in the test transcriptions were present in the adapted language model.

1. *identity mapping*: the input symbol v is correctly forwarded to the output (noise-free transmission);

2. *substitution*: symbol v is absorbed from the input stream and symbol $w \neq v$ is sent to the output stream;

3. *insertion*: given the next symbol v coming from the input stream, the channel keeps it there and emits symbol w;

4. *deletion*: the next symbol from the input stream v is absorbed without resulting in any output symbol.

If we introduce the *empty symbol* ε, all four mappings can be expressed in a uniform manner $v \rightsquigarrow w$, where $P(\varepsilon \rightsquigarrow \varepsilon) \equiv 0$:

identity:	$v = w \neq \varepsilon;$	insertion:	$v = \varepsilon,\ w \neq \varepsilon;$
substitution:	$v \neq w,\ v \neq \varepsilon,\ w \neq \varepsilon;$	deletion:	$v \neq \varepsilon,\ w = \varepsilon.$

Once we have the probabilities of these mappings, dynamic programming can be used to compute the probability of each string distortion as exemplified in Figure 4.6. In this example, the probability of seeing s as ASR-output where the speaker actually meant q, is computed along the best *distortion path* (the sequence of mappings that transform one string into another) π, i.e. the distortion path with the highest probability:

$$P_\pi(s|q) = P(v_1 \rightsquigarrow w_1)P(\varepsilon \rightsquigarrow w_2)P(v_2 \rightsquigarrow w_3)P(v_3 \rightsquigarrow \varepsilon)P(v_4 \rightsquigarrow \varepsilon). \tag{4.8}$$

Note that we use here the *decision-oriented* definition of the distortion probability, e.g. the probability is computed along the most probable distortion path. The second, "cleaner" alternative consists in adding up distortion probabilities of all possible distortion paths. Later we will use this alternative to train a distortion transducer for named entity localization at the phone level.

The only unresolved question now is how to estimate the probabilities of these mappings? Ideally, we should train distortion probabilities separately for each named entity. This, however, would fail on an insufficient amount of validation material to produce reliable probability estimates, as pointed out earlier. Alternatively, we can create just one model of symbol-wise distortions, not only shared by all named entities, but common for the entire corpus. Indeed, if our goal is to learn possible misrecognitions in *"May first"* as in "05.01.2003", it is as helpful to look at misrecognitions of this string in sentences like *"I may first consider other options"* instead of

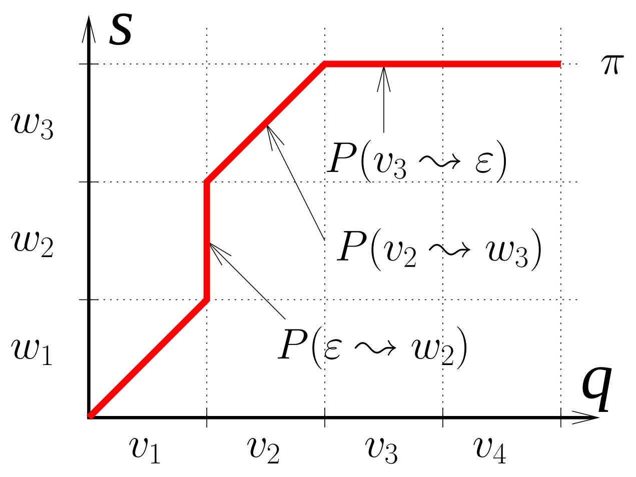

s
q
π
w_3
w_2
w_1
v_1
v_2
v_3
v_4
$P(v_3 \rightsquigarrow \varepsilon)$
$P(v_2 \rightsquigarrow w_3)$
$P(\varepsilon \rightsquigarrow w_2)$

Figure 4.6: Using dynamic programming to compute string distortion probability; in this example, string $q = v_1v_2v_3v_4$ is transformed symbol after symbol into string $s = w_1w_2w_3$.

waiting till a named entity with this exact date turns up. In other words, we can consult our general knowledge about the employed ASR to predict the errors that can occur in the NE-instances. This recognizer-dependent common distortion probability model will be then trained from a pair of two tied representations of the entire out-of-domain validation corpus: manual transcriptions and ASR-output. It is important to understand that we can resort to the out-of-domain data, because our goal here is not to conduct just another adaptation of the language model, but rather to account for acoustic similarity (and therefore confusability) of particular words (or phones) in the language.

The misrecognition statistics can be obtained at different levels of complexity. In [Pal99] word confidence score generated by an ASR for each word v is utilized to estimate the likelihood that the word will be recognized correctly. The probability of a misrecognition of this word is then integrated into the language model. Often however, we have enough data to estimate probabilities of more specific errors of the form: $v \rightsquigarrow w$. Statistics of this kind are easy to acquire: they are already contained in a confusion matrix, which can be computed using Viterbi alignment of manual transcriptions and ASR-output. Further specialization can be achieved by taking immediate left and right mapping contexts into account: $v \rightsquigarrow w|v^{-}\_v^{+}$ (e.g. the probability of misrecognizing v as w when v is surrounded by v^{-} and v^{+} in the input string). Even though context-dependent methods deliver better models for ASR-behavior [Lev03], they raise dramatically the complexity of the algorithms and increase the amount of validation data needed to produce reliable probability estimates.

It is helpful to notice that by estimating $P(s|q)$ in the way presented above, we managed to separate the statistic methods of compensation for recognition errors from the semantics of the task. Semantics are articulated in the handcrafted (application-dependent) named entity fragments, while the variability of the surface forms of these fragments is determined by the properties of the noisy channel that models the employed recognizer.

Word-based Strategy

We decided to use different strategies for word- and phone-level understanding systems. Let us start with words first. Here, we disregard the mapping contexts, but rather estimate distributions $P(v \rightsquigarrow w)$ from a simple confusion matrix.

For each deletion, insertion and substitution[7] $v \rightsquigarrow w$, a typical confusion matrix contains the number of its occurrences in the aligned pair of manual transcriptions and ASR-output of the validation corpus: $\#(v \rightsquigarrow w)$. These counts can be rewritten as probabilities based on the following stochastic conditions for each nonempty left hand side v:

$$\sum_{w} \left(P(v \rightsquigarrow w) + P(\varepsilon \rightsquigarrow w)\right) \equiv 1. \tag{4.9}$$

With the input lexicon $\Sigma^I = \{v^i\}$ and output lexicon $\Sigma^O = \{w^j\}$ (both including the empty symbol ε), this leads to the normalization formulae for substitutions and deletions of v^k ($v^k \neq \varepsilon$):

$$P(v^k \rightsquigarrow w^l) := \frac{\sum_{i,j:v^i\neq\varepsilon} \#(v^i \rightsquigarrow w^j)}{\sum_{i,j} \#(v^i \rightsquigarrow w^j)} \times \frac{\#(v^k \rightsquigarrow w^l)}{\sum_j \#(v^k \rightsquigarrow w^j)}, \tag{4.10}$$

and for insertions of w^l ($w^l \neq \varepsilon$):

$$P(\varepsilon \rightsquigarrow w^l) := \frac{\#(\varepsilon \rightsquigarrow w^l)}{\sum_{i,j} \#(v^i \rightsquigarrow w^j)}. \tag{4.11}$$

The first multiplicand in (4.10) accounts for the probability of not making an insertion when word v^k is the next to read from the input channel (undistorted word transcriptions). Since we assume context independency of all mappings, this probability does not depend on v^k and therefore can be approximated as the proportion of observed non-insertions to all observed mappings. The second multiplicand expresses the probability that of all possible mappings of v^k the one to w^l will be chosen. The estimation of insertion probabilities is also done context-independently and is rather straightforward.

[7]Since substitutions and identities are of the same nature, we don't have to distinguish between them explicitely.

Phone-based Strategy

From the literature we know that recognition of individual phones is significantly affected by their local neighborhood in the acoustic stream [Sch85]. Unlike word-based strategy where the number of possible context-dependent mappings becomes prohibitively high even when only immediate right and left contexts are considered[8] and thus only selective context-modeling is feasible, the rather restricted phone inventory (usually about 50) easily allows for an exhaustive modeling of immediate context. While in the context-independent case we considered mappings of the form $v \rightsquigarrow w$, with context modeling we are looking at mappings $v \rightsquigarrow w/v^-\_v^+$ with $v, v^-, v^+ \in \Sigma^I,\ w \in \Sigma^O$, e.g. recognizing v as w when in the original string v was surrounded by v^- and v^+. The idea is again to investigate what phones are acoustically confusable in which context and to compensate for the most typical confusions.

Up to now we considered mapping probabilities without explicitely mentioning their conditional part. However, at least by substitutions and deletions these probabilities were always conditioned on the next symbol in the input stream: $P(v \rightsquigarrow w|v)$. After introduction of context the conditional part will include this context as well. Let the undistorted input phone string be $q = v_0v_1\ldots v_{T-1}v_T$ with all v's from Σ^I and the current position in this string (i.e. the index of the next phone to read) t. The next mapping will be either substitution or deletion of v_t in the context $v_{t-1}\_v_{t+1}$ or insertion in context $v_t\_v_{t+1}$. Thus, at each point in time we not only have competing mappings in the same context, but also competing contexts, so that at each step the following must hold:

$$\sum_j P(v_t \rightsquigarrow w^j/v_{t-1}\_v_{t+1}|v_{t-1}v_tv_{t+1}) + \sum_{j:w^j\neq\varepsilon} P(\varepsilon \rightsquigarrow w^j/v_t\_v_{t+1}|v_{t-1}v_tv_{t+1}) \equiv 1.0;\ \forall v_{t-1}, v_t, v_{t+1}. \tag{4.12}$$

The problem with this formula is that we wish to circumvent maintaining statistics for insertions conditioned on two phones in the left context (as opposed to only one in $P(\varepsilon \rightsquigarrow w^j/v_t\_v_{t+1}|v_tv_{t+1})$). Let $\bar{P}(\mathrm{sd}(v_t)/v_{t-1}\_v_{t+1}|v_{t-1}v_tv_{t+1})$ denote the probability of making insertion in context $v_t\_v_{t+1}$ (i.e. not a substitution or deletion of v_t in context $v_{t-1}\_v_{t+1}$), given $v_{t-1}v_tv_{t+1}$, and $\bar{P}(\mathrm{ins}\,/v_t\_v_{t+1}|v_tv_{t+1})$ the probability of doing substitutions or deletions of v_t in context y_v_{t+1} with an arbitrary $y \in \Sigma^I$ (i.e. not an insertion in context $v_t\_v_{t+1}$), given v_tv_{t+1}. Then, we can split (4.12) in two:

$$\left\{ \begin{array}{lll} \sum_j P(v_t \rightsquigarrow w^j/v_{t-1}\_v_{t+1}|v_{t-1}v_tv_{t+1}) + \bar{P}(\mathrm{sd}(v_t)/v_{t-1}\_v_{t+1}|v_{t-1}v_tv_{t+1}) & \equiv & 1.0; \\ \sum_{j:w^j\neq\varepsilon} P(\varepsilon \rightsquigarrow w^j/v_t\_v_{t+1}|v_tv_{t+1}) + \bar{P}(\mathrm{ins}\,/v_t\_v_{t+1}|v_tv_{t+1}) & \equiv & 1.0. \end{array} \right. \tag{4.13}$$

[8]With 10000 words in the lexicon, there are 10^{16} of them ($O(|\Sigma|^4)$ complexity).

Again, we want to estimate context-dependent mapping probabilities from (4.13), so that for the out-of-domain validation corpus the likelihood of the distortion of its transcriptional representation into an ASR-output (both being sequences of phones) becomes maximal. We use the *Expectation Maximization (EM) algorithm* [Dem77] to achieve this goal. Each iteration of the EM-algorithm consists of an *expectation step* and a *maximization step*.

The expectation step for $q = v_0 v_1 \ldots v_{T-1} v_T$ and $s = w_0 w_1 \ldots w_{U-1} w_U$ is presented in Figure 4.7. Having previous estimates of the probabilities of the phone mappings in context (all probabilities are assumed equal in the beginning), we first recursively compute forward and backward probabilities $\alpha_{t,\tau}$ and $\beta_{t,\tau}$ [FJ96] and then update counters γ of these mappings.

During the maximization step, the probabilities of all phone mappings are re-estimated:

$$P(v \rightsquigarrow w/v^-\_v^+|v^-vv^+) := \frac{\gamma(v \rightsquigarrow w/v^-\_v^+|v^-vv^+)}{\sum_y \gamma(v \rightsquigarrow y/v^-\_v^+|v^-vv^+) + \bar{\gamma}(\mathrm{sd}(v)/v^-\_v^+|v^-vv^+)}; \quad (4.14)$$

$$P(\varepsilon \rightsquigarrow w/v^-\_v^+|v^-v^+) := \frac{\gamma(\varepsilon \rightsquigarrow w/v^-\_v^+|v^-v^+)}{\sum_y \gamma(\varepsilon \rightsquigarrow y/v^-\_v^+|v^-v^+) + \bar{\gamma}(\mathrm{ins}\,/v^-\_v^+|v^-v^+)}. \quad (4.15)$$

Given the number of phones in our dictionary[9] $|\Sigma|$, we have to estimate $O(|\Sigma|^3)$ different context-dependent probability distributions (or $O(|\Sigma|^4)$ context-dependent probabilities). Even with a moderate number of phones used (in our experiment $|\Sigma| = 43$), the amount of data practically available is not sufficient to estimate all context-dependent probabilities reliably. To alleviate this problem, we can interpolate among contexts with different degrees of specificity. Thus, the probability of the phone mappings: $v \rightsquigarrow w/v^-\_v^+$ can be computed as a linear combination:

$$\begin{aligned} P(v \rightsquigarrow w/v^-\_v^+) &= \xi_f P(v \rightsquigarrow w/v^-\_v^+|v^-vv^+) + \xi_l P(v \rightsquigarrow w/v^-\_*|v^-v) \\ &+ \xi_r P(v \rightsquigarrow w/*\_v^+|vv^+) + \xi_n P(v \rightsquigarrow w/*\_*|v) + \xi_z \frac{1}{|\Sigma|}, \end{aligned} \quad (4.16)$$

with $\xi_f + \xi_l + \xi_r + \xi_n + \xi_z \equiv 1.0$ and wildcard "*" standing for any symbol. In this case, before interpolation can be done, four optimization processes must be performed in parallel. Other smoothing technics (like, for instance, *back-up*) are also conceivable.

[9]Without loss of generality we can assume $\Sigma^I = \Sigma^O = \Sigma$.

```
INPUT:
probabilities of mappings in contexts estimated from the previous iteration
$\alpha_{-1,-1} := 1$
FOR $t = 0 \dots T,\ \tau = 0 \dots U$
    $\alpha_{t,\tau} := 0$
    IF   defined $\alpha_{t-1,\tau-1}$
    THEN $\alpha_{t,\tau} \mathrel{+}= \alpha_{t-1,\tau-1} * P(v_t \leadsto w_\tau / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1})$
    IF   defined $\alpha_{t-1,\tau}$
    THEN $\alpha_{t,\tau} \mathrel{+}= \alpha_{t-1,\tau} * P(v_t \leadsto \varepsilon / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1})$
    IF   defined $\alpha_{t,\tau-1}$
    THEN $\alpha_{t,\tau} \mathrel{+}= \alpha_{t,\tau-1} * P(\varepsilon \leadsto w_\tau / v_t\_v_{t+1} | v_tv_{t+1})$
Estimate backward probabilities $\beta_{t,\tau}$ in an analogous way
FORALL $v, v^-, v^+ \in \Sigma^I, w \in \Sigma^O : \neg(v = w = \varepsilon)$
    $\gamma(v \leadsto w / v^-\_v^+ | v^-vv^+) := 0$
    IF   $v \neq \varepsilon$
    THEN $\bar{\gamma}(\mathrm{sd}(v) / v^-\_v^+ | v^-vv^+) := 0$
    ELSE $\bar{\gamma}(\mathrm{ins} / v^-\_v^+ | v^-v^+) := 0$
FOR $t = 0 \dots T,\ \tau = 0 \dots U$
    $\gamma_s := \alpha_{t-1,\tau-1} P(v_t \leadsto w_\tau / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1}) \beta_{t,\tau} / \alpha_{T,U}$
    $\gamma_d := \alpha_{t-1,\tau} P(v_t \leadsto \varepsilon / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1}) \beta_{t,\tau} / \alpha_{T,U}$
    $\gamma_i := \alpha_{t,\tau-1} P(\varepsilon \leadsto w_\tau / v_t\_v_{t+1} | v_tv_{t+1}) \beta_{t,\tau} / \alpha_{T,U}$
    $\gamma(v_t \leadsto w_\tau / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1}) \mathrel{+}= \gamma_s$
    $\gamma(v_t \leadsto \varepsilon / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1}) \mathrel{+}= \gamma_d$
    $\gamma(\varepsilon \leadsto w_\tau / v_t\_v_{t+1} | v_tv_{t+1}) \mathrel{+}= \gamma_i$
    $\bar{\gamma}(\mathrm{sd}(v_t) / v_{t-1}\_v_{t+1} | v_{t-1}v_tv_{t+1}) \mathrel{+}= \gamma_i$
    $\bar{\gamma}(\mathrm{ins} / v_t\_v_{t+1} | v_tv_{t+1}) \mathrel{+}= \gamma_s + \gamma_d$
OUTPUT:
updated mapping counters
```

Figure 4.7: Expectation step of EM-algorithm for estimation of context-dependent phone mapping probabilities; we assume $v_t = \varepsilon\ \forall t < 0, t > T$. Operator " $+=$ " increments its left operand by the value of its right operand.

4.3.5 Expressing Parsing as FSM-operations

We have shown how individual components of maximum-likelihood estimation formula (4.6) can be interpreted and computed for unsupervised word- and phone-based NE-localization. To put these components together and carry out the search for the maximum-likelihood path (Φ^*, Q^*, S^*) we decided to use *Finite State Machines (FSM)*. Under FSMs one usually understands finite state automata, augmented by a collection of operators defined on them.

Before we explain some of the FSM-supported operations for finite state automata, let us bring in *transducers*, whose only difference from *acceptors* introduced in Section 4.3.1 is that they not only absorb input symbols, but also emit output symbols from a distinguished lexicon Σ^O [Hop79]. This additional feature renders weighted finite state transducers perfectly geared up to implement the noisy channel paradigm that we employ to model approximate matching. The usability of finite state transducers for spoken language understanding is emphasized in [Moh97]. Unlike acceptors which can only make an assertion about grammaticality of a sentence (or the degree of its grammaticality) in the language they represent, transducers actually translate the acceptable sentence in their output language. Each acceptor can be understood as a degenerated transducer, identically translating each acceptable sentence in itself.

Each multiplicand in (4.6) can be represented as a finite state transducer. There are many operations that can be performed on FSMs [FSM]. At this point we are going to explain some of the operations that are essential to understand the application of FSMs to the process of finding the optimal fragment sequence Φ^*.

Union of two or more FSMs is another FSM that recognizes (or translates) the languages of all of them (parallel connection). For example, a simple unigram language model for parsing with fragment probabilities defined in (4.7) that we have used in the majority of our experiments, can be expressed as a union transducer $\mathcal{LG}$, extended by a **Kleene closure** (arcs from all its final states to the initial state). The components of this union are FSMs representing all entries from the lexicon (4.3): words and one named entity fragment[10]. In general, following the strategy from Section 4.3.3, the n-gram language model FSM $\mathcal{LG}$ can be estimated from a corpus in which all occurrences of the named entity are replaced by a special symbol. In this FSM we then substitute all arcs labeled with this symbol by the corresponding named entity fragment FSM.

Composition is an FSM-operation on two transducers $\mathcal{F}_1$ and $\mathcal{F}_2$ which creates a new transducer $\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2$. For all sentences that can be translated by $\mathcal{F}_1$ and whose $\mathcal{F}_1$-translation can be translated by $\mathcal{F}_2$ the resultant transducer $\mathcal{F}$ outputs this latter translation. In the case of weighted transducers, where each arc is provided with individual cost (usually computed according to the –logprob scheme), the arc costs of the resultant transducer are computed as a sum of costs of corresponding arcs of $\mathcal{F}_1$ and $\mathcal{F}_2$ (see example in Figure 4.8).

Let us denote $\mathcal{D}$ the *distortion transducer* that accounts for approximate match (Section 4.3.4). Let also $\mathcal{S}$ be the FSM-representation of the ASR-output S. Note that $\mathcal{S}$ is not restricted to linear sequences of words (phones) S. In fact, word (and phone-) lattices are finite state acceptors par excellence, so that this transducer can also represent ASR-lattices with recognition

[10]In fact, for parsing in our experiments, we considered all named entity fragments in parallel.

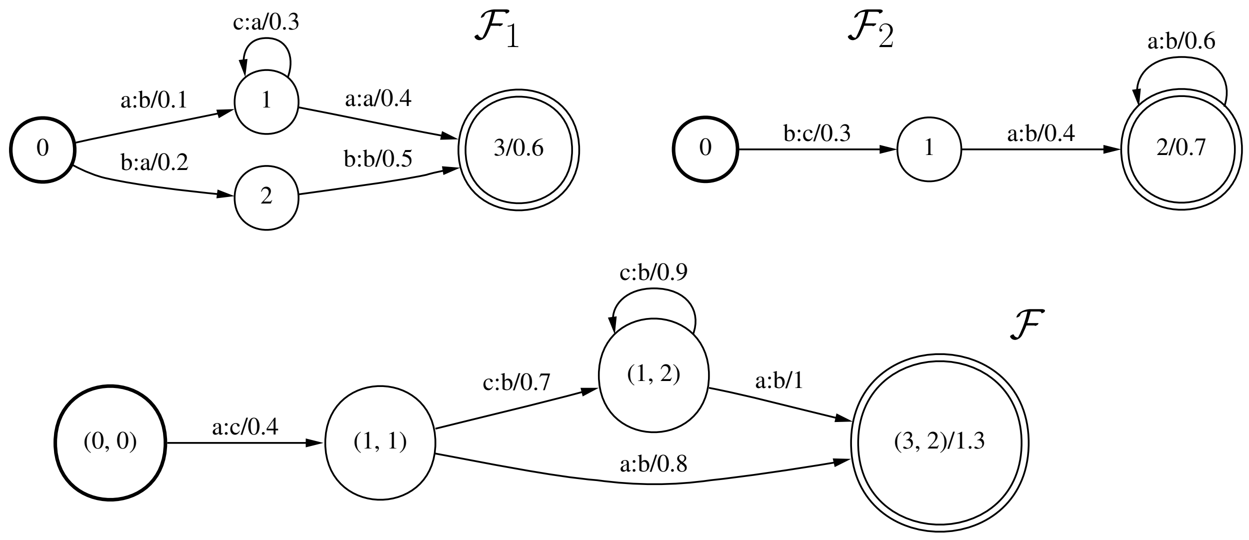


Figure 4.8: Example for FSM-composition of two weighted transducers: $\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2$ (after [Moh02]).

scores. Now, transducer $\mathcal{F}$ can be obtained as a composition:

$$\mathcal{F} = \mathcal{LG} \circ \mathcal{D} \circ \mathcal{S}. \tag{4.17}$$

It contains all possible parses of the ASR-output with named entity fragments and words from the lexicon Σ^{NE}, each parse being represented by a single path through the finite automaton $\mathcal{F}$. The cost of a parse can be computed as a cumulative of the costs for all arcs $\mathcal{F}$ that constitute the corresponding path. To find the maximum-likelihood parse, we need to select the path with the lowest cost. This is accomplished by a general shortest-distance algorithm, and the corresponding FSM-operation **bestpath**.

All experiments on which we report in this thesis were conducted using the AT&T FSM toolkit, which is publicly available for non-commercial use from [FSM]. Estimation of the parsing language model $\mathcal{LG}$ of higher order was done using the GRM library (Grammar Library) [GRM] operating on the top of the FSM toolkit.

4.3.6 Using Detection Score for Localization

In Section 4.2 we illustrated the importance of the syntactic context of named entities for the task of their detection. Likewise, NE-localization can profit substantially from context-based indicators as to the presence of a NE-instance at a particular location in the utterance. Besides, the context also helps to distinguish goal-oriented context-specific named entities from their look-alike's, as illustrated in the example for named entity ITEM_AMOUNT in HMIHY-system on page 58.

The roles of context for detection and localization differ in that by localization the adjacency is of a greater importance. However, in the cases where NE-priors are relatively low (in our corpus, each named entity occurred in ca. 3% of all utterances), both procedures come down to the same idea: *provided that a given context was (not) found in the utterance, adjust the probability of a NE-instance being present (or present at a specific location) in this utterance correspondingly*. This allows us to take advantage of our separate formulations of the detection and localization tasks. The information contained in the context can be utilized during the detection stage, which — as we pointed out earlier — does not require conjoint modeling of the named entities and their context and is capable of retrieving informational cues about NE-presence from the entire utterance. By taking care of the context at the detection stage, we now can fall back on named entity grammars with all context stripped off at the localization stage. For monetary expressions, for instance, this would mean modeling only *"one dollar and twenty cents"* instead of something like *"[paid] one dollar and twenty cents [for the call]"*. This corroborates our claim of independency of our algorithms from manually collected domain-specific data, since these "context-free" named entity models can be adapted at no expense from other applications (in the end, it doesn't matter if $1.20 is a charge for a phone call or the current Euro exchange rate).

Having detection oracle casting its vote about the necessity of localization is extremely paying from the computational point of view. Our tests showed that invoking localization tool, only when detection was signalized with a satisfying degree of certainty, could reduce the average processing time in a corpus-based test by up to a factor of 3.5, while also improving localization quality.

A remark concerning the robustness of this algorithm must be made. It is not guaranteed that even the best detection oracle will help avoid confusions by context-specific named entities in cases where one utterance contains instances of them *and* of their look-alike's. Even though, our experiments showed that cases like this are rare, one can still evade this shortcoming by splitting utterances in parts and performing detection followed by localization for each part independently.

4.4 Named Entity Value Extraction

4.4.1 Justification of the Task

Successful extraction of semantic attribute values is of crucial importance for the language understanding task. The automated dialog systems for call centers need the ability not only to localize the named entities in the signal but also to analyze their meaning. Whenever the system suspects that there is an instance of a named entity in the signal, it should be able to:

1. make sure the extracted named entity expression is syntactically correct.
   For example, *"five dollar and thirty forty cents"* is not a correct monetary expression;

2. normalize the entity into an unambiguous representation (if possible).
   For example, the following expression pairs are effectively synonymic:

USA	and	*United States of America*
twelve hundred dollars	and	*one thousand two hundred dollars*
phone number 123 4567890	and	*phone number 1 123 4567890*

3. check the semantic legitimacy of the extracted value.
   For example, the normalized date value *02.29.2003* is semantically unacceptable even though its underlying expression *"February twenty ninth two thousand three"* is syntactically correct;

4. interpret and use the extracted value for internal operations, like retrieval of a data subset that contains named entities with the given value.

These expectations allow us to talk about *named entity understanding* as opposed to simple *recognition* (also called *identification* in the NE-literature) which usually stops after the first item. As yet, the differences between named entity understanding and named entity identification have not been articulated strong enough in the literature. We ascribe this lack of attention largely to the fact that for many applications, there is no need for elaborated understanding mechanisms once the named entity has been found, because its surface representation, i.e. its wording, is considered equivalent to its meaning. This is usually the case, when there is only one way (or just a few ways) to express a semantic attribute like, for instance, color, but is far from true for dates, monetary expressions and others.

This is why the situation turns more complicated for automated dialog systems such as HMIHY, where all of the steps listed above must be executed in order to proceed with the dialog. Here, all information obtained from the customer through the dialog must be verified with the internal databases. If the customer mentions a call he made on February fifteenth to Dallas, then there must be a corresponding entry in the logs of the connections managed by the company.

The presence of this verification step ties up to the central difference between practical dialog systems and language learning systems[11]. Learning systems pursue the objective of new knowledge acquisition [Sik72, Roy99]. So, for instance, upon the conclusion that at a specific location in the utterance there seems to be a name of some previously unknown article of furniture, a simple learning system [Gor94b] would copy the signal from the corresponding interval and — if its acoustic representation is different from what it has learned before — accept it as a new item in the furniture lexicon. On the contrary, a dialog system is rather rigid concerning the extension of the existing knowledge base. Its main goal is to formulate a request to the preloaded database, which means that it has to lay the ground for search actions based on semantic comparison.

Even though we can talk of building classes of equivalent NE-values, it doesn't make sense to shape one class per normalized value and run a classifier which, when presented with a named entity wording, would produce this value, as we were practicing with the calltypes. The reason is an enormous number of classes this classifier would have to handle. To arrive at the NE-value starting from its word representation, one would rather need a parsing mechanism. The parser must accommodate our semantic knowledge about the normalized named entity expressions and is to determine how these expressions are generally put into words. Consider the following example of an expression for a phone number in US: *"one eight hundred one two three fourteen twelve"*. The expression is ambiguous in the sense that it can be interpreted in at least two alternative ways: *1 800 1 2 3 14 12* and *1 801 2 3 14 12*. However, the second expression can not possibly be accepted, because we know that if a phone number contains ten digits (as is the case for the second interpretation) its three first digits have to be grouped together to form an area code. For similar reasons we would have to dismiss other parsing alternatives *1 8 101 2 3 14 12* and *1 8 100 1 2 3 14 12*.

4.4.2 Value Extraction with FSM-transducers

While it can be difficult for a statistic learning system to acquire pragmatic knowledge of this kind, all these nuances can be easily accounted for by a parser that employs a manually hand-

[11] See also relevant polemic about the basic principles of human understanding nature in [Ger90, Rie94].

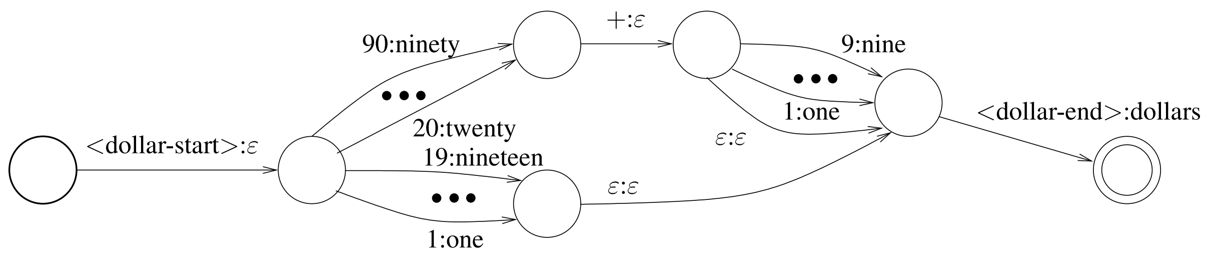


Figure 4.9: Example of an unweighted transducer for simple monetary expressions.

crafted grammar. The desired pre-condition, though, is that the grammar is unambiguous for unambiguous NE-expressions (that is, it provides at most one parse tree for these expressions).

In fact, it makes sense to use the same named entity fragment to localize a named entity in the utterance and to make the first step towards extraction of its value. Then, already during the stage of localization, we will be looking only for instances with internal structure that is semantically and syntactically coherent (also confer similar attempts to unite syntax and semantics in a grammar for time expressions in [Ehr90]). Thus, our acceptor from Figure 4.3 can be modified into the transducer from Figure 4.9 (here, we also incorporate the special markers that will delimit the instantiated named entity in the parsed ASR-output). When used as a part of the parsing language model $\mathcal{LG}$, for each legitimate surface form on the output side (e.g. *"sixty two dollars"*) it contains its symbolic representation on the input side (*"<dollar-start> 60 + 2 <dollar-end>"*). The only step left is to parse this intermediate representation to something like *"62.00"*, which is a straightforward task given a carefully designed fragment grammar. Similarly, the extracted NE-wording *"... on November twenty second of nineteen ninety nine"*, will result in the intermediate representation: *"<month-start> November <month-end> <day-start> 20 + 2 <day-end> <year-start> 19 * 100 + 90 + 9 <year-end>"*, before we can finally assemble *"11.21.1999"*.

The incorporation of the distortion FSM $\mathcal{D}$ estimated as suggested in Section 4.3.4 in the maximum-likelihood parsing scheme will preserve this intermediate NE-representation, while merely extending the scope of its acceptable surface forms, so that the composed transducer $\mathcal{LG} \circ \mathcal{D}$ will become error-tolerant and might, for example, recognize the same dollar amount as in the first example above even when presented with *"sixty too dollars"* on the output side. Summarizing, the exact procedure of named entity value extraction from each utterance is the following:

1. run a named entity detection classifier on the utterance (Section 4.2);

2a. dismiss the utterance if it is not likely to contain any named entities;

2b. otherwise, parse the utterance with the named entity fragments for those named entity types that are believed to be present there (Section 4.3);

3a. dismiss the utterance if no instances of named entities were discovered;

3b. otherwise, take the intermediate symbolic representation of each encountered NE-instance and normalize it into a final normalized standard representation.

Chapter 5

Acoustic Morphemes

In the previous chapters we have discussed the possibilities of automated spoken language understanding under the special condition of missing manual transcriptions for the domain of interest. As an extreme case of these conditions, we have studied the situation where not even a background word language model was available for the speech data, and we had to resort to the subword-level utterance representation, and to try and extract information useful for calltype classification from sequences of phones obtained from a phone recognizer.

In Section 4.4.2 we drew a line between language understanding and language learning grounding their differences in the comparison between recognizing existing and building new mental structures as a result of communication. Each language understanding system obviously requires some learning as well, however, the acquired salient bits of information by such application-driven learning do not necessarily have to comply with our intuitive perception of the language having words as basic building units. So, the aim of the classification task from Section 3.3.2 was extraction of calltypes and therefore, the choice of salient phone strings there was governed by the objective of achieving the highest classification rates possible. Similarly, the phone-level detection of semantic attributes from Section 4.2 was focusing on those phone strings which correlated with the presence of named entities in the utterances. In any case, the choice of the phone strings to select was largely determined by the semantics of the application, be it information about the calltype, the presence of named entities, or anything else.

In this chapter we will combine the idea of the application-driven learning, with the attempt to recreate the word lexicon from a continuous phone stream based solely on the intrinsic characteristics of the audio channel. We will start with the overview of today's most popular principles of unsupervised word discovery, considering separately word lexicon with syntax and semantics, followed by our own algorithm which combines syntactic and semantic criteria during the word discovery process.

5.1 Automated Acquisition of Lexicon, Syntax and Semantics

Unlike animals who have only a limited number of *displays*, i.e. primitive rigid patterns of behavior (expressive movements) signalizing their innate conditions or intentions, at their disposal, humans have elaborated a sophisticated mechanism of verbal communication which allows them to use flexible structures of words to express very complicated, richly structured thoughts.

Among five commonly acknowledged features that, from the psychological perspective, distinguish human language, are its *structurality* and *meaningfulness* [Gle91, pages 333ff]. At the linguistic level these qualities have their pendants in lexicon (with syntax) and semantics. Words are the basic syntactic elements of our language. Along with the rules of their combination in larger syntactic units, phrases, they lay foundation for the large expressive power of human languages. In [Pin99] it was shown that the relations among morphological structures inside words obey the same principles of combining non-divisible entries of a morphological lexicon using a number of rules. Even though syntax and morphology are clearly of different linguistic nature [Pin99, page 27ff], from the information-theoretical perspective, their constraints can be attributed to one common principle of merging neighbors that often occur together. Furthermore, that same principle will hold also for individual phones making up morphemes themselves. Starting at the very bottom, i.e. with the continuous phone streams, it is very difficult to draw a distinct line between these three processes [dM96]. Thus, even though in the later sections we will be talking about using the information-theoretical principle above to derive lexicon, to some extent, the results of its application will incorporate acquisition of syntax as well.

All the combinatorial diversity of words combined into phrases would remain pointless if not for the *meaning* associated with each of them. The Merriam-Webster dictionary defines word as *"a speech sound... that... communicates a meaning..."* [Mer].

The nomenclature of "meaning" itself requires some effort. In this work, we avail ourselves of a rather simplistic definition:

Definition 5.1 (Meaning) Meaning *of a word (or a word sequence) is a distinctive, consistent and stable over time association of this word (or this word sequence) with objects or facts of the real world that can also be perceived and recorded through other communication channels.*

Thus, we see that the role of words in a language is also distinguished by their property of being cornerstones of semantics.

Summarizing, it seems reasonable to distinguish three typical word characteristics that could be used as word-defining criteria to infer word lexicon where it is not known in advance, and only a continuous stream of acoustic observation paired with some semantic channel is accessible:

- *lexical consistency*: each word is likely to have a consistent internal structure (sequence of aligned phones);
- *syntactic consistency:* there's a number of likely contexts each word tends to occur within;
- *semantic consistency:* there's a number of likely meanings that go along with each word.

5.1.1 Lexicon and Syntax from Acoustic Signal

Induction of linguistic structure in a continuous speech signal is a problem that has been known since the sixties. It comprises extraction of lexicon and syntax, and, from the algorithmic point of view, the unsupervised lexicon derivation is the most interesting part, being usually accomplished by segmenting the acoustic stream into words.

To come up with an automated solution for this segmentation task, many researchers first looked at how children tackle it. As it turned out, there are many clues used by children to perform the segmentation. In [Cut91] speech rhythm of the spoken language is suggested as one possible motivator for segmenting decisions. Other types of prosodic information, such as high pitch, slow rate and exaggerated intonations, typical for the so-called *Motherese* language, i.e. the way mothers talk to the infants [Fer89], are also considered relevant for children's learning of phrase components [Mor86].

There is yet another group of segmentation cues, coming from phonemic and phonotactic constraints. Some phones are more likely to occur contiguously within a word than at word edges or across word boundaries [Bre96]: for example, the triphone $/ts\eth/$ can never occur within an English word, but is totally admissible across the word boundary, as in the phrase *"what's this"*. In Figure 5.1 we have plotted the occurrence statistics of phones and triphones observed in the manual transcriptions of the HMIHY-corpus that were translated onto the phone level using dictionary pronunciations. We compared two cases: the one where complete sentences were converted into continuous phone streams, and the other where this conversion was carried out word by word. Consequently, the first set of statistics accounted also for the (tri-)phones at the word boundaries, whereas the second only for the word internal (tri-)phones. The phones and triphones were ordered so as to ensure the non-increasing order for the corresponding statistics for the case of the isolated words. To corroborate the impression gained from the visually

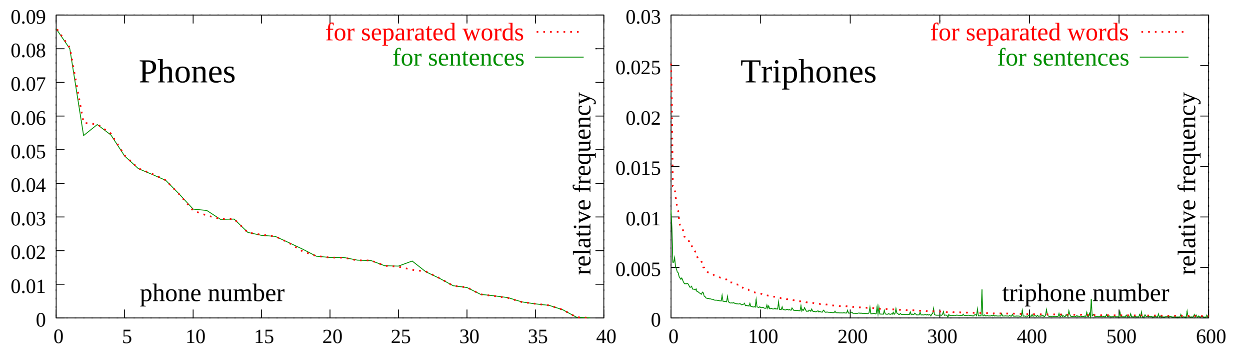


Figure 5.1: Phone and triphone distribution statistics for the transcription corpus when considering isolated words and complete sentences.

perceptible clues, we also computed the cosine measure between the observed distributions, a straightforward estimate as to whether two vectors point in the same direction:

$$D_{\cos}(\boldsymbol{x},\boldsymbol{y}) := \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{|\,\boldsymbol{x}\,||\,\boldsymbol{y}\,|}. \tag{5.1}$$

As expected, we observed that while phone distributions hardly changed when taking the across-word articulation phenomena into account ($D_{\cos}(\text{phones}) \approx 0.9997$), the distribution of triphones flattened a great deal ($D_{\cos}(\text{triphones}) \approx 0.79$) and got a long, spiky tail, due to many word-internally "untypical" triphones that could now be encountered[1]. In compliance with the Zipf's law [Zip35], there were only a few really prominent spikes, one characteristic example being the triphones: *"D l ay"* and *"K T uw"* which were never observed by isolated words and yet appeared to be extremely viable in the continuous case, owing to a high frequency of the word triple *"'d like to"* in the corpus.

Given this effect, the intuition by the automatic segmentation of a continuous phone stream based on the information-theoretical cues, is to look for word boundaries, such that as many of the frequent phone strings as possible wouldn't be broken apart.

The first experiments on word discovery along the information-theoretical line of thought were conducted an written documents. To recreate the lexicon from collapsed text, Olivier in [Oli68] based his iterative algorithm upon the concept of stochastic generative word grammars (the grammars like the one in Figure 4.2 but with respective probabilities for each rule). Every iteration of his algorithm consisted of the following steps:

[1]To make the Figure 5.1 more demonstrative, we clipped off the triphone distributions after the first 600 triphones, although they contained more than 3700/13300 entries in the isolated/continuous case respectively.

1. Maximum-likelihood (ML-) unigram parsing of the corpus, where the most probable sequence of rules that could have generated the corpus was sought. Since Olivier's corpus was represented as a linear sequence of letters, the parsing task was equivalent to determining word boundaries in a phone string;

2. ML-re-estimation of word probabilities in the revised dictionary;

3. application of several heuristics to adjust the dictionary.

The algorithm made no use of any extra-linguistic information and also offered no remedies to compensate for noise which is not an issue for collapsed text but plays a critical role when dealing with speech signal.

In [Ric97] Riccardi also used an iterative algorithm to acquire recurrent *variable length phone sequences* from speech signal. He suggested filtering the corpus on every iteration with the phone strings with high values of *weighted mutual information*: $\text{WMI}(w^1, w^2) = P(w^1 w^2) * I(w^1, w^2)^2$ if these strings also appeared to be entropy-reducing, i.e. if the corpus re-expressed in terms of these strings would possess lower entropy.

In [Del97] Deligne showed how to make use of the *multigram* framework [Del95] to infer lexicon from a speech signal. Her EM-like algorithm employed multigrams as a production mechanism. According to this model, the source emits multigram symbols and the latter give rise to sequences of observations of acoustically derived recognition units, which are then naturally concatenated in a continuous stream. Since one multigram can be responsible for several alternative sequences of recognition units (in the reported study HMM mechanism was integrated as the connection link between hidden multigram symbols and observable acoustic recognition units), the algorithm was capable of handling noise in the signal. Thus, experiments not only on collapsed text but also on acoustic data could be conducted, whereby the recognition based on the multigram units resulted in the phonetic accuracy lying between those of triphones and words.

The algorithm proposed by de Marcken in [dM96] was based upon *Minimum Description Length (MDL)* principle [Ris89] that tries to strike a compromise between the appropriateness of a model with respect to the training data and its size; when applied to a speech signal, it provided two hidden layers in the EM-algorithm with a decision-oriented estimation of word statistics: segmentations of the phone stream and correspondences between phone- and phoneme-sequences[3]. The presence of two hidden layers impaired the time behavior of the algorithm

[2]$I(w^1, w^2)$ denotes mutual information between w^1 and w^2.

[3]To restrain the possible correspondences between phonological and morphological representations of the signal, de Marcken made use of rudimentary knowledge about speech production mechanism.

drastically compared to the process of word extraction from text data. Although some promising results have been achieved, the performance on speech was also much poorer than on text data.

As pointed out earlier, statistical acquisition of syntax from word chains is in its nature similar to deriving a word lexicon from a phone stream. The role of syntax (from the Greek *"arranging together"*) is to impose a set of rules on the language, defining how words in it can be put together to form correct sentences, and what structural role each word plays [All95, page 10]. While the most popular way of thinking about syntax in the post-Chomsky's linguistics is in terms of rules (for instance, that an affirmative sentence in English usually consists of a *noun phrase* followed by a *verb phrase*), at the statistical level its (somewhat rough, but robust) approximation is given in terms of stochastic language models, such as n-grams. Since in our experiments we will start with the so-called *tabula rasa* model (missing word transcriptions), no explicite distinction between lexicon and syntax induction will be made (also see a similar view in [dM96]), but rather a smooth transition between words and syntactic structures facilitated.

5.1.2 Learning Semantics

In the movie "The thirteenth Warrior", the character of Antonio Banderas spends some time surrounded by Vikings who are discussing their affairs in a language totally unknown to the film protagonist. After a short while, he suddenly becomes able to understand and speak their language. Upon being asked, how did he learn the language, he gives a simple answer: *"I listened"*.

In reality, this utopian idea of language learning in absence of any extra-linguistic information, has been abandoned by psychologists a long time ago. Nowadays, it is widely agreed upon that human language understanding implies exploration of the associations of language elements with the objects and facts in the real world [Gle91]; some researchers call these associations *groundings* [Roy99]. So, early language learning in children (the so-called *one-word speech* stage) is driven by the *referential* concept of meaning: the word is predominantly used to refer to something the infant sees or touches, something that belongs to infant's immediate physical environment, rather than to denote abstract concepts [Nel73]. Expressed in technical terms, the first steps in learning to understand and use language come from pairing of acoustic input with another channel of side information (for instance, visual). This kind of understanding comes through learning cross-channel correlations in two (or more) input streams. To bypass possible ambiguities, two mechanisms can be employed:

- *correction and reinforcement*: this is a strategy where tentative semantic associations suggested by a child or a learning system are either confirmed or corrected by the teacher.

Reinforcement learning based on user's feedback have been implemented in practical language understanding systems such as [Gor91];

- *cross-situational inference*: repeated confrontation of the learner with paired stimuli possibly altered by legitimate variations. So, when presented with a living German shepherd, a picture of a poodle and a blue-colored plush dog, the child will finally find the right meaning of the basic-level category "dog".

For automated language acquisition, the second approach facilitates learning of word meaning by pairing not just one word and one semantic label (the meaning of this word in the non-acoustic channel), but a phrase or even a sentence with one or more semantic labels (or a complex structure thereof), and using information-theoretical measures to predict more detailed associations.

What can be learned from such pairings depends on the granularity level of the semantic labeling employed for the application. For complex internal organization of the associated semantic structures, the following *compositionality principle* can be used to derive the meaning of individual words (or contiguous word phrases): *"... the meaning of a sentence is derived from the meaning of the phrases it contains... "* [Tho98]. According to this principle, the utterance can be split in a number of (word-) phrases, each of which being held accountable for one component in the representation of sentence meaning [Sis96].

De Marcken in [dM94] describes a device which can automatically translate a continuous stream of phonemes into a linear sequence of corresponding *sememes* (semantic symbols that can be obtained from words by ignoring all inflections; for instance, "went" becomes GO, "mice" becomes MOUSE etc.). The lexicon to acquire during the training process is, in essence, a set of items of the form <*phoneme-sequence; set-of-sememes*>. It is obtained by means of a supervised process and optimized with respect to phonetic and semantic coverage of the training corpus (maximal coverage with minimal overlap). Thus, the weakness of this approach is that it presupposes that each part of the speech signal can be potentially equally important for the understanding of the utterance, be it a non-expressive word such as article THE or essentially meaningful word such as noun BALL. The system exhibited plausible behavior for written texts; the transition from word understanding to sentence understanding was factored out though.

Our primary interest lies in the understanding of complete spoken utterances. As pointed out in Section 2.1, a viable alternative to parsing these utterances word by word and combining the meaning according to the compositionality principle, is for many practical applications the holistic approach of classification of the entire utterance according to its semantic category, optionally extended by the extraction of the conforming semantic attributes. It has been shown in Chapter 2 that despite losing some depth of understanding, the approach turns out to be more robust for natural spoken language. Besides, it is always a good start to consider the rough labeling first; in fact, it had been suggested about children that their initial language learning is so successful, because their information processing limits are low, so that they ignore much of the linguistic information in speech [New90].

As described in Chapter 3, the only semantic labels available in our case are calltypes, so the only semantics learnable from the two channels, are word meanings as indicated by words' associations with these same calltypes which we also used to call "salience". In [Gor94b] this view on semantics was used to acquire lexicon and semantics from isolated word sequences with the only supervision provided as the expected machine action for each utterance. The system had no constraints on vocabulary or grammar; the word tokens were automatically acquired from the speech signal. To decide whether the observed word-token was already known to the system or represented a new word, the *dynamic time warping (DTW-)* based comparison procedure was employed.

The algorithm described in [Gor99] was designed to acquire salient phone strings from a continuous phone stream obtained as the output of a task independent ASR-system. The search for these strings in the training corpus took place in three steps:

1. extract frequent phone sequences of length less than or equal to four from the corpus;
2. filter the corpus with these phone sequences;
3. select frequent subsequences from the filtered corpus of fragment length less than or equal to four that also possess high values of mutual information and salience.

The algorithm had an obvious shortcoming in the use of filtering instead of parsing. Besides, no optimization concerning interchangeability within the set of generated salient phone strings was done which resulted in significant redundancy in this set. Since the extracted salient phone strings were used for calltype classification, this redundancy led to unnecessarily increased classification times. Next, we will explain our way to resolve these problems.

5.1.3 Multipass Algorithm for Acquisition of Salient Phone Strings

The *multipass algorithm* we describe in this section was first proposed in [Lev01] as a more efficient successor for the algorithm in [Gor99] mentioned earlier. Conforming to the lexical and semantic consistency features that make out words (cf. page 87), it searches for phone strings that increase the likelihood of the corpus initially expressed at the phone level, and expose high correlations with utterance calltypes embodied in their salience. This iterative algorithm is

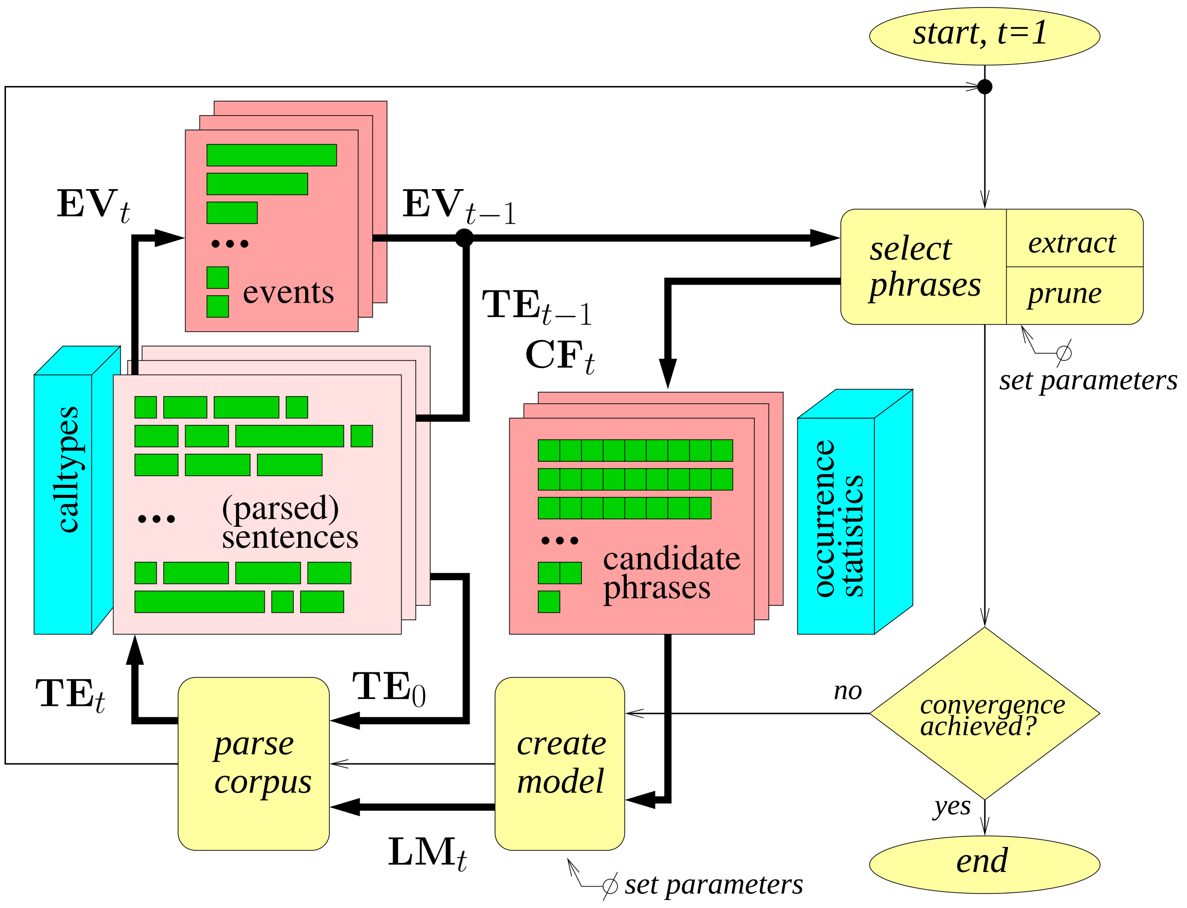


Figure 5.2: Iterative multipass algorithm for extraction of salient phone strings from the training set; bold lines show the data flow, thin lines the program flow.

schematically illustrated in Figure 5.2. We first present a high-level description of the algorithm and then provide specific details to each of its steps.

On each iteration we examine *phrases*: sequences of *events*. During the first iteration, these events are just phones, and afterwards they are created from phrases selected on the previous iteration. Let's denote by $\mathbf{TE}_0$ the initial corpus represented at the phone level. Then we iterate as follows:

Input

set of events $\mathbf{EV}_{t-1}$ and corpus $\mathbf{TE}_{t-1}$ from the previous iteration: set of sentences represented as sequences of events with corresponding calltypes labels;

Generate

set $\mathbf{CF}_t$ of phrases (subsequences of the above event sequences); each generated phrase consisting of not more than n events;

Prune
the set of generated phrases based on the within-language modeling utility and salience criteria;

Stop iterations
if there are no significant changes in $\mathbf{CF}_t$ compared to the previous iteration;

Estimate
a stochastic (unigram) language model $\mathbf{LM}_t$ using selected phrases from $\mathbf{CF}_t$ as a lexicon;

Parse
the original corpus $\mathbf{TE}_0$ using $\mathbf{LM}_t$ and, thus, express it in terms of the phrases from $\mathbf{CF}_t$, creating a new corpus representation $\mathbf{TE}_t$;

Output
the new set of events $\mathbf{EV}_t$ consisting of the phrases from $\mathbf{CF}_t$ observed in $\mathbf{TE}_t$ plus the original phone lexicon.

To illustrate this procedure, let's consider the following sequence of phones from $\mathbf{TE}_0$ (here and further in this thesis we will use ARPABET to illustrate our algorithms working at the phone level, albeit our internal representation roots in an alternative phonetic symbol set):

ay n iy D T uw m ey K ey K ax l eh K T K ao l

representing the sentence *"I need to make a collect call"*. Let $n = 4$. Under our experimental conditions, the set of the phrases generated and selected on the first iteration, includes [*n iy D*], [*T uw*], [*m ey K ey*], [*K ax l*] and [*T K ao l*], and the parser segments the original sequences of phones into the following sequence from $\mathbf{TE}_1$:

ay n_iy_D T_uw m_ey_K_ey K_ax_l eh K T_K_ao_l,

where *a_b_c* denotes a new event created out of the phrase [*a b c*]. On the second iteration we acquire new longer phrases: [*K_ax_l eh K T_K_ao_l*], [*n_iy_D T uw m_ey_K_ey*] and [*ay n_iy_D T uw*]. With these phrases, the original phone sequence will be parsed into:

ay n_iy_D_T_uw_m_ey_K_ey K_ax_l_eh_K_T_K_ao_l

which will be its representation in the training corpus $\mathbf{TE}_2$. In the end, we hope to obtain a segmentation of the original corpus, which resembles its corresponding representation at the level of words (or common word phrases).

Generation and Pruning of Phrases

We now describe the generator module of the algorithm in more detail. As stated earlier in this section, there are several criteria that can be used to identify words in a continuous phone stream. Hence, the generation module must be arranged to produce phrases compliant with these criteria. On each iteration we first accumulate all observed phrases that contain no more events than some fixed number n, and then compute their control values to prune away all phrases that do not satisfy the word-defining criteria.

Given a phrase $s = [w_1 w_2 \dots\ w_T]$ the following criteria are checked:

- utility of the phrase for within-language modeling:
  Tying up to the interpretation of mutual information as the reduction in code length achieved by coding a sequence of symbols (in our case, phones) $w_1\ \dots\ w_T$ by a new composite symbol $w_1\_\dots\_w_T$, we define mutual information[4] $I(s)$ of s as:

  $$I(s) = \log_2 \frac{P(w_1 \dots w_T)}{P(w_1) \dots P(w_T)}. \tag{5.2}$$

  Since each iteration only knows phrases consisting of events which, for their turn, can represent phrases collected on previous iterations, we have to be able to re-express (5.2) in terms of these events. To illustrate how this can be done, we consider an example of mutual information of phrase $s = [w_1\_w_2\_w_3\ w_4\_w_5\ w_6\_w_7]$ consisting of three such "composed" events. According to (5.2):

  $$\begin{aligned} I(s) &= -\log \frac{P(w_1\, w_2\, w_3\, w_4\, w_5\, w_6\, w_7)}{P(w_1) * P(w_2) * P(w_3) * P(w_4) * P(w_5) * P(w_6) * P(w_7)} \\ &= -\log \frac{P(w_1\, w_2\, w_3\, w_4\, w_5\, w_6\, w_7)}{P(w_1\, w_2\, w_3) * P(w_4\, w_5) * P(w_6\, w_7)} \\ &\quad -\log \frac{P(w_1\, w_2\, w_3)}{P(w_1) * P(w_2) * P(w_3)} - \log \frac{P(w_4\, w_5)}{P(w_4) * P(w_5)} - \log \frac{P(w_6\, w_7)}{P(w_6) * P(w_7)} \\ &= \tilde{I}(w_1\_w_2\_w_3\ w_4\_w_5\ w_6\_w_7) + I(w_1\, w_2\, w_3) + I(w_4\, w_5) + I(w_6\, w_7). \end{aligned}$$

  where $\tilde{I}(\cdot)$ denotes mutual information computed in terms of the events at the current iteration. This principle is used to compute mutual information of a phrase at the phone level in the most general case. By assuming the intra-channel relevance measure for the

[4]The notation $I(s)$ is to be understood as $I(w_1, w_2, \dots, w_T)$; note also that unlike mutual information of two random variables, the measure we are using always applies to one particular outcome combination.

phrase s as its mutual information normalized by the length of the underlying phone string:

$$I^{\mathrm{norm}}(s) = \frac{I(s)}{\text{length}_{\text{phone}}(s)}, \tag{5.3}$$

we expect to select phrases that, when used to reparse the corpus, will increase its likelihood (see equivalence (5.5) below). Since for the unigram-based estimation maximizing corpus' likelihood is equivalent to minimizing its entropy, we can call the phrases (and their underlying phone strings) that pass the mutual information pruning *entropy-reducing*.

- utility for understanding (salience for the task):
  In Section 3.1 we already mentioned two salience measures, one (3.1) being PMAX, the maximum of conditional semantic posteriors, and the other (3.2) KL, a measure based on the Kullback-Leibler (KL) distance between prior and posterior semantic distributions that can also be rewritten in terms of mutual information, as it was introduced in [Gor95] (using c to denote the random variable of a semantic category):

  $$\text{KL}(s) = D[P(c|s) \parallel P(c)] = \sum_{m=1}^{M} P(c^m|s) I(s, c^m). \tag{5.4}$$

  Both of these measures can be used to select semantically relevant salient phrases.

- reliability of these characteristics:
  While computing the control values above, the reliability of the estimates must be examined critically in each individual case. Some phrases are not observed frequently enough to possess usable ML-estimates of mutual information or salience. Consider a phrase that occurs only once in the corpus. For this phrase, the estimate of PMAX-salience is 1.0, which is the highest value possible, although it is clear that hardly any statements can be made about such phrase. The number of occurrences $\#s$ of the phrase s in the corpus is a simple correlate of the reliability of its statistically estimated characteristics, and is used in the pruning mechanism of the multipass algorithm. An even better criterion of the reliability of the salience measure is provided by the *multinomial significance test* [Wri97]. It will be described in detail in Section 5.2.

Language Model and Parsing

To parse a corpus with the selected phrases we need to create a language model. On each iteration t we create a simple unigram language model $\mathbf{LM}_t$, whereby the nature of the costs the phrases are provided with, is determined by the parsing goal. If we are interested in replicating the lexical consistency of words, we will try and find the parse that maximizes the likelihood of the corpus. In this case, the costs of every phrase can be set to the negative mutual information of this phrase computed according to (5.2). Then, among all possible competing parses the one with the lowest overall costs is chosen. It is not difficult to see that the parser built in this way is equivalent to a ML-parser. Indeed, on every iteration we are looking for a segmentation S^* of the sentence in phrases $s_1 \dots s_T$ such as to maximize the objective:

$$\begin{aligned}
S^* &= \operatorname*{argmin}_S \left(\sum_{s_t \in S} - \log \frac{P(s_{t,1} \dots s_{t,U(t)})}{P(s_{t,1}) \dots P(s_{t,U(t)})} \right) \qquad (5.5)\\
&= \operatorname*{argmax}_S \left(\log \prod_{s_t \in S} \frac{P(s_t)}{\prod_{u=1}^{U(t)} P(s_{t,u})} \right)\\
&= \operatorname*{argmax}_S \frac{\prod_{s_t \in S} P(s_t)}{\prod_{\tau=1} P(w_\tau)} = \operatorname*{argmax}_S \prod_{s_t \in S} P(s_t) = \operatorname*{argmax}_S P(S),
\end{aligned}$$

where each $s_{t,u}$ is the u^{th} phone of the t^{th} phrase in the parse S and at the same time the τ^{th} phone w_τ in the sentence, so that there is a one-to-one correspondence between pairs (t, u) and indices τ.

For each phrase s, its probability is ML-estimated as:

$$P(s) = \frac{\#s}{\sum_j \#s^j}. \qquad (5.6)$$

Special precautions must be taken in the situation when the maximal allowed number of events in a phrase is greater than one. One solution is to split the space of phrases on each iteration according to the number of events the phrases consist of, putting up with local probability distortions till the next iteration where all selected phrases will become events and thus have the same length one.

As earlier for the task of named entity extraction, we used the FSM toolkit [FSM] to implement the multipass algorithm. Here, the unigram language model is build as a union of the FSMs representing phrases and also trivial phones. This union is then composed with the sentences represented as sequences of phones, producing (after taking bestpath through the composition result) their parse of the highest likelihood.

Remarks on Convergence

The reason why the iterative process converges is its relation to the EM-algorithm with decision-oriented re-estimation of statistics where the phone representations of the sentences are observed variables and their segmentations in phrases are hidden variables. In fact, even if we keep the maximal phrase length permanently greater one throughout the iterations, it takes only a few iterations before the process converges. We say that convergence is attained if there is not more than 5% differences between $\mathbf{CF}_{t-1}$ and $\mathbf{CF}_t$; in other words: if the number of the one-event phrases selected on some iteration doesn't exceed 5% of the total number of phrases selected on this iteration.

5.2 Semantic Significance Test

When computing phrase statistics such as salience for the task, we rely on the maximum-likelihood estimators of probabilities obtained from the training corpus. Although one might rightly expect these estimators to be close to the true values when the number of occurrences of the phrase in the corpus is high, the question remains: what number of occurrences is "high enough"? Given one particular phrase with a high salience estimate, what is the risk that this value came about purely by chance, whereas in reality there is no significant dependency between occurrences of this phrase in an utterance and the semantic calltype of this utterance. In other words, how can we estimate the probability that an incident of such kind has happened?

A common approach for such problems is called *goodness-of-fit test* [Rea88] and is generally used to assess the credibility of a hypothesized model on the basis of a given data sample. Given the hypothesized model, it calculates expected statistics of the presumed distribution and then compares them against the statistics extracted from the available sample. χ^2-*test* [Hog95] is a very popular example for this approach. It investigates the asymptotic convergence of the distribution of Pearson's function $\mathbf{X}^2()$ (other appropriate functions can also be used) to the χ^2 distribution. Pearson's function is defined as:

$$\mathbf{X}^2() = \sum_m \frac{(N_m - N * p_m)^2}{N * p_m}, \tag{5.7}$$

where $p_1 \dots p_M$ are a-priori probabilities[5] of classes $c^1 \dots c^M$ in the hypothesized model and $N_1 \dots N_M$ are observed quantities of the latter in the sample of size N. The χ^2 has been proven to be the correct target distribution for $\mathbf{X}^2()$ when the data samples are sufficiently large [Rea88] (this also explains the notation of *asymptotic convergence*). However, in our case the main difficulty is the sparsity of data samples which makes the goodness-of-fit test inapplicable for our needs.

5.2.1 Multinomial Signifi cance Test

An alternative method to assess the credibility of high salience values which is also immune against small data samples is the *multinomial significance test* [Wri97]. It examines all possibilities for the partitioning of the total of N observations of phrase s in M groups, each representing one calltype.

Let's denote $N_{i,m}$ the number of times s occurs for the m^{th} calltype with prior p_m in the partition r_i. We now consider the null-hypothesis of statistical independence of the phrase s and the calltype of the utterances, s is a part of. If the null-hypothesis holds, the conditional distribution over calltypes induced by s will be the same as the prior calltype distribution, and this means that the probability of a partition $r_i = (N_{i,1}, \dots, N_{i,M})$ will obey the multinomial distribution:

$$P(r_i|N) = N! \prod_{m=1}^{M} \frac{(p_m)^{N_{i,m}}}{N_{i,m}!}, \quad N = \sum N_{i,m}. \tag{5.8}$$

To accept or reject the salience estimate of the phrase the following rule is employed:
Accept the high salience estimate as reliably correct if under the null-hypothesis that the conditional semantic distribution is governed by priors, the sum of probabilities of the partitions which are not more probable than the actually observed one doesn't exceed threshold α:

$$\sum_{r_i:\, P(r_i|N) \leq P(r_{\text{obs}}|N)} P(r_i|N) \leq \alpha. \tag{5.9}$$

Otherwise, the estimate will be rejected as unreliable and the phrase itself will be dismissed. There are altogether C_{N+M-1}^{M-1} possible partitionings of N events into M groups. If this number is moderate the exact multinomial significance test can be conducted, i.e. all possible partitions will be considered one by one. Otherwise alternative methods must be used.

[5]Here and in the next section we will use subscripts instead of superscripts to distinguish indices from exponents.

5.2.2 Exact Multinomial Significance Test

The rule for accepting or rejecting of salient phrases formulated in the previous section can be restated in terms of probability thresholds. Indeed, if we introduce a random variable[6] $X = P(r_i)$, then the multinomial significance test is equivalent to finding an $x(\alpha)$ such that

$$F(x(\alpha)) = P(X \leq x(\alpha)) = \alpha, \tag{5.10}$$

where $F(x)$ is a cumulative distribution function (see left plot in Figure 5.3). Thus, if we were able to exhaust the partition space by generating partitions in the order of non-decreasing probabilities, we could stop the generation process as soon as $F(x)$ exceeds α and declare the found value $x(\alpha)$ to be the corresponding threshold on the probability. In order for the examined phrase s to be accepted as salient at significance level α, the probability $P(r_i(s))$ of the partition it induces in the data sample, given the null-hypothesis, may not exceed $x(\alpha)$. The approach is advantageous in the sense that it will suffice to calculate $x(\alpha)$ only once for every pair of N and M, and not for each phrase that must be examined.

Usually, the task of generating partitions in the order of increasing probabilities is not trivial though, and we are forced to first enumerate all partitions in order to generate the entire distribution of X, after which the partitions can be reordered according to their probabilities. This results in additional time and space complexity of the algorithm. The space aspect of this problem can be alleviated, if we store the distribution of $X = P(r_i)$ instead of the entire (ordered) set of possible partitions with corresponding probabilities. The most natural way to store the distribution of X is the histogram form, for which we also suggest to use the logarithmic scale for the computational reasons (right plot in Figure 5.3).

Thus, to compute the probability threshold $x(\alpha)$ we use the following off-line algorithm for each N (in our task M is fixed):

1. Create histogram $\mathbf{H}(x) \sim F(X \leq x = P(r_i))$ by iterating through the entire partition space;

2. Iterate from left to right over $\mathbf{H}(x)$ accumulating partition probabilities, stop by the first $\bar{x}$ such that threshold α is exceeded;

3. Store this $\bar{x}$ as the threshold on probability $x(\alpha)$ that corresponds to significance level α.

During the online phase of the exact multinomial significance test, compare for each phrase the probability (under the null-hypothesis that the conditional and prior semantic distributions are

[6]For simplicity reasons we will skip the conditioning on N.

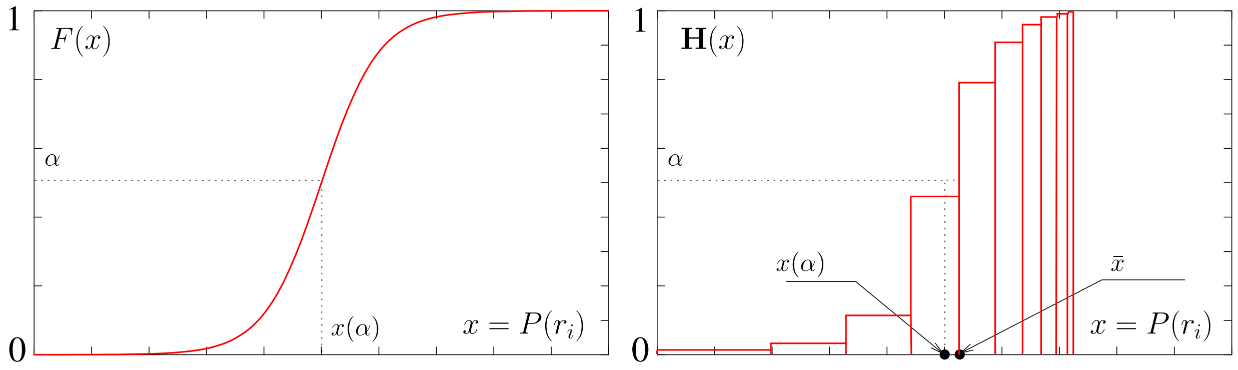


Figure 5.3: Practical realization of exact multinomial significance test; left: estimating the probability threshold $x(\alpha)$; right: histogram-approximation in practice produces $\bar{x}(\alpha)$.

the same) of semantic partition it induces against this threshold and accept phrase if the threshold is not exceeded.

5.2.3 Possible Alternatives

If the number of partitions r_i is large, iterating through the entire partition space becomes onerous and an approximate solution is to be looked for. Two alternatives to the exact multinomial significance test seem to be appropriate:

- Histogram approximation:
  Based on a small sample of randomly generated partitions[7] of size $\hat{N} << C^{M-1}_{N+M-1}$, estimate $\mathbf{H}_{\hat{N}}(x)$. The claim is that $\mathbf{H}_{\hat{N}}(x) \approx \mathbf{H}(x)$ when the sample is representative enough;

- Threshold on the false rejection rate of the null-hypothesis (*Monte-Carlo method*):
  Again, we generate a random sample of $\hat{N}$ partitions, but this time according to the prior calltype distribution. The probabilities of these partitions are then stored in an array sorted in the increasing order. In this array we declare the element number $\alpha\hat{N}$ to be the probability threshold, arguing that proportion α of all further partition samples randomly generated according to the semantic (calltype) priors, will have probability less than this, leading to the false rejection of the null-hypothesis that the induced distribution is governed by the priors for this proportion.

[7]Here the random sampling plan must be such that the probability of generation of any partition is equal to $1/C^{M-1}_{N+M-1}$.

In practice, both methods turned out to produce acceptable and largely concordant results, albeit with high errors by the estimation of the contribution of the highly probable partitions to the cumulative probability distribution function, in cases where the sample was smaller than the actual partition space by more than 10 orders of magnitude.

5.3 Clustering of Salient Phone Strings into Acoustic Morphemes

After the phrase generation and pruning stage has come to an end, the set of the phone strings that the surviving phrases represent must be compressed by ignoring the irrelevant variations these strings might reveal. Phone strings that differ only insignificantly in their acoustic and semantic characteristics, will be combined to form clusters that we call *acoustic morphemes*. The name comes from the definition of morpheme as a minimal unit of meaning in the language, and the fact that, in our task setup, all information is extracted from audio signals.

Two factors justify the need for such clustering:

- Since the extraction algorithm is designed to operate on ASR-output, it has to be able to deal with noisy data and handle (at least the most frequent) misrecognitions. If *"collect call"* is recognized times as *"K ax l eh K T K ao l"* and times as *"K ax l eh K ao l"*, this minor difference shouldn't become an impediment to identification of both phone strings as one acoustic morpheme[8]. Moreover, pooling these strings together will allow for a more robust estimation of the semantic characteristics of the (hopefully less frequent) erroneous variant *"K ax l eh K ao l"*, based on more observations.

- As a matter of fact, we don't have to limit ourselves to different phone representations of the same word to carry out this kind of pooling: *"I would like to"* and *"I'd like to"* can be easily thrown into the same pool, as long as this doesn't contradict application semantics. Still, we decided not to neglect acoustic similarity completely when making decision whether to put two synonymical phone strings in the same acoustic morpheme or not.

[8]The same holds for alternative pronunciations.

5.3.1 Overview of Existing Clustering Methods

The problem of clustering arises for various applications in machine learning, database administration, image processing and other domains. We are interested in the *distance-based* approach to clustering; clustering of this type can be performed whenever there exists a well-defined distance metric[9] $D(s^i, s^j)$ between any two points (in our case, these points are phone strings) s^i and s^j in the data space which reflects adequately the similarity of the pair.

Most of the clustering methods described in the literature are of either *partitioning* or *hierarchical* type. While hierarchical clustering [Mur83] returns a path from the top-level cluster containing all elements of the data set to each one-element cluster in terms of embedded subclusters, in the definition of partitioning clustering methods in [Kau90] following requirements are imposed. Given the set of elements to cluster and the desired number of clusters, the partitioning approach will search for a set of clusters $\{O^l\}$ fulfilling two necessary conditions:

- each cluster contains at least one element;
- each element belongs to exactly one cluster.

Suppose that we have the means to calculate the distance from an element to a cluster. Then, similar to the value quantization algorithms, among all possible partitions we choose the one minimizing the approximation error:

$$\epsilon = \sum_i D(s^i, O^{l(i)}), \tag{5.11}$$

where s^i is the i^{th} element in the data set (a set of phone strings in our case) and $l(i)$ is the index of the cluster corresponding to s^i in the partition.

Various examples of partitioning algorithms are presented in [Lin80, McQ67]. The algorithm we use for our application bears the most similarities to the method called ISODATA [Bal65] and is capable of changing dynamically the number of clusters in the partitioning.

5.3.2 Multi-distance Clustering

For our experiments we decided to use the distance-based clustering approach, i.e. we assume that it is possible to compute distance between any two phone strings. When designing distance-based clustering mechanism, three questions must be answered:

[9]In [Kau90] an important remark is made that in general, this measure doesn't have to satisfy the distance criteria, and a more loose notation of *dissimilarities* is proposed.

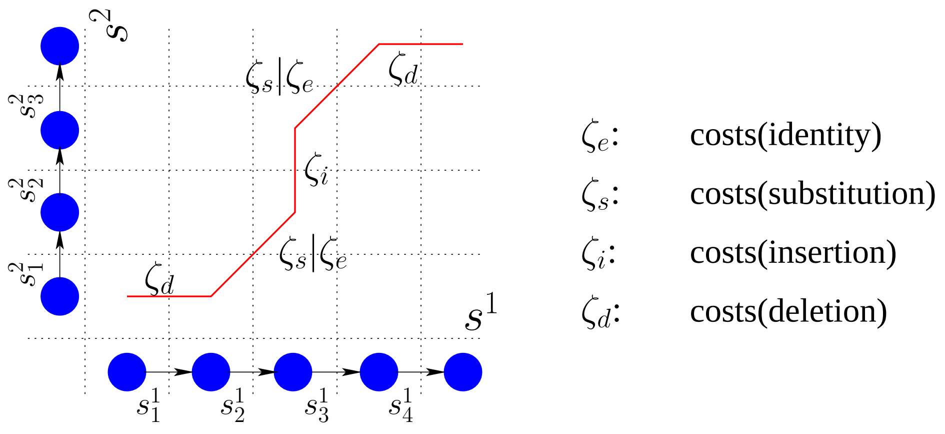


Figure 5.4: Simple Levenshtein distance $D_{\mathrm{SL}}(s^1, s^2)$ between phone strings s^1 and s^2 computed using dynamic programming.

- of what nature are the underlying distances?
- how to initialize clusters?
- which clustering strategy to employ?

In the remainder of this section we will address these questions subsequently.

Distance Definitions

We are aiming at clusters of phone strings exhibiting a high degree of homogeneity in both acoustic and semantic parameters. The simplest and still very popular measure of acoustic distance between two phone strings is the *simple Levenshtein* (SL-) distance which uses dynamic programming to align two strings, penalizing equally every deletion, insertion and substitution of phones respectively, and then defines the acoustic distance D_{SL} as the sum of these penalties (costs). See Figure 5.4 for the illustration of this mechanism.

Using SL-distances has an advantageous time behavior but can not be adjusted to the specific characteristics of the ASR used to produce phone-level transcriptions of the utterances. This shortcoming is taken care of by the *weighted Levenshtein* (WL-) distance which assigns unique costs to each phone deletion, insertion and substitution.

These costs are determined according to the requirements of the task. In the case where clusters are only designed to compensate for the mistakes made by ASR, the weighting scheme presented in [Ris98] is a good choice. However, as was suggested above, we would like to put in one cluster also synonyms, which still have to be acoustically similar but not necessarily because of the same lexical origin[10].

The above considerations led us to the decision to leave the search for the optimal costs needed by WL-distance to future work and to limit the experiments of this thesis to simple Levenshtein distance which, in the simplest case, effectively encodes the distance between two phone strings as the number of phone-mismatches in their alignment. The longer the phone strings the more mismatches can be potentially observed. Our intuition is that the extent of tolerated variety should grow along with the string lengths. Thus, we define the *normalized* SL-distance between strings s^i and s^j as:

$$D_{\mathrm{SL}}^{\mathrm{norm}}(s^i, s^j) := \frac{2 * D_{\mathrm{SL}}(s^i, s^j)}{\text{length}_{\text{phone}}(s^i) + \text{length}_{\text{phone}}(s^j)}. \tag{5.12}$$

As far as semantic distances are concerned, we employ two different metrics in our experiments to estimate the distance between two conditional semantic distributions $P(c|s^i)$ and $P(c|s^j)$ for calltypes c.

1. *Kullback-Leibler distance*

   This asymmetrical distance between two distributions shows how much information one distribution provides about the other:

$$D_{\mathrm{KL}}(s^i, s^j) = \sum_{m=1}^{M} P(c^m|s^i) \log \frac{P(c^m|s^i)}{P(c^m|s^j)}. \tag{5.13}$$

2. *Weighted Semantic Square distance*

   This distance represents a mean-square semantic distortion measure which considers correlations between two distributions and can compensate for small sample size [Wri97]. If $\mathcal{H}_0$ is the null-hypothesis that s^i and s^j induce the same posterior semantic distribution, and $\boldsymbol{n} = (N_i, N_j, N_{ij})$ is the vector containing the numbers of utterances containing s^i, s^j and both strings respectively, then the WSS-distance is computed as:

$$D_{\mathrm{WSS}}(s^i, s^j) = \frac{1}{M} \sum_{m=1}^{M} \frac{\left[P(c^m|s^i) - P(c^m|s^j)\right]^2}{\text{var}\left(P(c^m|s^i) - P(c^m|s^j)|\mathcal{H}_0, \boldsymbol{n}\right)}. \tag{5.14}$$

[10]Another reason for abstaining from WL-distance paradigm are the artifacts of the phrase acquisition algorithm described in [Lev01].

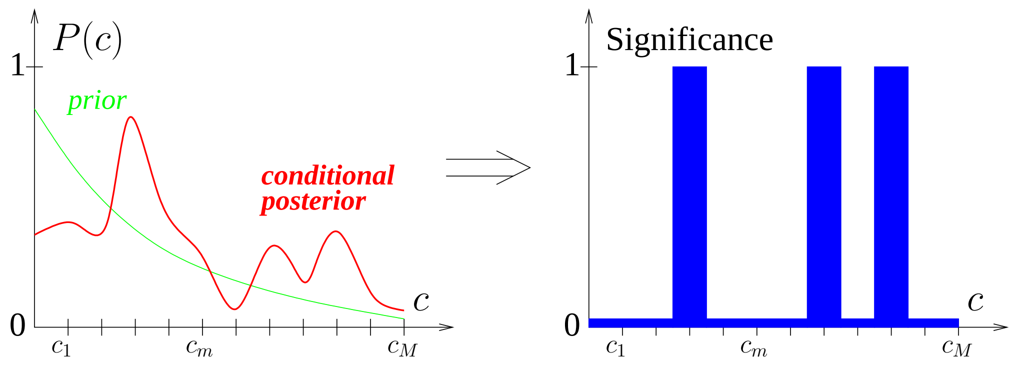


Figure 5.5: Fisher's significance test.

We can also express this formula in terms of co-occurrence statistics:

$$D_{\text{WSS}}(s^i, s^j) = \frac{1}{M} \sum_{m=1}^{M} \frac{\left[N_j N_i^m - N_i N_j^m\right]^2}{N_i N_j \left[N_i^m + N_j^m - 2N_{ij}^m\right]}, \tag{5.15}$$

where N_i^m and N_{ij}^m are numbers of utterances of semantic class c^m containing string s^i and both strings s^i and s^j respectively.

The introduced measures of SL-, WL- and KL-distances are asymmetrical in general case. To obtain a symmetrical version we average:

$$D^{\text{sym}}(s^i, s^j) = \frac{D(s^i, s^j) + D(s^j, s^i)}{2}. \tag{5.16}$$

Cluster Initialization

To bootstrap the clustering process we need one (possibly rough but easy to evaluate) distance measure which automatically induces a set of initial clusters. In our experiments we start by looking only at the semantics and base this distance measure upon *Fisher's exact significance test* [Ped96]. Conditional posterior distributions are binarized as follows: for each phone string and each semantic calltype we set the corresponding bit in the array representing conditional distribution induced by this string to one, if there is a significant positive correlation between occurrences of the string and the probability of this calltype, and to zero otherwise.

In general to ascertain correlations between two events X and Y Fisher's exact significance test suggests creating contingency tables with fixed marginal totals N_X, N_Y, N as shown in Ta-

	X	$\bar{X}$	
Y	$N_{X,Y}$	$N_{\bar{X},Y}$	N_Y
$\bar{Y}$	$N_{X,\bar{Y}}$	$N_{\bar{X},\bar{Y}}$	$N_{\bar{Y}}$
	N_X	$N_{\bar{X}}$	N

Table 5.1: Contingency table Θ for Fisher's significance test.

ble 5.1. The probability of each table Θ can be computed using the hypergeometric distribution:

$$P(\Theta) = \frac{N_X! N_Y! N_{\bar{X}}! N_{\bar{Y}}!}{N_{X,Y}! N_{\bar{X},Y}! N_{X,\bar{Y}}! N_{\bar{X},\bar{Y}}!} * \frac{1}{N!}. \tag{5.17}$$

The dependency is considered significant if under the null-hypothesis (events are statistically independent) the sum of probabilities of the tables not more probable than the actually observed one doesn't exceed threshold α:

$$\sum_{\Theta_i:\, P(\Theta_i) \leq P(\Theta_{\text{obs}})} P(\Theta_i) \leq \alpha. \tag{5.18}$$

Furthermore, we consider the significant correlation to be positive if an increase in the number of observations of one event causes increase in the number of observations of the other one. Thus, in the binary distribution array for each phone string we write 1 at position m if and only if this string has a positive significant correlation with semantic calltype c^m and 0 otherwise. It's easy to observe that Fisher's test applied to s and c has the binarizing effect pictorially depicted in Figure 5.5.

Now the distance between two binary distribution arrays D_{BIN} can be defined as the number of positions in which both arrays differ, and initial clusters can be created out of the phone strings with the pair-wise distances $D_{\text{BIN}} = 0$.

Computing Distances to Clusters

When employing the distance-based clustering procedure, one has to ensure the computability not only of the distances between individual strings but also between string and cluster, cluster and cluster (*inter-cluster distances*) and also within a cluster (*intra-cluster distances*). Because of the versatile nature of distances (acoustic and semantic) we employ in our experiment, we relinquish the idea of compression of all distances into one scalar. Instead we decided to work with a distance vector: $\boldsymbol{d} = (D_{\text{BIN}}, D_{\text{SL}}^{\text{norm}}, D_{\text{KL}}, D_{\text{WSS}})$, with D_{SL} representing normalized sim-

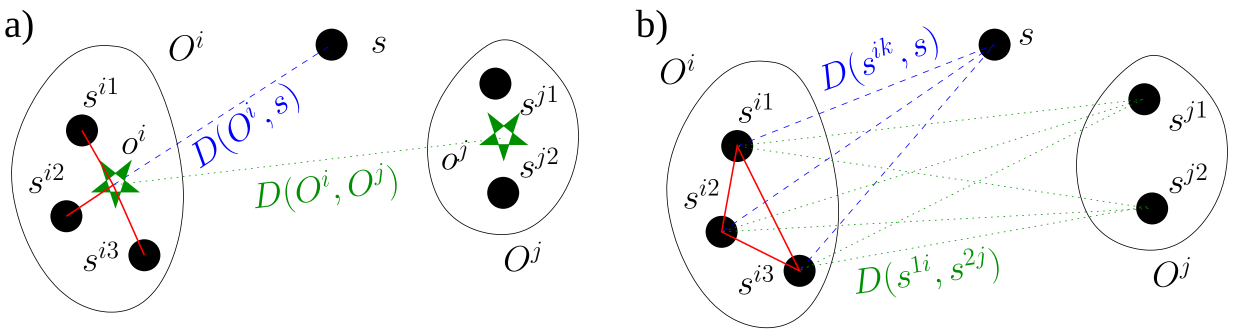


Figure 5.6: Two ways to compute distances to a cluster: a) centroid-based b) averaged. Solid lines show which string-to-string distances must be computed for intra-cluster distances, dashed lines for string-to-cluster distances and dotted lines for inter-cluster distances.

ple Levenshtein distance and D_{KL} and D_{WSS} Kullback-Leibler and weighted semantic square distances respectively (symmetrical versions).

When computing any of these measures, we have the choice between two strategies:

- *centroid-based distances*;
- *averaged distances*.

In the case of centroid-based distances, whenever we consider a distance to a cluster O^i, we replace O^i by a single (pseudo-) string o^i chosen so as to have acoustic and semantic characteristics representing the middle point for the entire community of strings s^{ik} contained in O^i. Then, the distance to O^i is replaced by the distance to this string o^i (Figure 5.6a). In particular, to obtain the semantic centroid of a cluster as its mean, we sum up the conditional semantic distributions for all $s^{ik} \in O^i$. The situation is slightly more complicated with the acoustic middle point of cluster. While definition of acoustic mean seems to be not trivial in our case, it is much easier to replace it by acoustic median (*medoid*) of the cluster: this is the phone string from O^i observed most frequently in the training corpus.

With the determined middle-point o^i of cluster O^i, the string-to-cluster and inter-cluster distances can be rewritten for any component of the vector $\mathbf{D}$ as:

$$\begin{cases} D(s,O^i) := D(s,o^i) \\ D(O^j,O^i) := D(o^j,o^i). \end{cases} \tag{5.19}$$

We will also need the intra-cluster distance as a homogeneity measure for the cluster, which we postulate as its *radius*. Denote the number of observations of string $s^{ik} \in O^i$ as $\#s^{ik}$, then the intra-cluster distance of O^i is:

$$D(O^i) := \sum_{s^{ik} \in O^i} \frac{\#s^{ik}}{\sum \#s^{ik}} D(s^{ik}, o^i). \tag{5.20}$$

Although calculation of the centroid-based distances is fast, it is not straightforwardly applicable to D_{WSS} because this metric relies on the availability of co-occurrence statistics, and by the latter, the transition from strings to clusters is not easy. One solution to this problem is the computation scheme where distance to a cluster is calculated as the average distance to the strings this cluster contains (Figure 5.6b). The weights are determined according to the observed frequencies of the strings. Thus, when considering a string-to-cluster distance, we use the formula[11]:

$$D(s^{jl}, O^i) := \sum_{s^{ik} \in O^i}^{(j,l) \neq (i,k)} \frac{\#s^{ik}}{\sum \#s^{ik}} D(s^{jl}, s^{ik}). \tag{5.21}$$

The natural generalization of this metric for the inter-cluster distance is:

$$D(O^i, O^j) := \sum_{s^{ik} \in O^i} \sum_{s^{jl} \in O^j} \frac{\#s^{ik} \#s^{jl}}{\sum \#s^{ik} \sum \#s^{jl}} D(s^{ik}, s^{jl}), \tag{5.22}$$

and the corresponding intra-cluster distance is *diameter*:

$$D(O^i) := \sum_{s^{ik} \in O^i} \sum_{s^{il} \in O^i}^{l>k} \frac{2\#s^{ik} \#s^{il}}{\sum \#s^{ik} (\sum \#s^{il} - 1)} D(s^{ik}, s^{il}). \tag{5.23}$$

The clear disadvantage of this method is the increased computational complexity; when calculating inter-cluster distance, for instance, it becomes $|O^i| * |O^j|$ instead of $|O^i| + |O^j|$ (with $|O|$ standing for the number of elements currently in O) as it is the case with the centroid-based distances. However, it is possible to avoid the complete recalculation of intra- and inter-cluster distances each time one cluster is altered by employing a recursive scheme that adjusts these distances as soon as one string is added to or taken out from the cluster.

Shaping Clusters

After a number of initial clusters (we also call them *pre-clusters*) has been created, the next step is to refine this set so as to increase the degree of acoustic and semantic homogeneity within

[11]If the string pertains to the cluster to which we estimate its distance, the distance from the string to itself is not considered.

clusters while keeping distances between different clusters as large as possible. This general criterion has its analog in the classification task, where the feature transformation is selected so as to maximize inter-class distances while preserving small intra-class distances [Nie, page 198]. At this point we define an iterative procedure that fulfills this objective.

A very popular procedure for distance-based clustering was originally developed for vector quantization purposes and is known as *LBG-algorithm* [Lin80]. It optimizes successively the codebook (set of cluster centroids) by the given decomposition in clusters and the decomposition in clusters by the given codebook. Note that in the case of averaged distances, when no cluster centroids are computed, instead of a codebook there is a matrix of distances between clusters.

The modification of the LBG-Algorithm with an immediate correction of the codebook after any one of the elements has changed its cluster affiliation, is called *K-means algorithm* [McQ67] (or *K-medoids* when considering medians instead of means). For the efficiency reasons explained above, we employ this version together with averaged distances and stick to the classical LBG when using centroid-based distances.

An additional improvement concerns the possibility of changing the number of clusters on-the-fly and introduces elements of hierarchical clustering in the LBG-algorithm. One possible enhancement is to split a cluster in two if its intra-cluster distance exceeds a certain threshold; also any two clusters can be merged together if they don't lie far enough apart. All decisions about cluster modifications are made based on all elements of distance vector $\boldsymbol{d}$ or only on some of them, so that on each iteration we:

1. split cluster O^i in two if
$$\boldsymbol{d}(O^i) > \boldsymbol{\theta}_{\text{split}} \;\Big|_{\text{chosen distances}} ;$$

2. merge clusters O^i and O^j if
$$\boldsymbol{d}(O^i, O^j) < \boldsymbol{\theta}_{\text{merge}} \;\Big|_{\text{chosen distances}} ;$$

3. move element s^{ik} from cluster O^i to cluster O^j if
$$\boldsymbol{d}(s^{ik}, O^j) < \boldsymbol{d}(s^{ik}, O^i) \;\Big|_{\text{chosen distances}} ,$$

where the notation $\Big|_{\text{chosen distances}}$ means that the inequality has to hold only for selected components of the distance vector $\boldsymbol{d}$ and/or threshold vector $\boldsymbol{\theta}$.

Chapter 6

Experiments and Results

In this chapter we will present a series of experiments conducted to evaluate our algorithms for the task of spoken language understanding under the conditions of missing manually created transcriptions for the domain of interest. We will start by describing the data sets that we have chosen for these experiments, and by specializing the corresponding experimental setups. After that, we will describe baseline language models used in our experiments and show the consequences of the unsupervised language model adaptation on the word and phone error rates. In compliance with the structure of the previous chapters, different aspects of the spoken language understanding in the call center scenario will be then addressed in the following order:

1. Calltype classification;

2. Named entity processing (detection, localization and extraction);

3. Extraction of acoustic morphemes from a continuous phone stream.

6.1 Data Sets and Experimental Setups

Table 6.1 shows some of the general statistics collected for the corpora that we used in this work. In the following, we will describe each of these corpora in more detail.

The main source of data we used for the experiments in this thesis is the HMIHY corpus. The database is a collection of recordings of callers responding to the prompt *"AT&T. How may I help you?"* [Gor97]. Each recording contains one or more natural language sentences uttered by a caller in response to a single system prompt (also referred by the term *utterance*) and is represented as an audio file with speech signal of standard telephone quality, mu-law encoded

Corpus	duration (hours)	utterances	words	diff.words
HMIHY-1	93	47139	542712	5549
HMIHY-2	60	29744	367257	4646
HMIHY-3CA-train	63	35912	327783	4019
HMIHY-3CA-test	16	9146	84944	2184
HMIHY-3CI-train	27	11638	158135	3269
HMIHY-3CI-test	7	2972	40522	1715
HMIHY-3N-train	63	36021	327940	4037
HMIHY-3N-test	same as HMIHY-3CA-test			
PRES	32	19355	171957	3756

Table 6.1: Application-independent statistics of the corpora used in this work.

with a sampling rate of 8kHz. We declared this corpus to our target domain and postulated that no manual transcriptions generated for it can be used for training except for the experiments where manual transcriptions for the target domain are explicitely assumed available ("second baseline" experiments). Instead, for our main experiment threads, we restricted our annotated training sources to corpora from alternative discourse domains. The only requirement we imposed on these data was that they had to pertain to the same language (English) and possess the same acoustic quality (telephone speech), so that a generic acoustic model could be used for the recognizer. In particular, we trained our background language model on transcriptions from the SWITCHBOARD corpus [God92]. Experiments that use this language model in the recognizer will be called "first baseline" experiments. The out-of-domain data to train distortion FSMs for the employed word- and phone-recognizers (Section 4.3.4), came from the recordings of prescription-related telephone dialogs between customers and representatives of a drug company. This set – which we will refer to by PRES – contains ca. 19K utterances, for which also manual word transcriptions are available.

We created several tranches of data from the HMIHY corpus for the tasks of calltype classification and named entity processing:

1. HMIHY-1:
   This is a corpus consisting of about 47K utterances. We use it to train or adapt our language models. For the second-baseline experiments we train the word language model on the manually created word transcriptions of these utterances. Otherwise the corpus is used to adapt the background language model to the HMIHY-domain in the unsupervised way;

2. HMIHY-2:

The audio signals and manual transcriptions of this data set (about 30K utterances) are only to be used for the second-baseline experiments, and serve the purpose of training the distortion FSM;

3. HMIHY-3CA:
We use this data for calltype classification experiments. It includes all turns of the recorded dialogs. The corpus is split into training (HMIHY-3CA-train) and test (HMIHY-3CA-test) parts, ca. 36K / 9K utterances large respectively. The training and test corpora are also separated with respect to the dialogs, i.e. there is not a single pair of utterances from training and test corpora pertaining to the same dialog. Each utterance is paired with one or more calltype labels out of the 52 supported ones, including an open-end calltype OTHER (see Appendix B for the complete list);

4. HMIHY-3CI:
Same as the previous corpus, but only initial turns of the dialogs are considered. This comprises only those utterances of the dialogs prior to which no useful information about caller's intention could be retrieved. Training part contains ca. 12K utterances and the test part ca. 3K. 49 calltypes are supported for this experimental setup. The occurrence statistics of all calltypes in HMIHY-3CI are plotted in Figure 6.1;

5. HMIHY-3N:
We created this corpus for the named entity processing task. The training partition (HMIHY-3N-train, 36K) is used to train the detection classifier as described in Section 4.2, and the test partition (HMIHY-3N-test, 9K) to evaluate detection, localization and value extraction algorithms. We consider three named entity types altogether: PHONE_NUMBER, DATE and the context-specific named entity ITEM_AMOUNT that comprises all monetary expressions referred to on the customer bill, but nothing else[1]. Exact definitions of the supported named entity types and some examples hereto can be found in the Appendix C. Generally speaking, PHONE_NUMBER and ITEM_AMOUNT are subtypes of NE-type NUMEX in MUC, and DATE is a subtype of TIMEX [Chi97].

In Table 6.2 we pulled together some important statistics of the concerned HMIHY-corpora. We see that the initial utterances are on the average 1.5 times longer than the ones from all dialog turns. In fact, once the comprehension is attained, the caller's task often becomes to simply

[1]Thus, if the customer says: *"I sent you a check for fifty dollars"*, the *"fifty dollar"*-part is not an instance of named entity ITEM_AMOUNT.

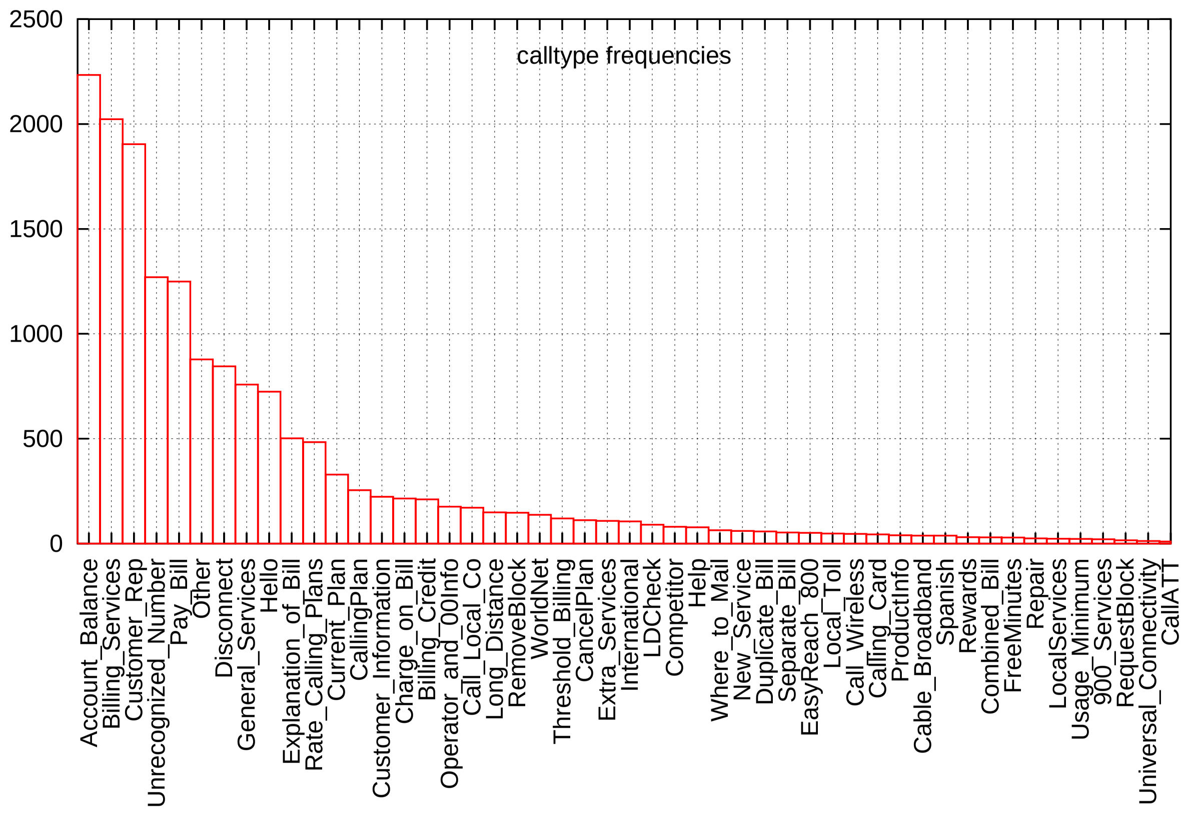


Figure 6.1: Occurrence statistics of calltypes in corpus HMIHY-3CI.

Task	Corpus (train+test)	words/utt. (average)	size of semantic lexicon	percentage of utterances with K labels				
				K=0	K=1	K=2	K=3	K≥4
CC	HMIHY-3CA	9.16	52	–	90.44	8.85	0.66	0.05
CC	HMIHY-3CI	13.60	49	–	88.97	10.28	0.70	0.05
NE	HMIHY-3N	9.14	3	94.49	4.59	0.72	0.15	0.05

Table 6.2: Selected statistics for corpora used in calltype classification (CC) and named entity processing (NE) tasks.

reassert (or negate) systems assumptions using short yes/no answers, or to provide specific information, e.g. date of the call, which is also mostly rather short. We found it useful to reduce the calltype dictionary for initial utterances by three categories that are only relevant for the subsequent parts of the dialog: YES, NO and REPEAT. Besides, as shown in the same table, most of the utterances have only one calltype label, and about 95 percent of them include no instances of supported named entity types[2]. Only an insignificant percentage of all utterances (ca. 0.05%) belong to more than three semantic categories or contain more than three instances of named entities.

For those utterances from HMIHY-3N that do contain named entities, exact annotation of the latter has been carried out, so that exact positions and values of each present NE-instance is available[3]. Table 6.3 shows detailed NE-statistics for the training and test partition of HMIHY-3N. For each named entity type we see how many utterances contain at least one instance of it and how many instances of this type there are altogether. Apparently, all three considered named entity types are equally represented in both corpora, whereby of all named entities, instances of DATE tend to co-occur more often than the rest (in the test corpus, for example, 294 instances of PHONE_NUMBER are distributed among 279 utterances, whereas 295 instances of DATE only among 197).

6.2 Original and Adapted Language Models

In this section we look at the results of the simple method for unsupervised language model adaptation that we outlined in Section 2.4.2 and discuss these results from the perspective of word- and phone accuracy. In all of the experiments in this and the following sections, the employed speech recognition engine was AT&T's WATSON recognizer [Sha97].

[2]All statements are based on manual labels.

[3]Unfortunately, because of some labeling artifacts, many NE-annotations refer to named entities *along* with their immediate context, e.g. *"charged twenty dollars"*.

NE-name	#NE		#phrases with NE	
	HMIHY-3N-train	HMIHY-3N-test	HMIHY-3N-train	HMIHY-3N-test
Date	1019	290	722	193
Item_Amount	1057	294	903	252
Phone_Number	1108	294	1045	279

Table 6.3: Occurrence statistics for named entities of different types and utterances containing at least one named entity of the given type for the training and test partition of HMIHY-3N.

iteration	lexicon size	κ	word accuracy HMIHY-1	word accuracy HMIHY-3CI-test	corresponding phone accuracy	real-time factor
backgr. LM (baseline 1)	27226	8	60.6%	63.1%	76.5%	1.8
1	9317	12	65.0%	70.4%	83.0%	1.1
2	8809	16	65.9%	70.1%	82.9%	1.1
3	8315	16	65.9%	70.1%	82.8%	0.9
4	8250	–	–	70.1%	82.9%	0.9
transcr. LM (baseline 2)	5551	–	–	74.7%	84.5%	0.8

Table 6.4: Unsupervised adaptation of the word language model and its effect on word and phone accuracy and real-time factor.

For the word-based adaptation a background language model was trained on ca. 186K utterance transcriptions from the SWITCHBOARD corpus. This language model was iteratively refined by recognizing HMIHY-1 (the adaptation corpus) and building new language model from the obtained ASR-transcripts, whereby on each successive recognition pass the language model's relative influence κ from (2.7), page 35 was increased. To assess the effect of the adaptation on the word accuracy, we evaluated it on HMIHY-3CI-test.

Table 6.4 summarizes the effect of the unsupervised language model adaptation on the word accuracy attained on the adaptation and test corpora. For the latter, the corresponding phone accuracy and the recognition real-time factor were gauged as well. Note that the word accuracy on the test set was always measured in the "field"-condition, that is, with the setting $\kappa = 16$ traditional for the WATSON recognizer. Additionally, the observed changes in the size of the recognition lexicon were followed throughout the iterations, starting with the background language model (baseline 1). The second baseline entry of the table (baseline 2) illustrates the performance of the language model trained on the manual transcriptions of the adaptation corpus. Three major positive adaptation effects could be distinguished:

- as the iterations progressed, the lexicon of the active recognizer narrowed down to the words that actually occur in the adaptation corpus. This rapid shrinkage led to a significant decrease in the recognition time. On the test corpus, the real-time factor was cut in half from 1.8 down to 0.9, coming close to the value of 0.8 measured for the case of the transcription-trained language model;
- on the other hand, the word accuracy (and the corresponding phone accuracy) increased dramatically as a result of the adaptation, making up for 60% of the initial difference between the two baselines. The main gain here was achieved after the first iteration;
- we also estimated the changes in the language model perplexity through cross-entropy values computed on the manual word transcriptions of the test corpus HMIHY-3CI-test according to the following formula:

$$\hat{H} = \frac{1}{I} \sum_{i=1}^{I} \frac{1}{|S_i|} \log_2 P(S_i), \tag{6.1}$$

  with sentences $\{S_i\}_{i=1}^{I}$ in this corpus. It turned out that after the adaptation took place, the perplexity dropped from 62.7 (background language model) to 13.5 coming very close to the perplexity of the transcriptions-trained language model estimated at 12.2.

Since the phone-based spoken language understanding is one of the main pillars of this work, we conducted a similar adaptation process for the language model used in the phone recognizer, and compared its results with those obtained using the word-based adaptation. The unsupervised adaptation of the phone language model was performed exactly as described in [Als03]: we started with a simple phone loop of 40 phones and three special symbols (begin-of-sentence, end-of-sentence and empty symbol ε) and proceeded recognizing corpus HMIHY-1 and estimating new phone language model while increasing the order of the model gradually to $n = 4$. The real-time factor measured for the recognition of the test corpus HMIHY-3CI-test with the resulting phone ASR amounted to 0.38, being much lower than the one achieved with the word-based strategy. To evaluate the actual recognition performance, we translated manual word transcriptions onto the phone level using WATSON's built-in text-to-speech (TTS) system. Only one standard pronunciation was allowed for each word, which necessarily led to a rather cautious estimate of the achieved phone accuracy of 69.5%. The estimate is significantly lower than what we could get by converting the word accuracy obtained in the word-based experiments onto the phone level. However, we must bear in mind that the conversion procedure, as described earlier in this paragraph, guarantees the same spelling rules for words no matter whether they occur in manual transcription or in ASR-output. These rules rely heavily on the lexical constraints of the language, while in the case of phone recognizer, its output is more tightly bound to the actual acoustics. We will see in the later sections, how this difference in the phone accuracy afflicted the performance of our phone-based NE-processing algorithms compared to their word-based counterparts.

6.3 Calltype Classification Experiments

In this section we describe the series of calltype classification experiments that we conducted following the blueprints of Chapter 3. We considered two alternative experiment sets of which the first was concerned only with the callers' initial dialog utterances collected in the HMIHY-3CI corpus, and the second counted all utterances available (HMIHY-3CA). The classification of the initial utterances is the most difficult of the two tasks, because of the following reasons:

1. its utterances typically contain more words (cf. Table 6.2), the majority of which occur introductory non-salient phrases like *"Hi, here is Mary Johnson calling. I want to know…"* or *"Yes, my wife and I, we went on holidays last month, and as we came back we noticed that…"*, and thus introduce additional noise in the information extraction task, especially through the higher out-of-vocabulary (OOV) rate;

2. despite having three labels less than HMIHY-3CA, the corpus of initial utterances has a prior semantic distribution less skewed than that corpus, which leads to the calltype perplexity of 20.3 as opposed to 14.6 by all utterances.

6.3.1 Evaluation Procedure

First, we are going to explain our evaluation strategy for calltype classification experiments. We ascertain the quality of classification by means of *Receiver Operating Characteristic* (ROC-) curves. Assume that we have a confidence rejection threshold which, for each utterance, determines whether the calltype with the highest classification score should be passed on to the evaluation procedure as an attempted guess or a rejection should be made (in all our experiments we pull together rejection and the garbage-collector calltype OTHER). For each value of this threshold, we can compute two measures:

- *correct classification rate (CCR)*: proportion of the correctly classified utterances to all utterances that were not rejected;
- *false rejection rate (FRR)*: proportion of the utterances that were falsely rejected to all rejected utterances.

An ROC-curve shows dependency of correct classification rate on false rejection rate when varying the rejection threshold. To obtain a formal vehicle for comparison of two ROC-curves, for each point on the curve we compute the overall *rate of correct decisions* as the proportion of the correctly classified and correctly rejected utterances to the size of the entire test set.

Additionally we evaluate classifier performance in terms of a *confusion matrix* showing how much of each calltype is classified into each calltype. For all our experiments we restrict the evaluation to *rank-one* results, that is, only the calltype with the highest score is considered a valid guess. While deployed SLU-systems may under circumstances profit from one or several ersatz hypotheses as well (cf. system prompt: *Would you like to make a payment or ask for account information?*), we ignore this extension altogether, for its value is only significant for the dialog manager, and our agenda is the classification per se.

6.3.2 Effect of Unsupervised Adaptation for Word-based Strategy

The goal of this section is to show the effect of the unsupervised word-level language model adaptation on the quality of calltype classification. We employed two different classification mechanisms to compare the results achieved with the adapted language model against those from two baselines (cf. Section 6.2). The first classifier is the BOOSTEXTER described in Section 3.2.2 and the second one is the SVM LLAMA introduced in Section 3.2.1. For the boosting approach gappy 5-grams formed the pool of potential weak classifiers, and altogether 300 of them were allowed to be selected[4]. In the case of SVMs, we used classification features that were all n-grams for n up to 5, having the expectation of their number of occurrences in one utterance of at least 10^{-4} (cf. Section 3.2.1). The plots in Figure 6.2 show the ROC-curves for calltype classification using BOOSTEXTER and LLAMA respectively.

It can be seen from these plots that the positive effect of unsupervised language model adaptation on calltype classification is striking indeed. So, when using boosting to execute calltype classification and taking the highest correct decision rate achieved along the ROC-curves as the classifier performance measure, the adaptation allowed to almost entirely compensate for the gap between background language model and language model trained on manual in-domain

[4]We observed that allowing for more rounds to take place wouldn't change the classification results significantly.

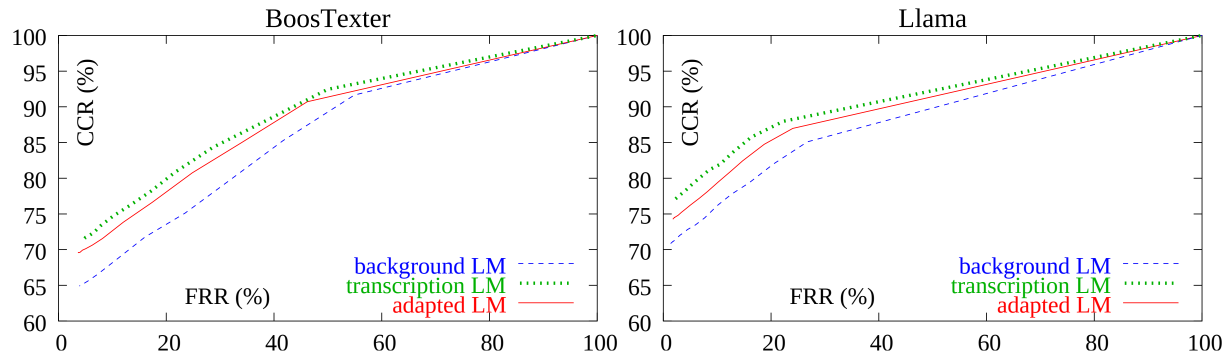


Figure 6.2: Calltype classification results with BOOSTEXTER (left) and LLAMA (right) on ASR-output from corpus HMIHY-3CI using background, transcription-trained and adapted ASR language models.

transcriptions, advancing from the first baseline 63% to about 68% and not reaching the second baseline by about one percent point. Similarly, by the SVM-based calltype classification the difference between background and transcription-trained language models was cut in half with correct decision rates for both baselines and the adopted case at ca. 70%, 76% and 73% respectively.

Furthermore, we observe that the contrast in classification rates between the two classifiers is remarkable. One suggested reason is the LLAMA classifier having better procedure for converting scores to probabilities when working with more than two classes [Haf03a]. Besides, because of a more loosely-coupled feature selection mechanism, LLAMA is friendlier to work with weighted ASR-lattices (see Section 6.3.5). On the other hand, training and test procedures are much faster with BOOSTEXTER, especially when using non-linear SVM-kernels (the latter being of crucial importance for outstanding classification performance).

There is yet another reason why we wanted to keep using boosting for calltype classification despite its inferiority to SVMs. Unlike SVMs that only select and store training examples without saying explicitely which features in these examples they considered useful, boosting shows the n-grams it selected as potential cues for the classification, so that we can analyze these n-grams in terms of their salience. Figure 6.3 confirms our suspicions concerning the usability of the simple salience measures for calltype classification expressed Section 3.1. In this plot, we compared PMAX distribution of all n-grams ($n \leq 5$) occurring more than twice in the training corpus against the corresponding distribution for those of them selected by the BOOSTEXTER. We see that there is not much difference between these distributions; in fact, the proportion of very low salience values is even higher among the n-grams selected by the BOOSTEXTER. Thus, we can

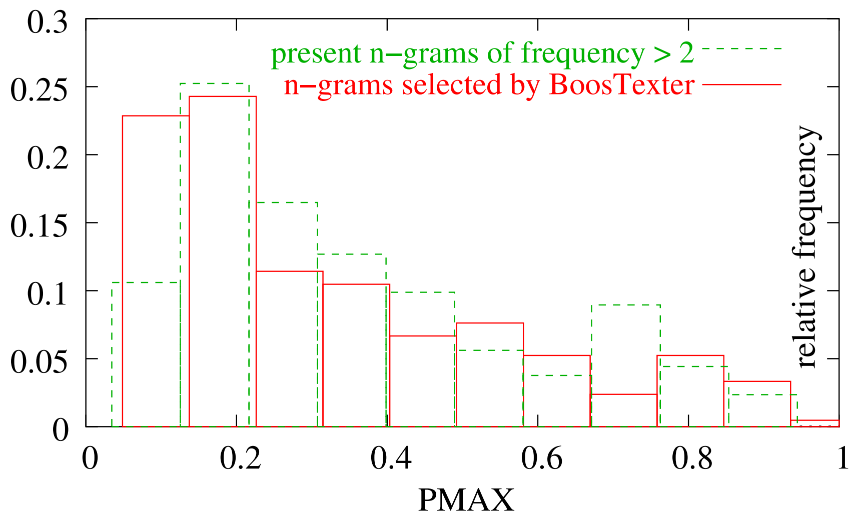


Figure 6.3: Distribution of n-grams by PMAX; comparison between all n-grams ($n \leq 5$) present in the training corpus more than twice and those selected by BOOSTEXTER.

conclude that the independently measured salience does not play a crucial role for classification. We believe that what matters is the conditional salience, like for instance, the salience of an n-gram given that another n-gram is *not* present in the utterance.

Similar classification experiments were conducted on the HMIHY-3CA corpus. Their results are shown in Figure 6.4. First of all, our assumption of an easier task could be confirmed: in the area of low false rejection rates, ROC-curves on HMIHY-3CA lie on the average ten percent points above those on HMIHY-3CI. In fact, about 30% of all utterances in the HMIHY-3CA corpus belong to the calltype YES (confirmation of system's suggestions) and, being usually realized by only a few affirmative words, are exceedingly easy to recognize.

Furthermore, once again it could be concluded on the usefulness of the unsupervised language model adaptation. The highest correct decision rates on the transcription-trained and adapted ASR language models estimated by ca. 83% and 82% respectively, were standing two to three percent points better than by the background language model.

In Table 6.5 a confusion table for the ten most frequent calltypes in the corpus is put together. All results in this table have been achieved with the adapted language model. The strong diagonal indicates good classification results for at least these frequent calltypes. Due to the high number of rare calltypes, however, the overall class-wise average of the correct decision rates amounts to only 56%. It is also intuitively clear that particular difficulties were encountered by trying to recall calltype OTHER which plays the role of a collector for all requests that could not be assigned to any of the other categories.

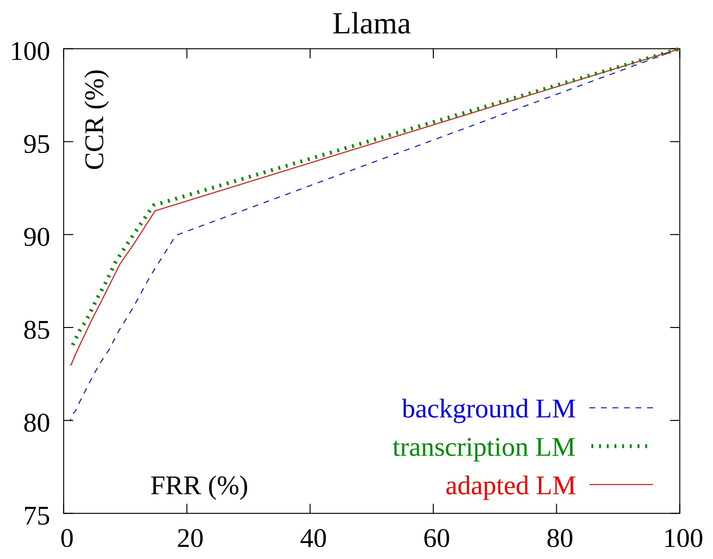


Figure 6.4: Calltype classification results with LLAMA on ASR-output from corpus HMIHY-3CA using background, transcription-trained and adapted ASR language models.

6.3.3 Further Improvements

One procedural innovation that we decided to try out for the calltype classification task, consisted in an additional filtering of the input utterances. In fact, since quite a few non-initial utterances contained named entities, we deemed it helpful to replace some of the numerical and date-related words by special category tokens beforehand. This pre-filtering should bring in the advantage of reducing the entropy of the corpus, while preserving the meaningful information contents. Intuitively, we would expect that if, say, *sixteenth* appears to be salient for some calltype, so will be *seventeenth* too, and by replacing them all with one special token we would reduce the number of relevant classification features. Table 6.6 displays all categories created. An example for the nonterminal $month_s is *"I got my April's bill"*.

Also, we decided to test if any extra-linguistic information is capable of further improving the classification results. One noise-free source of side-information for calltype classification comes from the interactive nature of the domain. As was previously stressed in Section 2.2.1, each information the systems gets from the user is elicited by a system prompt within a dialog framework. In the beginning of the dialog, this prompt can allow a broad range of possible answers and therefore of calltypes to expect; however, as the dialog successfully progresses the wide spectrum gets narrowed down, and so the perplexity of the posterior calltype distribution drops. Obviously, this observation doesn't spread on the HMIHY-3CI corpus, since it contains only user utterances following the initial (re-)prompt. Different from that, the HMIHY-3CA

hypothesis / reference	ACCOUNT_BALANCE	BILLING_SERVICES	CUSTOMER_REP	EXPLANATION_OF_BILL	GENERAL_SERVICES	NO	OTHER	PAY_BILL	UNRECOGNIZED_NUMBER	YES	recall
AB	**86.8**	3.0	0.0	1.4	0.6	0.3	0.9	1.9	0.9	0.6	0.9
BS	3.1	**85.2**	0.2	1.6	0.8	0.8	0.8	0.2	0.6	1.8	0.9
CR	0.1	0.1	**96.9**	0.1	0.7	0.2	0.2	0.0	0.3	0.8	1.0
EOB	5.2	3.1	0.7	**57.7**	1.3	0.0	1.8	0.8	8.8	0.5	0.6
GS	2.1	2.7	4.2	1.5	**54.4**	0.8	3.8	1.2	2.3	5.2	0.5
NO	0.6	0.0	0.4	0.1	0.9	**90.5**	0.2	0.4	0.6	3.4	0.9
O	5.0	4.2	2.3	5.6	3.4	1.3	**39.1**	1.4	7.5	5.6	0.4
PB	4.9	3.1	0.9	0.9	0.8	0.8	0.4	**81.3**	0.7	0.9	0.8
UN	0.1	1.5	1.2	2.2	0.1	0.0	2.4	0.3	**85.1**	0.3	0.9
YES	0.1	0.1	0.0	0.2	0.1	0.5	0.2	0.0	0.3	**97.8**	1.0
precision	0.9	0.8	1.0	0.7	0.8	1.0	0.8	0.9	0.8	1.0	

Table 6.5: Excerpt from the confusion table: classification results (in %) for the ten most frequent calltypes; adapted language model. In the reference column calltype abbreviations are used.

corpus accounts for all turns of the dialog; therefore, it seems reasonable to use the preceding system prompt as an additional source of information for classification of utterances from this corpus.

Figure 6.5 shows the results of implementing both of the ideas above. We see that using pre-filtering with category tokens and introducing system prompt as a supplemental classification feature did pay off by lifting the ROC-curve by about one percent point (maximal correct decision rate rose to ca. 83%). At the same time, the improvement didn't turn out as large as one might rightly expect. We explain it by arguing that about 30% of all user utterances in HMIHY-3CA are responses to the initial prompt or to the re-prompt, and a large proportion of the rest are confirmations or negations (about 55%) which are typically easily recognized just based on their wording.

We experimented with other potential classification features, like utterance length, as well, but they didn't result in any significant improvement of classification rates.

category	member words
\$month	january february march april june july august september october november december
\$digit	one two three four five six seven eight nine zero
\$teen	ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen
\$ten	twenty thirty forty fifty sixty seventy eighty ninety
\$month_s	january's february's april's may's june's july's september's october's november's december's
\$digit_th	first second third fourth fifth sixth seventh eighth ninth
\$teen_th	tenth eleventh twelfth thirteenth fourteenth fifteenth sixteenth seventeenth eighteenth nineteenth
\$ten_th	twentieth thirtieth

Table 6.6: Word categories introduced in the pre-filtering stage, before classification.

6.3.4 Unsupervised Adaptation for Phone-based Strategy

In the case that not even a background language model at the word level is available, but rather the phone lexicon (with corresponding acoustic models) of the language, calltype classification must be attempted on the raw output of a phone recognizer. The adaptation model for the latter was suggested in [Als03] and restated in Section 2.4.3. With the phone language model obtained in this way, we examined the performance on the ASR phone transcriptions of the HMIHY-3CI corpus. The results shown in Figure 6.6 prove that calltype classification can be carried out without upstream word recognition, and one doesn't have to reconcile this omitted step with any loss in performance. On the contrary, at some places the ROC-curve for the phone case is even higher than for the word case, although both have approximately the same highest achievable correct classification rate lying between 73% and 74%. We consider it a very encouraging fact, for it opens the way for quickly readjustable spoken language understanding systems (at least for those rooting in topic classification), requiring minimum training effort, and, in particular, not dependent on the cumbersome manual transliteration of thousands upon thousands of spoken utterances.

6.3.5 Classifi cation on Lattices

Finally in this section, we explore possible advantages of calltype classification when conducted on lattices as opposed to simple best paths. Lattices preserve much more information from the ASR regarding possible transcript alternatives. While best paths, as the name says, contain only one best hypothesis, lattices incorporate a number thereof, and encode them in an efficient way

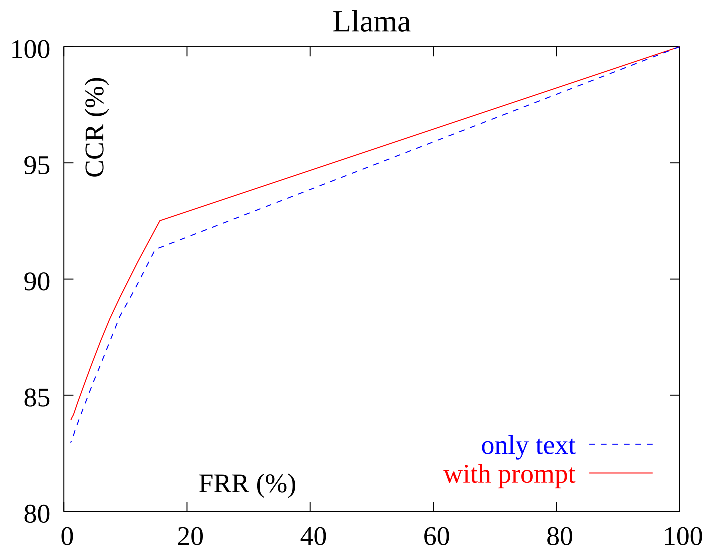


Figure 6.5: Calltype classification with system prompt as an additional information source; results on HMIHY-3CA using LLAMA-classifier.

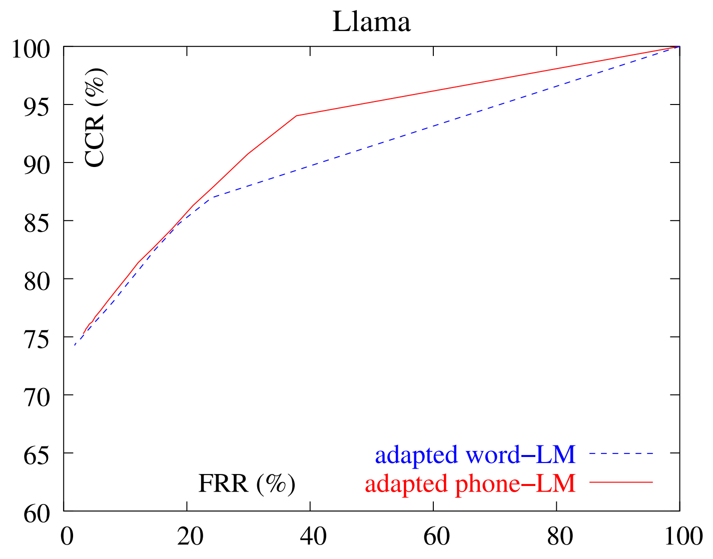


Figure 6.6: Calltype classification with adapted word and phone language models; results obtained on the HMIHY-3CI corpus using LLAMA-classifier.

as weighted directed graphs. Like the best paths, they also can be straightforwardly encoded as finite state machines.

While incorporating weighted word-lattices in BOOSTEXTER is not trivial, LLAMA's feature extraction mechanism is flexible enough to allow counting and collecting n-gram features from the lattices as well. However, by doing so, we increase dramatically the space and time requirements of the algorithm, since now there are much more n-grams contained in each particular ASR-output. While it is sometimes recommended to smooth the probability distribution over all possible paths through a lattice by multiplying all arc costs (recall that these costs are computed as $-\log(\text{prob})$) by a real-valued factor between 0.0 and 1.0 [Haf03a], this smoothing additionally increases the number of classification features. Thus, by switching from best paths to lattices, the dimension of the feature vectors grows from approximately nine hundred thousand to about twenty millions, becoming quite a challenge even for the high-dimension friendly support vector machines.

In terms of the overall classification rates no significant improvement could be achieved neither for words nor for phones, the only positive effect attained being class-wise averaged classification rates rising by 1% to 1.5% percent points for words and phones respectively.

6.4 Named Entity Detection Experiments

For our named entity detection experiments we consider the HMIHY-3N corpus and the three named entities from Table 6.3. In compliance with the task description in Section 4.2, the detection goal for each utterance is to determine whether it contains at least one instance of these three named entity types. We employed the detection-by-classification scheme introducing for each named entity one special class to indicate its presence and also one rejection class which stands for the absence in the utterance of any of the known named entities at all. By comparison of the classification scores of the NE-bound classes p^k (cf. (4.2), page 60) with some empirically chosen threshold θ, we are able to make separate assertions about the presence of each named entity type. Before presenting the detection results we would like to recall once again how unbalanced our data is with respect to named entities: each named entity occurs only in about 3% of all utterances.

6.4.1 Evaluation Procedure

According to the three subtasks we distinguished in the NE-processing in Chapter 4, three different evaluation procedures have been designed. To ascertain the detection results we use the following simple scheme: four different outcomes are possible for each utterance and each NE-type:

- *hit*:
  utterance contains at least one NE-instance, and the algorithm decides that there are some;
- *miss*:
  utterance contains at least one NE-instance, but the algorithm decides that there is none;
- *false positive*:
  utterance contains no NE-instances, but the algorithm decides that there are some;
- *correct negative*:
  utterance contains no NE-instances, and the algorithm decides that there is none.

From these four outcomes averaged over the entire test corpus we can assemble two measures commonly used for the information extraction tasks: *precision* P and *recall* R:

$$\begin{cases} P &= \dfrac{\#\text{hits}}{\#\text{hits} + \#\text{false positives}} \\ \\ R &= \dfrac{\#\text{hits}}{\#\text{hits} + \#\text{misses}} \end{cases} . \tag{6.2}$$

In the detection task, we also plot ROC-curves to evaluate the detection algorithms, however in this case, it reflects the dependency of precision and recall, when varying the detection threshold θ. The ROC-curve is usually plotted in the $P \sim 1 - R$ coordinates and its convexity is one visually perceptible criterion of the detection performance. Throughout this chapter we will also use the term *F-measure*, a uniformly weighted harmonic mean of precision and recall, which "encodes" the goodness of the classifier at some particular point on the ROC-curve[5]:

$$F = \frac{2PR}{P + R}. \tag{6.3}$$

Using the F-measure we can compare two ROC-curves "at their best" similar to the classification task: for each curve a point with the highest value of F is selected, and then these maxima are compared against each other whereby the classifier with the highest value is declared the winner. We will refer to the points on ROC-curves where these maxima are achieved as *operating points*.

[5]In the absence of any particular application-specific recommendations, we adopt the symmetric cost function providing errors in precision and recall with the same degree of severity.

6.4.2 Effect of Unsupervised Adaptation for Word-based Strategy

As earlier in Section 6.3.2, we compare the detection results based on the output of an ASR that uses three different language models: generic background LM, transcription-based LM and the background language model adapted in an unsupervised manner. Again, we assess the detection success of two alternative large margin classifiers: BOOSTEXTER (boosting) and LLAMA (SVM). The classifiers are trained on the HMIHY-3N-train and detection is tested on the HMIHY-3N-test corpora (see Section 6.1). The plot in Figure 6.7 shows detection ROC-curves for named entities DATE, ITEM_AMOUNT and PHONE_NUMBER from Table 6.3.

First of all, we notice that the difficulty of detection is not constant for different named entities. Thus, phone numbers are much easier to detect than dates and context-specific monetary expressions. We explain it in the first place by the higher length of the former (a typical phone number in US has 7, 10 or 11 digits) compared to just one word for many dates (the month name), and also by the context-dependency of the named entity ITEM_AMOUNT (cf. example on page 58). Comparatively low recall by DATE and precision by ITEM_AMOUNT may also corroborate these assumptions[6]. Secondly, similar to the calltype classification task, support vector machines consistently outperformed boosting by about 10%, but they required an order of magnitude more time for training and classification.

As to the unsupervised adaptation, we see that its overall positive effect could be observed also in the named entity detection experiments. At the same time, a detailed insight in the individual named entities reveals that they didn't profit equally from it. From the plots and also from Table 6.7, expressing the characteristics of the LLAMA-based detection algorithms in terms of precision and recall of the corresponding operating points, it can be seen that PHONE_NUMBER was the actual winner. While not having much problems detecting it even on the ASR-output with the SWITCHBOARD-trained background language model, the detection mechanism was almost as good on the output of the adapted ASR as in the case of transcription-trained language models. However, only a relatively slight improvement could be achieved for named entity ITEM_AMOUNT and BOOSTEXTER even performed much worse on the adapted language model for DATE.

We believe, the reason for the disparity in the gains through language model adaptation by calltype classification and named entity detection tasks comes from the more prominent locality of the named entity phenomena. While the calltype motif can frequently be pursued throughout

[6]This is a somewhat daring inference though, since a particular constellation of precision and recall in the operating point can turn out rather haphazard if the ROC-curve is more or less isothermal with respect to the F-measure.

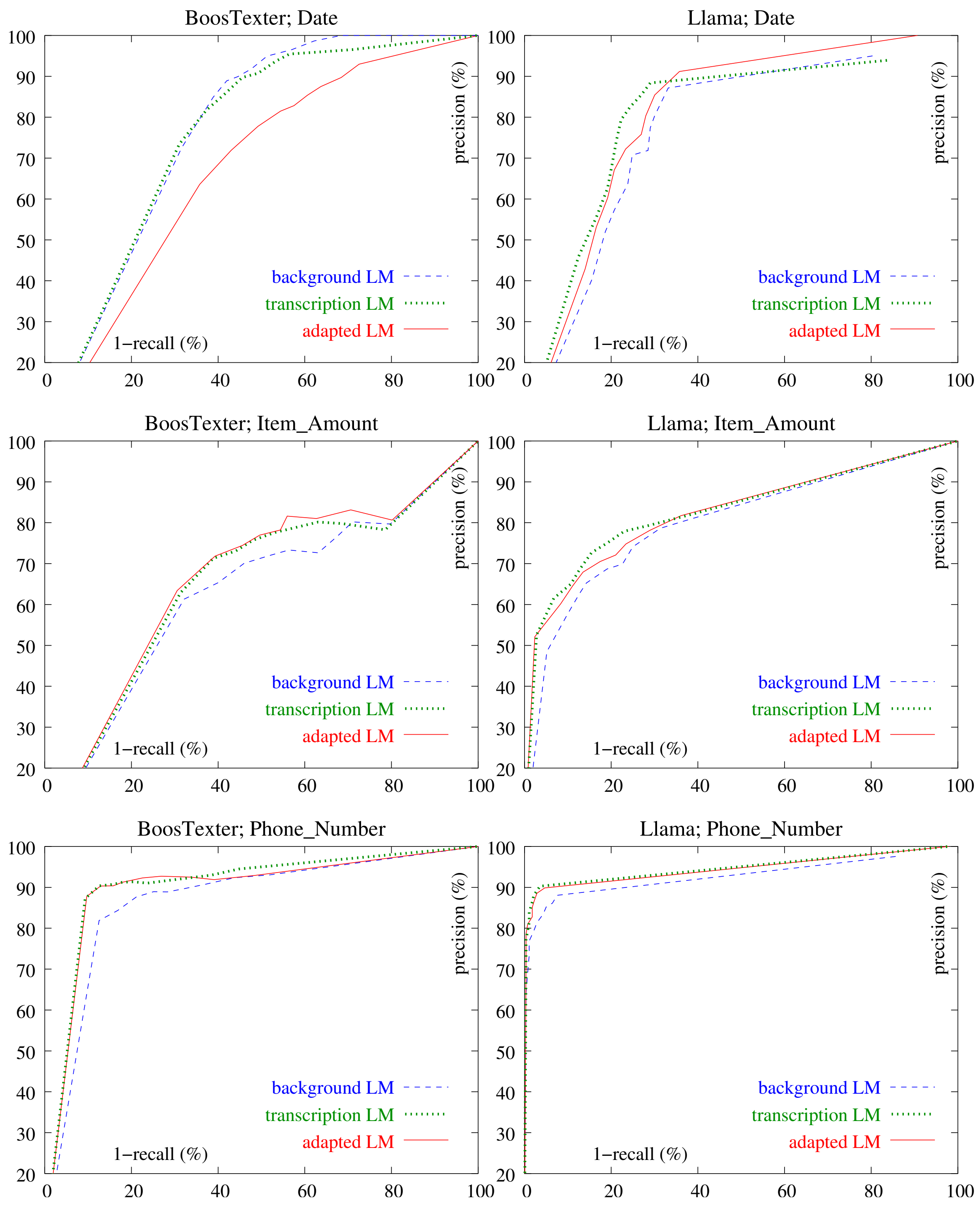


Figure 6.7: Comparison of detection ROC-curves for background, adapted and transcription-based language models, using BOOSTEXTER and LLAMA classification engines.

	DATE			ITEM_AMOUNT			PHONE_NUMBER		
	P(%)	R(%)	F	P(%)	R(%)	F	P(%)	R(%)	F
background LM	87	67	0.76	74	75	0.75	88	92	0.90
adapted LM	85	70	0.77	68	87	0.76	89	97	0.93
transcription LM	82	76	0.79	73	85	0.78	90	96	0.93

Table 6.7: Operating points characteristics for LLAMA-based detection of the three named entities using ASR-output obtained with adapted and two baseline language models.

the entire utterance (e.g. calltype UNRECOGNIZED_NUMBER in *"There is this phone number on my last bill which I don't recognize"*), named entities are often indicated by just their wording (e.g. *"I don't recognize this call on my* [September] *bill statement"*), which leaves this wording as the only detection cue for the classifier. If misrecognized, however, the chance of recovery is small. This is why the recognizer has to achieve a higher word accuracy than for calltype classification to guarantee that a sufficient amount of keywords is correctly recognized.

The following statistics support our assumption: for calltype classification and named entity detection, we selected all words whose salience values (PMAX) lay above threshold 0.5. Then, for each task (and the corresponding list of its salient words) we computed one check value indicating how well these words were recognized by the ASR using the adapted language model. The check value we employed was similar in its nature to the traditional word accuracy, but yet again expressed as F-measure based on precision and recall values. Both of the latter were computed through an alignment of manual transcriptions and ASR-output followed by a consecutive mapping of the salient words found in one corpus representation into the other, and by an analysis of the mapping results. It turned out that recognition of salient words for calltype classification ($F = 0.84$) surpassed the one for salient words for named entity detection ($F = 0.78$). This fortified our suspicion that much more significant ASR-improvement is needed to achieve a similar improvement by the NE-detection as was obtained by the calltype classification.

There is yet another reason for the comparatively weak performance on the named entity DATE which we will elaborate in more detail in the section dedicated to NE-localization.

6.4.3 Possible Extensions

The pre-filtering of the utterances comprised in Table 6.6 and employment of other information sources, such as system prompt, is certainly applicable for named entity detection as well as for calltype classification earlier. Table 6.8 summarizes the improvement in detection rates caused by the combination of both extensions. It can be seen that the overall F-measure of the detection

	DATE			ITEM_AMOUNT			PHONE_NUMBER			overall
	P(%)	R(%)	F	P(%)	R(%)	F	P(%)	R(%)	F	F
baseline	85	70	0.77	68	87	0.76	89	97	0.93	0.83
extended	84	70	0.77	78	79	**0.79**	87	99	0.93	**0.84**

Table 6.8: Named entity detection with system prompt as an additional information source; results obtained using LLAMA-classifier.

was raised by about 6% (relative), mostly due to the better results for the difficult named entity ITEM_AMOUNT.

The detection using n-gram features extracted from word-lattices was also tried out (combined with the prompt as an additional information source). It did bring a good improvement for the named entity DATE (F-measure rose to 0.81), while other named entities were detected as before. At the same time, the time and space requirements of the classifier increased drastically.

6.4.4 Unsupervised Adaptation for Phone-based Strategy

Unlike calltype classification, the detection experiments carried out at the phone level resulted in lower ROC-curves and F-measures in their operating points compared to the word-based NE-detection (see Figure 6.8), especially for the shorter named entity DATE. This results remained consistent while using boosting and support vector machines alike. Again, for now we explain it by the locality of the named entities. Nonetheless, phone number detection turned out to function very well, and detection of monetary expressions printed on the bill resulted in reasonable detection rates as well.

6.5 Named Entity Localization and Value Extraction Experiments

We now come to the next part of our experiments. In this section we will talk about instantiation of named entity fragments in the ASR-output and extraction of values from the localized instances. The grammars for the considered named entities were manually designed and realized in form of regular grammars, whose exact definition can be looked up in Appendix C. Then, the grammars were converted into finite state machines which were used for actual localization and value extraction. For the phone-based variant, a text-to-speech system was used to re-express

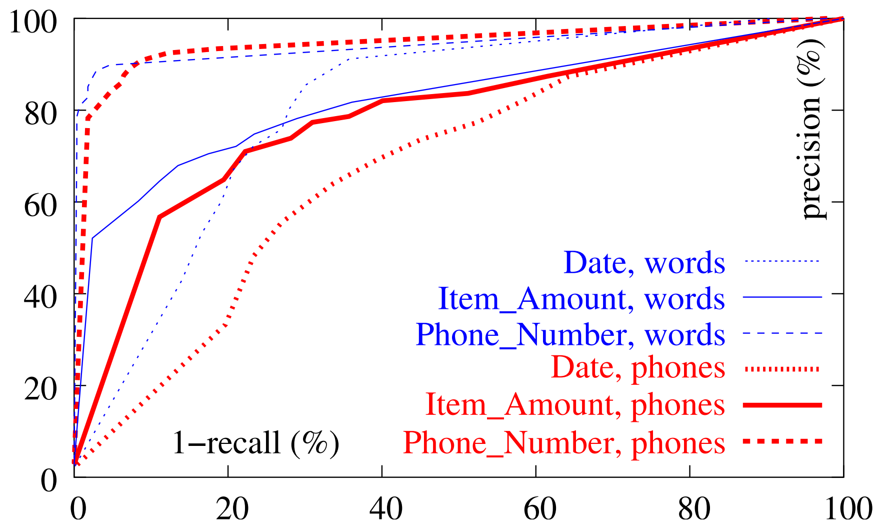


Figure 6.8: Comparison of LLAMA-facilitated detection ROC-curves for adapted word-based and phone-based language models; F-measures in the operating points of ROC-curves for phones: ITEM_AMOUNT 0.74, DATE 0.65, PHONE_NUMBER 0.91.

each word as a sequence of phones, whereas only the most probable pronunciations were taken into account.

Throughout the experiments in this section, we stick to the maximum-likelihood parsing mechanism described in Section 4.3.2, with optional enhancements presented in the other parts of Section 4.3. We ascertain the possible gains of the unsupervised language model adaptation, of pre-filtering by detection and of the approximate matching strategy that we elaborated for the maximum-likelihood parsing. The NE-localization and value extraction results will be considered in parallel to enable a comprehensive judgment of the effect of various algorithm parameters. The tests were conducted on the HMIHY-3N-test corpus, and for the optional detection pre-filtering we took the detection results from the previous section.

6.5.1 Evaluation Procedures

We will start by formulating the evaluation techniques as usual. Evaluation of localization results is principally done in the style of MUC [Chi97]. Since named entities are localized on the ASR-output and their references are provided for manual transcriptions, we align ASR-output and manual transcriptions word-wise beforehand, as suggested in [Kub98]. The NE-positions in the ASR-output are then remapped onto transcriptions. In general, there is no guaranty that remapping will be always correct. This is why we decided to aim for a kind of localization,

a) DATE

reference	*fourth of July seventeen seventy six*								
normalized ref.	0	4	**.**	**J**	**.**	**1**	7	7	**6**
normalized hyp.			**.**	**J**	**.**	**1**	0	0	**6**
hypothesis	*July thousand six*								

b) PHONE_NUMBER

reference	*one two three forty five sixty seven eight hundred*													
normalized ref.	**1**	**2**	**3**		**.**	4	**5**	**6**	7	**8**	**0**	**0**		
normalized hyp.	**1**	**2**	**3**	4	**.**		**5**	**6**		**8**	**0**	**0**	0	0
hypothesis	*one two three four five sixty eight oh oh oh oh*													

Table 6.9: Examples of alignment of normalized named entity values; a) date expressions, b) phone numbers.

where the reference and hypothesis must at least overlap to produce a *match*, as opposed to a more rigorous rule of correct localization, by which both boundaries must coincide. In both cases the types of the reference and of the hypothesis must be the same; for each hypothesis at most one reference is allowed to be aligned to it and vice versa. If there are several possible matches, the one with the smallest difference in values between hypothesis and reference is selected first[7]. Again, we ascertain the goodness of localization in precision P, recall R and the F-measure derived from the two. Also, ROC-curves can be plotted in the $(P, 1 - R)$-axes. Note that a match during localization can be interpreted as a hit for detection, but not the other way round.

In order to make a decision about the exact steps of the evaluating procedure for named entity value extraction, one has to bear in mind the goal of the value extraction. In many tasks the requests refer to NE-objects that are accessible for the system through its internal databases. For instance, in HMIHY task the caller is likely to mention locations, phone numbers, charges etc. that have to do with his recent calling history:

> *"I don't know anyone in Austin, Texas with the number * * * * * * * * * *. There is a call on August 3^d to this number..."*

In this contrived example the system has a list of calls charged to the client's number at its disposal and is to select one of them or, if none of them was likely to be meant, come up with an alternative. In this context, the *distance* between a hypothesis and a reference is what really matters. Alternatively, we can speak of *degree of correct retrieval* which is reciprocal to the distance.

How are these measures to be defined? First of all, successful localization is a necessary prerequisite for an attempt of value extraction. Given that this condition is fulfilled for some

[7]Extracting NE-values and computing distances between them is described below.

reference/hypothesis pair, both are parsed with a special grammar to produce a normalized form, which is specific to the particular NE-type and allows for a decomposition of each NE-instance in a sequence of basic information units (we will call them *meaning units*), e.g. digits in case of phone numbers. If a named entity consists of several parts, each meaningful in itself (e.g. month, day and year), their boundaries also represent valuable bits of information and are encoded as meaning units as well. So, in the example illustrated in Table 6.9a, both reference and hypothesis, represent normalized forms for named entity DATE and are written in format: *DD.M.YYYY* (the month coding here can be understood as one "12-al digit").

After the normalized forms have been obtained, they are aligned using the DTW-algorithm[8]. For each reference/hypothesis pair $(\mathbf{ref}, \mathbf{hyp})$, the results of the alignment are interpreted in terms of precision $P(\mathbf{ref}, \mathbf{hyp})$ (proportion of meaning units in the hypothesis which are aligned by identity) and recall $R(\mathbf{ref}, \mathbf{hyp})$ (percentage of correctly recognized meaning units in the reference). In the example above, the alignment of $\mathbf{ref}$ and $\mathbf{hyp}$ will produce:

$$P(\mathbf{ref}, \mathbf{hyp}) = 5/7 \qquad R(\mathbf{ref}, \mathbf{hyp}) = 5/9. \tag{6.4}$$

Again, we can also consider an F-measure for the pair:

$$F(\mathbf{ref}, \mathbf{hyp}) = \frac{2PR}{P+R} = \frac{2 * \frac{5}{7} * \frac{5}{9}}{\frac{5}{7} + \frac{5}{9}} = 0.625. \tag{6.5}$$

Analog, for the example in Table 6.9b dealing with normalized reference and hypothesis of named entity PHONE_NUMBER, we obtain:

$$P(\mathbf{ref}, \mathbf{hyp}) = 9/12 \qquad R(\mathbf{ref}, \mathbf{hyp}) = 9/11 \qquad F(\mathbf{ref}, \mathbf{hyp}) \approx 0.783. \tag{6.6}$$

To evaluate the algorithm performance, however, we must switch from the utterance level to the corpus level. The corpus-based statistics for each NE-type are computed as follows:

1. consider the space of all present references $\mathcal{R}$ and the space of all made hypotheses $\mathcal{H}$;

2. for each hypothesis $\mathbf{hyp} \in \mathcal{H}$ take, if present, its reference-match from the localization stage $\mathbf{ref} = \text{match}(\mathbf{hyp}) \in \mathcal{R}$ and compute $P(\mathbf{hyp}) := P(\mathbf{ref}, \mathbf{hyp})$. If the reference-match doesn't exist, set $P(\mathbf{hyp}) := 0$;

3. for each reference $\mathbf{ref} \in \mathcal{R}$ take, if present, its hypothesis-match from the localization

[8]In general, the alignment mechanism must be the same as the one used in search through the database, so that each particular insertion/deletion/substitution may obtain unique costs.

	DATE		ITEM_AMOUNT		PHONE_NUMBER		OVERALL	
	loc.	val.extr.	loc.	val.extr.	loc.	val.extr.	loc.	val.extr.
background LM	0.71	0.66	0.52	0.44	0.61	0.47	0.60	0.51
adapted LM	0.63	0.59	0.56	0.50	0.86	0.77	0.69	0.62
transcription LM	0.74	0.70	0.61	0.57	0.85	0.76	0.73	0.68

Table 6.10: Effect of unsupervised language model adaptation on the F-measure for named entity localization and value extraction experiments.

stage $\mathbf{hyp} = \text{match}(\mathbf{ref}) \in \mathcal{H}$ and compute $R(\mathbf{ref}) := P(\mathbf{ref}, \mathbf{hyp})$. If the hypothesis-match doesn't exist, set $P(\mathbf{ref}) := 0$;

4. compute overall precision and recall:

$$P := \frac{1}{|\mathcal{H}|} \sum_{\mathbf{hyp} \in \mathcal{H}} P(\mathbf{hyp}); \quad R := \frac{1}{|\mathcal{R}|} \sum_{\mathbf{ref} \in \mathcal{R}} R(\mathbf{ref}); \tag{6.7}$$

5. compute F-measure according to (6.3).

6.5.2 Effect of Unsupervised Adaptation for Word-based Strategy

The main thrust of our localization and value extraction experiments conformed to the three-step diagram depicted in Figure 4.1, page 56. Yet, to answer the first question of how the unsupervised adaptation influences named entity localization and value extraction, we omitted the stage of pre-filtering by detection scores and also committed ourselves to ML-parsing with only exact matching allowed. This means that as the string distortion probability $P(s^t|f^t)$ from decomposition (4.6) a simple Kronecker delta function was employed:

$$P(s^t|f^t) = \begin{cases} 1 & \text{if } s^t = f^t; \\ 0 & \text{otherwise}. \end{cases} \tag{6.8}$$

Figure 6.9 illustrates the effect of language model adaptation on this simplest version of named entity localization, and Table 6.10 shows the localization F-measures in the corresponding operating points along with the F-measures for the consecutive value extraction.

It can be seen that for two named entities out of three, language model adaptation resulted in substantial improvement of both localization and value extraction rates. In the case of phone numbers, the achieved performance even turned out to slightly outperform the case of transcrip-

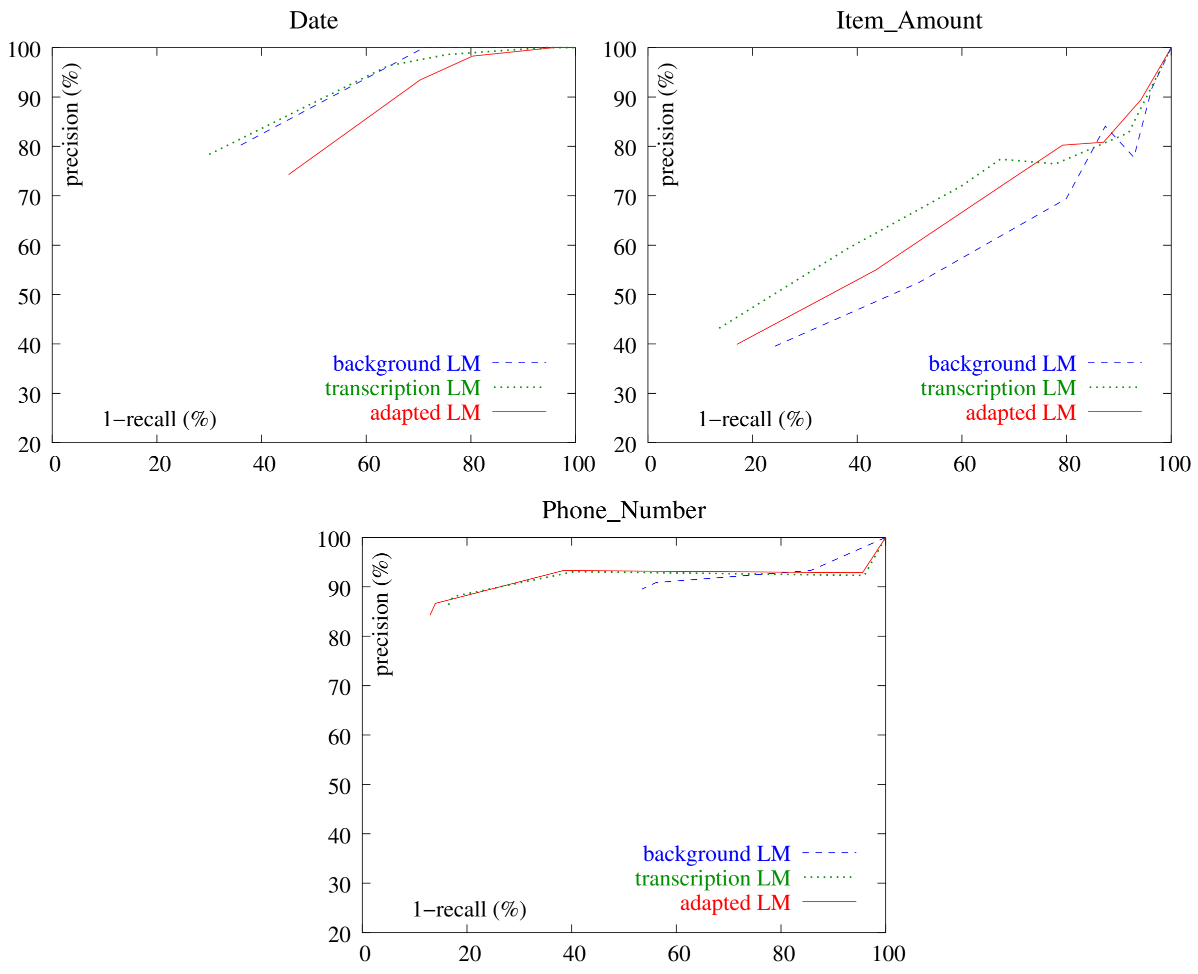


Figure 6.9: Comparison of NE-localization ROC-curves for background, adapted and transcription-based language models, using the simple version of the NE-localization algorithm.

tion trained language models. At the same time, the adaptation led to a dramatic plunge of the localization and value extraction rates for named entity DATE. This truly surprising outcome kept us rather puzzled until we discovered that the reason for this behavior lay in the time disparity between the test and adaptation corpora. Indeed, our definition of named entity DATE has only one obligatory part, which is the month's name. While the overall word accuracy measured on the HMIHY-3N corpus (and in particular the recognition of the words from the named entity fragment representing DATE) increased after adaptation[9], the exact opposite was observed with respect to months' names. To investigate this phenomenon we plotted the relative frequencies

[9]To ascertain the recognition quality of individual words we used the same F-measure-based mechanism as on page 130.

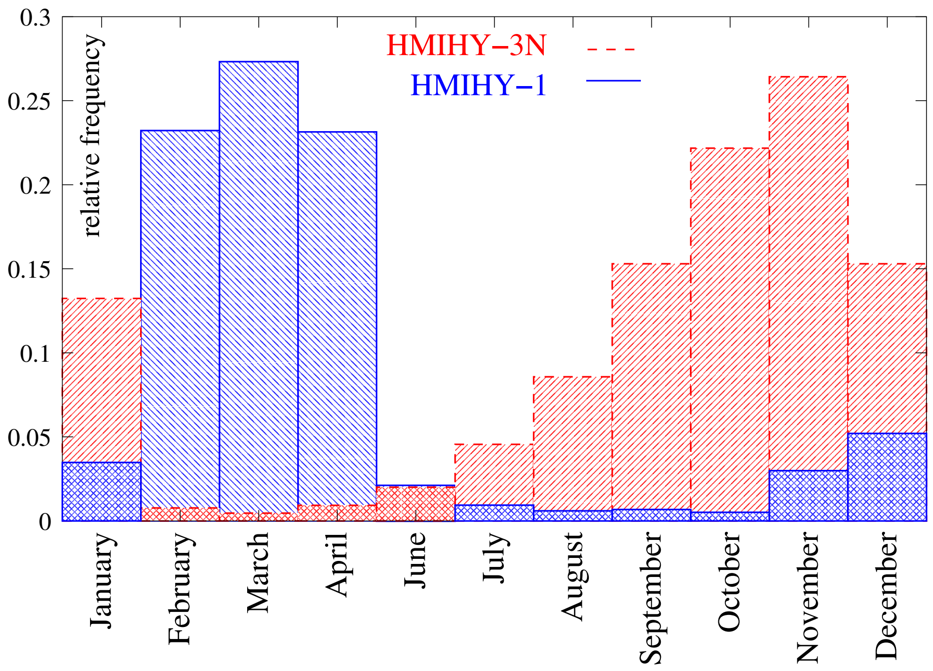


Figure 6.10: Occurrence statistics of months (except May) for the adaptation corpus HMIHY-1 and the HMIHY-3N corpus.

of the months in HMIHY-3N and adaptation corpus HMIHY-1 (see Figure 6.10), and discovered that the histograms were complementary for the most part. A-posteriori, we found out that the adaptation corpus was recorded several months before HMIHY-3N. Since the callers usually want to talk about recent events, there happened to be a natural shift between the month references in the dialogs from these two corpora. Accordingly, regarding date expressions, the HMIHY-1 corpus was not "in-domain", as required for the adaptation in Section 2.4.2, but rather "out-of-domain". Thus, a correct adaptation couldn't take place for date expressions which led to the noticeable decrease in localization and value extraction rates for named entity DATE. Remarkably, the negative adaptation effect on the detection of this named entity didn't come out that prominent, for, as pointed out in Section 4.2, the classification mechanism we used for detection relied not only on the wording of the named entity itself, but on a variety of other textual and non-textual indicators.

6.5.3 Detection Filter

Let us now turn to the effect of the NE-detection step on the results of the consecutive localization and value extraction. Earlier in Section 4.3.6 we have already alluded to the potential use of a fast named entity detection algorithm for the downstream localization and value extraction. Now we are going to underpin this statement by showing how these two mechanisms can be hooked up in practice. In formula (4.2), page 60 we have seen that the information gained after the detection-by-classification step from Section 4.2 is a vector of scores reflecting the presence probabilities for each named entity type, as well as the probability that none of the known named entities is present at all. Furthermore, we set up a threshold that the scores of a named entity must exceed in order for this named entity to be considered present.

The general idea from Section 4.2 was to attempt localization of a named entity only for those sentences in which this named entity was previously detected. There are two motivators for this strategy:

1. **time factor:** localization involves maximum-likelihood parsing which can be computationally intense; when detection is fast (which is the case with boosting or linear kernel SVMs), the time savings can be considerable;

2. **better performance:** especially in the case of context-specific named entities like ITEM_AMOUNT preceding detection is the only way to avoid false positives. In other words, from the detection pre-filtering we expect to prune away many of false positives otherwise unheeded, and consequently a better localization precision. Similarly, for our misadapted named entity DATE, the words within the named entity itself cease to be reliable predictors of its presence, but others indicators must be granted more importance.

Our implementation uses a softer version of filtering by detection. Instead of a simple thresholding, we provide each named entity type for each utterance with scores, obtained by means of a logistic regression from the corresponding detection scores. These scores, lying between 0.0 and 1.0, are then introduced in the ML-parsing formula (4.6), page 67 as an additional penalizing multiplicand in the language model term.

In particular, with K considered named entity types we set the threshold θ from (4.2):

$$\theta = T/\hat{K}, \text{ with } T = p^{\text{rej}} \text{ and } \hat{K} = 2K, \tag{6.9}$$

and use the empirically optimized circumcised sigmoid function centered around θ (see Figure 6.11) to convert the detection score into a confidence score for localization. Assume now that a named entity's detection score in an utterance is p^k. Then its confidence score for localization in this utterance will be:

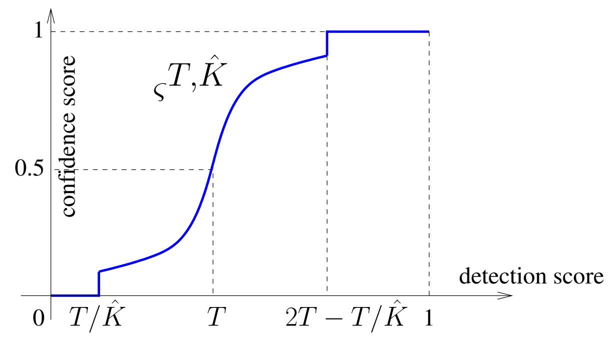


Figure 6.11: Using logistic regression to converting detection score into confidence score for localization.

	DATE	ITEM_AMOUNT	PHONE_NUMBER
no detection filter	0.59	0.50	0.77
with detection filter	**0.62**	**0.53**	0.76
with detection oracle	0.65	0.67	0.81

Table 6.11: Effect of pre-filtering by detection on named entity value extraction; impact on the F-measure.

$$\varsigma^{T,\hat{K}}(p^k) := \begin{cases} 0, & \text{if } p^k < T/\hat{K} \\ 1, & \text{if } p^k > 2T - T/\hat{K} \\ 1.0/(1 + \exp\frac{2\hat{K}(T-p^k)}{T(\hat{K}-1)}), & \text{otherwise} \end{cases} \tag{6.10}$$

The effect of such pre-filtering on the word-level localization and value extraction with the adapted language model (second line of Table 6.10 as the new baseline) was the following: with the additional multiplicand coming from (6.10) integrated in (4.6), we obtained significantly better localization and value extraction results for the "difficult" named entities DATE and ITEM_AMOUNT with a marginal loss by value extraction for PHONE_NUMBER. The changes in the localization ROC-curves for all three named entities are demonstrated in Figure 6.12, and the corresponding comparison of the F-measures in the operating points for named entity value extraction is given in Table 6.11. For these comparative experiments we created a second baseline which shows what localization and value extraction results could be achieved in the hypothetical case of a perfect detection procedure (we call it *detection oracle*). Not surprisingly, the most dramatic improvement owing to the detection oracle was observed for the context-specific named entity ITEM_AMOUNT.

Similar improvement could be observed on the output of the ASR with the other language

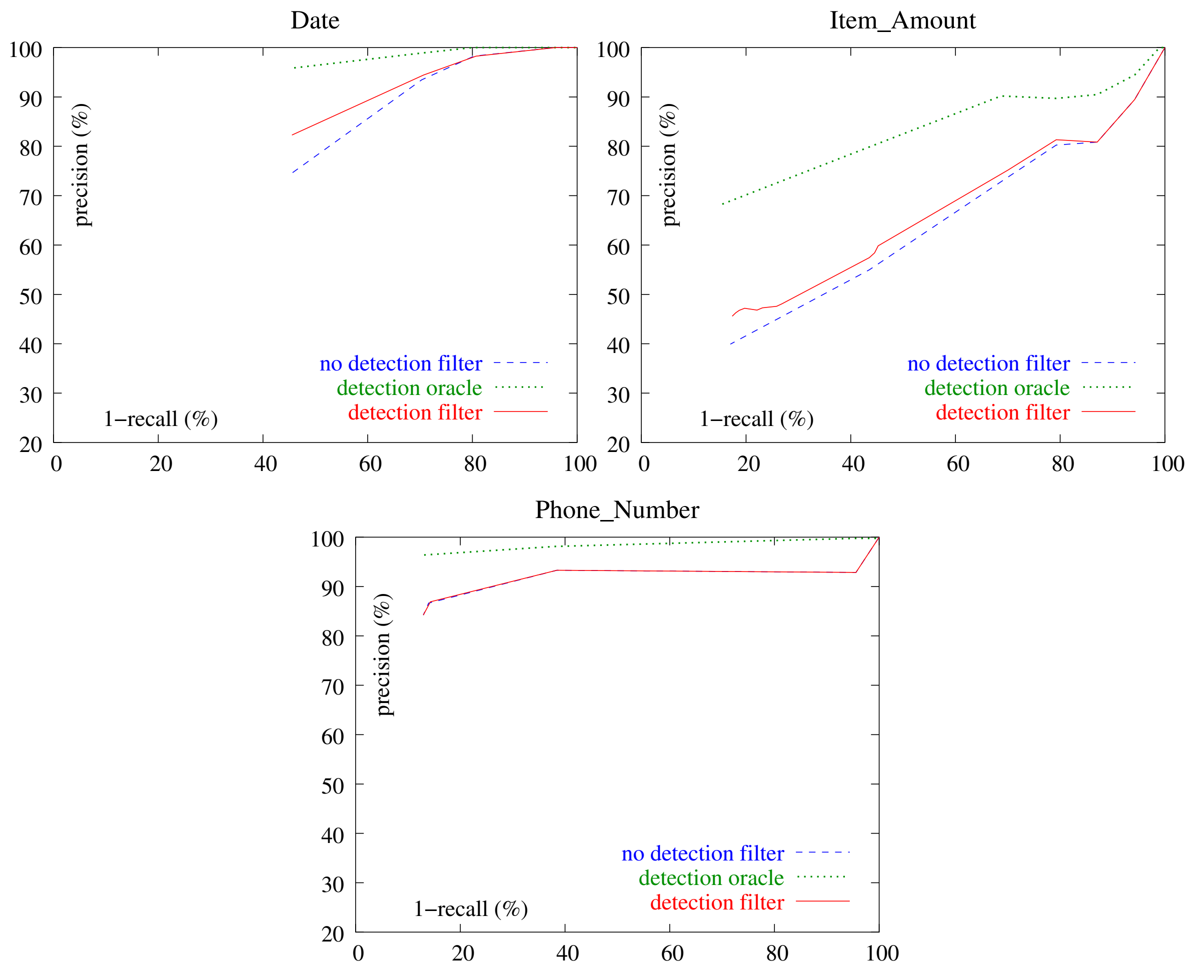


Figure 6.12: Effect of pre-filtering by detection on localization of named entities.

models. So, in case of named entity DATE and the background language model the value extraction F-measure rose from 66% to 69% solely due to the pre-filtering by detection.

As far as the localization complexity is concerned, a drastic reduction of the number of utterances that were searched for at least one named entity type was observed. In fact, localization was started only on each tenth utterance, and in many cases where it was indeed started, only named entities of one type were looked for. While this didn't result in the overall speed-up for this particular experiment (exact maximum-likelihood parsing is rather fast), we will see in the next section how important this adjustment turned out to be when approximate matching option was turned on.

6.5.4 Approximate Matching for Named Entity Localization

As long as we allow only for exact matches of named entity fragments in the recognized sentences we are doomed to operate in the low recall area. This is because of the ASR-errors which, even by word accuracy 78%, are far from being neglectable. It is also clear that even 100%-correct detection oracle from previous section can't compensate for misses due to the recognition errors. In fact, no matter how certain we are about NE-presence, there is nothing we can do about it if there are simply no allowed matches. The problem aggravates further with an increasing length of the named entities. For instance, for a string of length n its recognition recall behaves as R^n with the average word-based recall R. Correctly estimated language model is certainly one way to extenuate the problem; instantiation on lattices is another one. In this section we pursue yet another alternative: parsing with approximate matching. The strategy makes it possible to overlook an occasional misrecognition error by instantiation of name entities in the utterance, as long as the whole instance is reasonably probable at the given location, and the error to skip over is known as being rather typical for the employed ASR. In a way, detection by pre-filtering and approximate matching stand in a complementary relationship to each other, since from the former we expect to remove false positives, and the latter should take care of the misses.

In essence, the estimation of word misrecognition probabilities adheres to formulae (4.10) and (4.11) from Section 4.3.4. However, we felt the need to smooth the estimated probabilities by ignoring all word mappings from the confusion matrix that occurred only once, and also by sustaining identity mappings for all words even if they didn't occur in the confusion matrix at all. We also discovered that a successful combination of likelihood $P(s^t|f^t)$ with other probabilistic components of formula (4.6) required some rescaling of the estimated word mapping probabilities. Namely, we obtained optimal results by multiplying all non-identity mapping probabilities by factor 2, and making identities cost-free.

Next, we are going to demonstrate the effects of the approximate matching on the named entity localization and value extraction for word- and phone-level ASR-output.

Word-based Algorithms

For approximate matching of word-based named entity fragments we used the context-independent distortion transducer computed from paired transcriptions and ASR-output of corpus PRES as described in Section 4.3.4. We also decided to extend the generic named entity fragments by a small amount of context (see Appendix C).

The resulting localization and value extraction rates confirmed the presumed advantage of

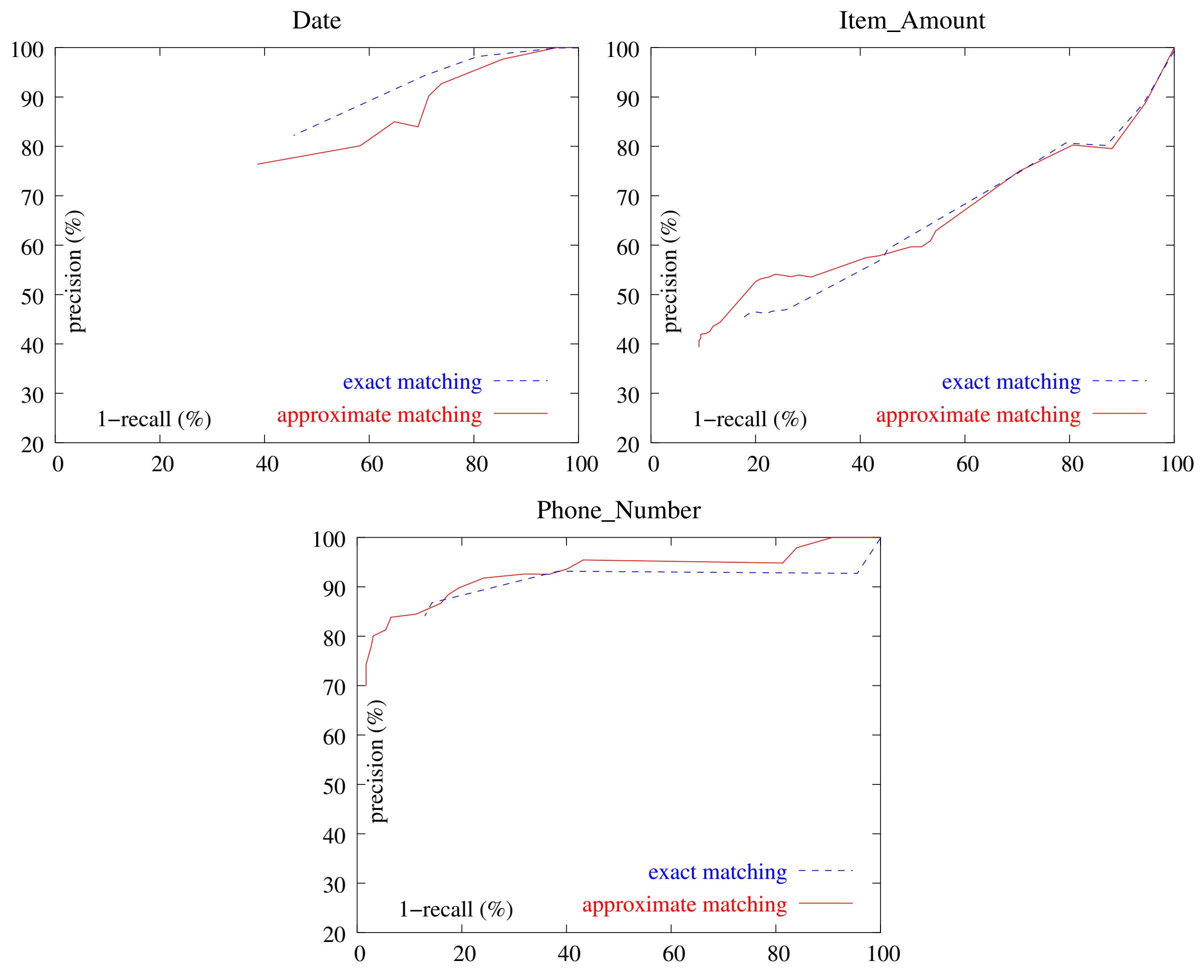


Figure 6.13: Influence of approximate matching during parsing on named entity localization results.

approximate matching in the information retrieval tasks: from the comparative analysis of the ROC-curves in Figure 6.13 we see that the approximate matching enabled their extension in the area of high recall, with no (or minimum) loss in precision. The corresponding operating point F-measures for value extraction gathered in Table 6.12 show that unlike exact matching on lattices, approximate matching on best paths does in fact manifestly improve value extraction results as well.

Effect of Context-dependency in Phone-based Distortion FSM

The approximate matching mechanism for named entity localization and value extraction from a phone stream remains exactly the same as in the word case. What can be changed, how-

	DATE	ITEM_AMOUNT	PHONE_NUMBER	OVERALL
exact; best paths	0.62	0.53	0.76	0.63
exact; lattices	0.61	0.52	0.79	0.63
approximate; best paths	**0.64**	**0.58**	**0.80**	**0.67**

Table 6.12: Influence of approximate matching during parsing on named entity value extraction (F-measure); additionally results of exact matching on lattices are presented.

ever, is the procedure for the estimation of the distortion transducer. On the one hand, we can get more reliable estimators for probabilities of individual phone-misrecognitions, since (with the smaller lexicon) their number becomes very limited, on the other hand the practical use of context-independent phone misrecognition statistics seems to become questionable. In the part of Section 4.3.4 dedicated to approximate matching at the phone level, we elaborated mathematical foundations of the context-*dependent* phone misrecognition statistics. The need for smoothing was also stressed there.

Now, we present the results of three experiments illustrating the correctness of these ideas. We avail ourselves of the named entity detection pre-filtering, which this time was conducted at the phone-level (i.e. we took over the results from Section 6.4.4). We consciously leave out any mention of the baseline exact-match experiments at the phone level, for these experiments proved to be hopelessly futile, non resulting in any practical localization or value extraction rates at all (except maybe for ITEM_AMOUNT where value extraction F-measure in the operating point was about 0.34).

Three approaches to modeling of phone misrecognitions were tested and compared against each other as well as to the results achieved with exact matching on phone lattices:

1. in the first approach, context-independent phone confusion statistics were combined in a simple distortion transducer in a manner similar to the one used for words; the difference to the word case, however, was that here we estimated the phone mapping probabilities using Expectation Maximization algorithm, and not from the Viterbi alignment (cf. Section 4.3.4);

2. the second approach made use of the direct left and right contexts of the phones as stipulated in formula (4.13); although, no interpolation between contexts with different degrees of specificity, as suggested in formula (4.16), was done;

3. finally, we considered the interpolated version with empirically chosen weights from (4.16): $\xi_f = 0.5,\ \xi_l = 0.2,\ \xi_r = 0.2,\ \xi_n = 0.09,\ \xi_z = 0.01.$

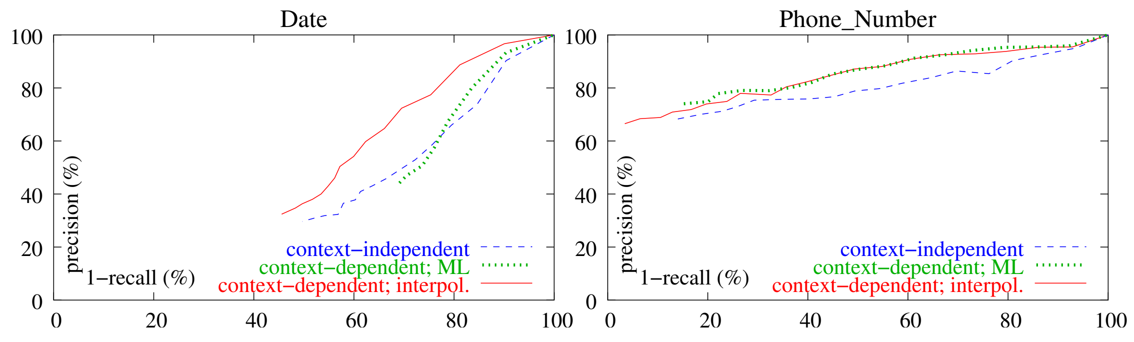


Figure 6.14: Approximate matching for NE-localization at the phone level; effect of context and interpolation.

	DATE	ITEM_AMOUNT	PHONE_NUMBER	OVERALL
lattices exact match	0.26	0.38	0.44	0.37
best path, context-indep. approximate match	0.34	0.52	0.47	0.45
best path, context-dep. approximate match	0.33	0.52	0.55	0.48
best path, smoothed context-dep. approx.	**0.42**	**0.55**	**0.59**	**0.53**

Table 6.13: Approximate matching for NE-value extraction at the phone level, effect of context and interpolation; additionally results of exact matching on lattices are presented.

In Figure 6.14 the localization ROC-curves for named entities DATE and PHONE_NUMBER and three approximate matching techniques are shown. It can be seen that while the overall localization rates remain clearly under the ones achieved using adapted word language model, the relative advantage of the interpolation technique over the context-independent and the context-dependent but not smoothed versions is manifest. We compare the value extraction F-measures for the operating points of the curves as well, whereby also exact match on phone lattices is considered. Table 6.13 shows that, as expected, also named entity value extraction from phone-level ASR-output produced the best results when using the interpolated version of context-dependent approximate matching. This version outperformed by far the exact match on phone lattices, as well as context-independent and context-dependent but not smoothed approximate matching at the parsing stage.

	words			phones		
criterion	baseline	SVM	boosting	baseline	SVM	boosting
RT-factor, detection	—	0.06	≈ 0	—	< 0.01	≈ 0
RT-factor, localization	0.02	< 0.01	< 0.01	4.10	1.20	1.20
RT-factor, overall	0.02	0.06	$<$ **0.01**	4.10	**1.20**	1.20
F-measure, localization	0.72	**0.74**	0.72	0.59	**0.64**	0.61
F-measure, val.extraction	0.65	**0.67**	0.65	0.50	**0.53**	0.51

Table 6.14: Pre-filtering by detection and approximate matching; baselines do not include pre-filtering by detection, otherwise accomplished by LLAMA or BOOSTEXTER.

Time Savings Using Pre-filtering by Detection

While introduction of the approximate matching improved the localization and value extraction results, it also significantly slowed down the entire localization process, emphasizing once again the utility of the upstream pre-filtering by detection, especially for the phone-based experiments.

In Table 6.14 we assembled performance statistics for several word- and phone-based parsing runs employing approximate matching[10]. In particular, we looked at the real-time factor of the localization (and, where applicable, detection) procedures, as well as the resulting overall F-measure of the localization and value extraction. For the phone-level experiments we see that pre-filtering by detection not only improves the localization and value extraction rates, but also speeds up the costly localization process, the effect we anticipated in Section 6.5.3. In the word-based case, this acceleration is not as critical; however it still can be achieved by replacing SVMs by the much faster albeit less accurate BOOSTEXTER, thus waiving the improvement in the localization and value extraction rates for the benefit of a more favorable time behavior.

6.5.5 Parsing with Higher Order Language Models

Finally in this section, we report on our experiments regarding the choice of the parsing language model for named entity localization by means of ML-parsing. We are interested in the influence of the language model order on the localization and value extraction results. The previous experiments were all conducted using a simple unigram parsing language model, estimated from the training corpus HMIHY-3N-train. In particular, since no manual transcriptions of this corpus could be used for training we availed ourselves of a rather crude approximation: the probabilities of words were estimated via simple counting in the ASR-output, and the priors of all named en-

[10]In the case of phones, the described above context-dependent distortion FSM with interpolations was considered.

tity types were approximated as the relative frequencies of the utterances containing them. This approximation was legal only because of the low priors of such utterances and also because most of them contained at most one NE-instance of the type (cf. Table 6.3). One possible extension of this strategy was suggested in Section 4.3.3. There, an iterative mechanism was designed to alternately parse the training corpus in terms of words and named entities (in the same way that we used to localize NE-instances in the test data) and re-estimate a new parsing language model with words *and* named entities in it.

The results were rather disillusioning. Parsing with the trained bigram parsing language model in place of the simple unigrams did make the average utterance parsing costs per word measured on the test corpus drop by about 1.0 (which corresponds to the growth of the normalized likelihood by factor 2). At the same time, it had almost no effect on the named entity localization quality. One plausible explanation for this is the relative low occurrence frequencies of the named entities in the corpus and as a consequence the need for more data in order to obtain reliable statistics for bigrams that include them. Remember also that many of the named entities typically occur at the end of syntactic phrases (e.g. *"My number is 123 4567890."*), where the perplexity is higher than average. Furthermore, with some minimum context, as used for localization with approximate matching, they often align with the beginning of syntactic phrases as well (*"My number is 123 4567890."*). All these factors seem to have made the transfer from unigram- to bigram- (and trigram-) language models for parsing not advantageous.

6.6 Extraction of Acoustic Morphemes

The next part of this chapter is dedicated to our experiments on inducing word-like units from a continuous phone stream. Following the recommendations from Chapter 5, two criteria are used for the constructive definition of words: their entropy reducing property for within-channel modeling and their salience for the task (cross-channel).

Correspondingly, our experiments are split in two parts. In the first part, the goal is to extract phone strings from a continuous phone stream such that the test corpus, when expressed (parsed) in terms of these strings, would have the probability as high as possible. The second part pursues a similar goal but applied to the overall salience of the test utterances. In both cases, the extraction is done using the HMIHY-3CI-train corpus and its results are tested on HMIHY-3CI-test.

6.6.1 Utility for Within-channel Modeling

On page 97 of Section 5.1.3 we suggested the following configuration for the iterations of the multipass algorithm (Figure 5.2) to produce entropy-reducing phone strings:

- during the phrase selection stage, thresholding of the normalized mutual information (5.3) takes place, while another threshold is imposed on the number of phrase occurrences in the training corpus.

- for the parsing stage, phrase costs are based on their mutual information (5.2).

To evaluate the phone strings obtained in this way, we parse the test corpus on each iteration in a similar manner as by the training corpus. From the parsed training corpus we estimate a simple unigram language model and evaluate this model on the parsed test corpus. The figure of merit is the average length-normalized utterance probability. Let the utterance S_i that in the original phone-level representation counted $|S_i|$ phones, consist of $|S_i^*|$ phone strings after the parsing took place: $S_i^* = s_{i1} \ldots s_{i|S_i^*|}$. Each of these strings has a probability p_{ij} estimated from the training corpus, so that the unigram-based probability estimator for S_i is

$$p_i = \prod_{t=1}^{|S_i^*|} p_{it}. \tag{6.11}$$

To exclude the influence of the utterance length on the probability, we normalize it. Then, using costs $\omega_{it} = -\log p_{it}$ and $\omega_i = -\log p_i$, we write:

$$\omega_i = \frac{1}{|S_i|} \sum_{t=1}^{|S_i^*|} \omega_{it}. \tag{6.12}$$

The final figure of merit for the entire corpus of I utterances is the inverse of the following average costs:

$$\Omega_{\text{av}} = \frac{1}{I} \sum_{i=1}^{I} \omega_i. \tag{6.13}$$

Our first question is how the new lexicon of the extracted phone strings influences the value of Ω_{av} compared to a continuous phone stream and also to a corresponding word-level representation[11].

[11] Note that for the word-level representation normalization still has to be done with respect to the number of phones in the utterance.

	phones	phone phrases	words
Ω_{av}	5.0	2.1	1.5
Ω_{dev}	0.2	0.8	0.2

Table 6.15: Mean and standard deviation for the normalized cost of the test corpus, when expressed as phones, phone strings and words.

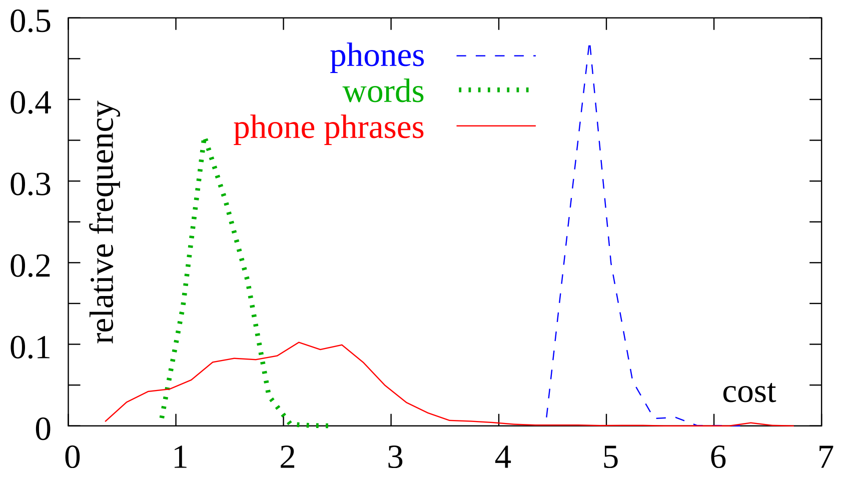


Figure 6.15: Distribution of utterance costs for test corpus expressed as phones, entropy-reducing phone strings and words.

In Table 6.15 we compare the three average costs (and also the standard deviations of the costs ω_i) in the test set, whereby the phone strings were obtained after five iterations with the threshold on the normalized mutual information $\theta_I = 1.0$. Besides, on each iteration the phrase had to occur at least five times in the latest corpus representation in order to survive pruning. The same results we also plotted as a comparative study of normalized cost distributions in Figure 6.15.

We see that test corpus expressed in terms of the phone strings extracted with the multipass algorithm has a much higher probability than when simple phone representation is used. It comes close to the word-based probability, and for some utterances even surpasses it. The reason for such a behavior is that by growing phone phrases on each iteration we ultimately acquire many phone strings which — often corresponding to very common word strings like *"I'd like to"* — represent an even better alternative for language modeling units than words. In fact, out of curiosity we applied the same multipass algorithm to the word representations of the HMIHY-3CI corpus. After some iterations, the average normalized cost Ω_{av} dropped further and reached

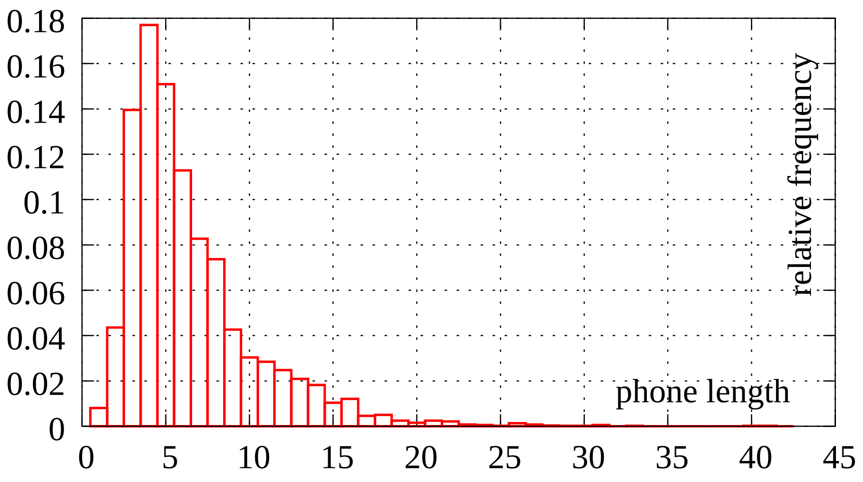


Figure 6.16: Distribution of selected phone strings by phone length.

1.0. On the other hand, because of the limited size of the training corpus, there is a considerable amount of utterances in the test corpus that are tough to parse with the phone strings extracted from the training corpus. As a result of these two observations, the standard deviation of the cost distribution grows.

The length of the phone strings generated by the multipass algorithm varied between 1 and 41. The longest string extracted corresponded to an entire sentence: *"I'd like to speak to a customer service representative"*, and came into existence solely due to the nature of the training corpus. The distribution of the selected phone strings by the number of phones they contain is shown in Figure 6.16. It is remarkable that the center of gravity of this distribution (and even more so its mode) lies very close to the average length of words in English which is 4.5.

There are several parameters in the algorithm that can be varied. Some of them, while reducing the average cost of the corpus, also lead to a dramatic increase in the dictionary size. One example of such parameters is the threshold $\theta_{\#}$ on the number of phrase occurrences in the parsed training corpus which must be exceeded in order for this phrase to survive the iteration. In the experiments above this threshold was set to 5. The plot in Figure 6.17 shows that reducing this threshold (i.e. allowing to select phrases with lower reliability of high estimates of their within-language modeling utility), tends to reduce the costs of the test corpus representation at the expense of quickly expanding dictionary.

We prefer not to make a judgment as to which point on the curve to choose as optimal, even though there are suggestions relating to it in the literature. In [dM96], for instance, the *Minimum Description Length (MDL-)* strategy [Ris89] is applied to the problem of lexicon selection from

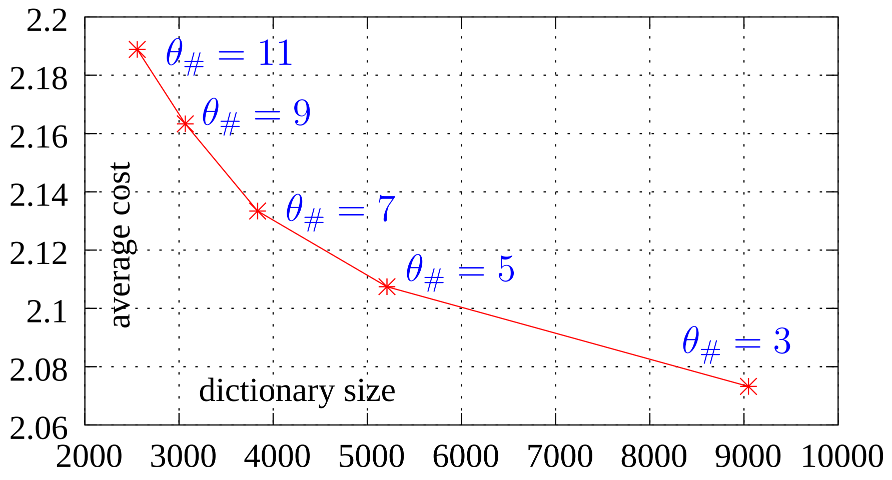


Figure 6.17: Dependency of the test corpus cost on the dictionary size; changing threshold $\theta_{\#}$ on the number of phrase occurrences.

continuous speech. According to the MDL-principle, the choice between competing models is guided by the combined complexity of the corpus representation with this model and the model itself. Since the essential model in our case is the lexicon (we use unigram language model to evaluate lexicon's quality, but could certainly do the evaluation with language models of a higher order as well) and complexity of the lexicon is stipulated by its size, the final figure of merit measured on the test corpus will depend on the ratio between the size of the lexicon and the size of this corpus. We didn't like the idea of the size of test data significantly influencing the selection of algorithm parameters, and therefore decided to delegate the final choice to the potential end-user.

6.6.2 Salience Property

The second criterion defining a word is its meaningfulness that, according to Definition 5.1, is evidenced in strong semantic associations of the word (e.g. with calltypes). Word meaning always roots in the semantics of the application. The richness of these semantics, i.e. the size and intrinsic complexity of the semantic hierarchy mentioned on page 15 determines to what extent the semantic associations of a word learnable from a pair of acoustic and extra-linguistic channels will resemble its common meaning in the general language, the one that we can find in an encyclopedia. With only calltype classification at hand we are bound to curb our desire for a detailed categorization of words with all the subtle nuances familiar to us from the everyday life. What we are left with instead, is the exploration of word associations with the available calltypes. The kind of associations we used to call *salience*.

The bottom part of the list of the entropy-reducing phone strings generated by the first stage of the multipass algorithm (after five iterations) and sorted in the order of decreasing salience, contained many phone strings like, for instance, *"hh_ae_z" (has)* or *"ay_d_l_ay_k_t" (I'd like t(o))*[12]. These strings are in fact very common in the call center scenario we are working in, but convey virtually no information as to a particular issue the customer wants to discuss. Yet, if we take the average cost, based on the posterior conditional probability of the correct label $c^{m(i)}$ (a.k.a. "semantic probability") for all phone strings s_{ij} in the parsed utterance S_i^*:

$$\bar{\omega}_i := -\frac{1}{|S_i^*|}\sum_{j=1}^{|S_i^*|}\log P(c^{m(i)}|s_{ij}), \tag{6.14}$$

and average this *semantic cost* also over all utterances of the corpus:

$$\bar{\Omega}_{\text{av}} := \frac{1}{I}\sum_{i=1}^{I}\bar{\omega}_i, \tag{6.15}$$

we will see that extracting entropy-reducing phone strings makes this cost (which we obviously want to be as low as possible) decrease from ca. 4.3, as it was the case by a continuous phone stream, to 3.7! We think that this is a very interesting result for it shows that lexicon (and for that part also syntax) remotely stipulate semantics of the task. The famous example is the "Jabberwocky" poem by Lewis Carroll. Despite the fact that all supposedly meaningful words are utter gibberish, some shimmer of the meaning does reach the reader, one of the reasons being the "natural" structure of the verses. Thus, the observation we have made, presents an indirect confirmation of our original premise that words should be characterized as units of syntax *and* semantics at the same time.

Let us concentrate more on the semantic associations of words now. To select only the phone strings that can shed light on the topic, we can not base the parsing cost on PMAX, for the convergence of the algorithm wouldn't be guaranteed anymore. Instead, we introduce the threshold θ_{PMAX} on PMAX on the last iteration of the multipass algorithm. Then, to estimate the predictive power of the selected strings, we measure the semantic cost of each utterance (6.14) averaged only over the salient phone strings in it. Since not every utterance will now contain such strings, we also modify (6.15), averaging only over the utterances that do. The dependency of this redefined average semantic cost and also of the percentage of the utterances taken into account while

[12] Here and below we use the ARPABET phonetic alphabet [ARP] to represent phone-level transcriptions.

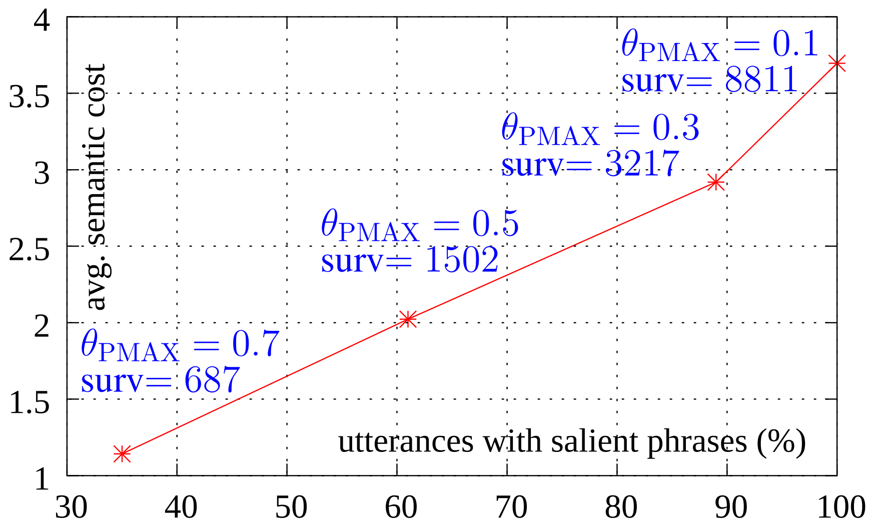


Figure 6.18: Dependency of the average test corpus semantic cost and the percentage of utterances with salient strings; additionally the corresponding salience threshold θ_{PMAX} and the number of phone strings $surv$ that survived this thresholding is shown.

computing it on the imposed salience threshold θ_{PMAX} is shown in Figure 6.18. This situation is similar to the precision-recall trade-off: the stronger we want the semantic associations of the selected salient phone strings to be (higher precision), the lower is the number of cases where these strings are present (lower recall).

Sometimes, it is the complexity of the algorithm (e.g. calltype classification using the selected salient phone strings) that is decisive for the parameter choice, for it requires the number of selected salient phone strings to remain as low as possible. This is where the significance test described in Section 5.2 kicks in very effectively. For instance, we applied the multinomial significance test (5.9) to all selected phrases with PMAX exceeding threshold $\theta_{\text{PMAX}} = 0.1$. Here, whenever the exact multinomial significance test wasn't computationally feasible, the Monte-Carlo approximation from page 101 was used. As a result, the average semantic cost 3.8 was achieved, just slightly higher than the reference value 3.7, but the number of selected salient phone strings was cut down by more than one half, now counting only about forty three hundred.

6.6.3 Putting It All Together: Acoustic Morphemes

Now that we have selected linear phone sequences that possess the fundamental qualities idiosyncratic to words, the next step is to identify the irrelevant component in them (noise) and to group together those phone strings that can be considered indistinguishable from the word definition

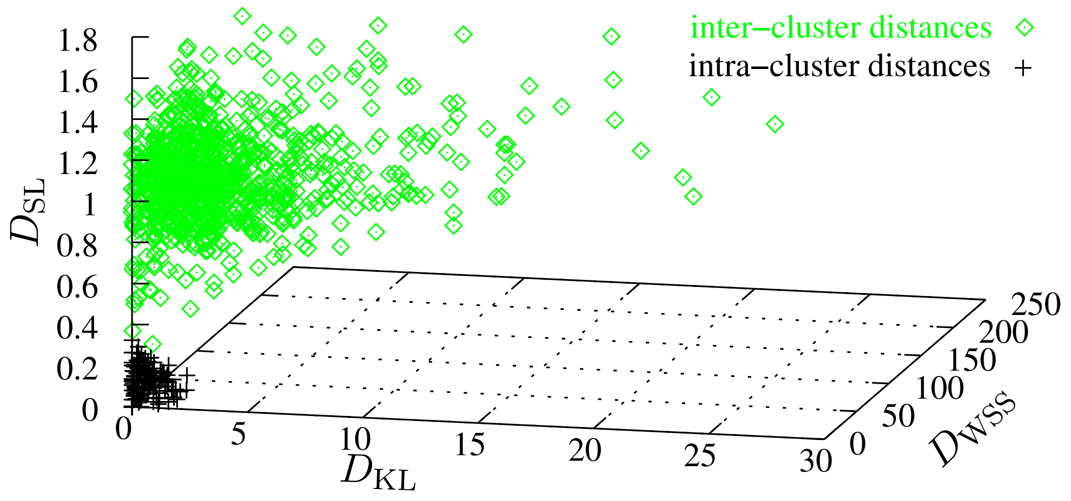


Figure 6.19: Intra- and inter-cluster distances for created acoustic morphemes; for the sake of clearness, only each 50th inter-cluster distance is plotted.

perspective. In other words, we group the selected salient entropy-reducing phone strings into *clusters* according to the acoustic and semantic similarity measures, described in Section 5.3.2.

We applied the multi-distance clustering to 687 phone strings selected by the multipass algorithm. After carrying out the cluster initialization method using binarized vectors of significant semantic associations (cf. Section 5.3.2, page 106), and requiring $D_{\text{BIN}} = 0$ for all phone strings forming one pre-cluster, we obtained 37 such pre-clusters. Following that, we split clusters based on the acoustic distances between each string s^{ik} and the centroid of the cluster O^i it belonged to. In particular, the cluster was split in two if:

$$\exists s^{ik} \in O^i : \quad D_{\text{SL}}^{\text{norm}}(s^{ik}, O^i) > 0.5. \tag{6.16}$$

After 18 iterations, we obtained 323 clusters and started merging them based on all four distances D_{BIN}, D_{SL}, D_{KL} and D_{WSS} introduced in Section 5.3.2 which resulted in 301 clusters. Finally an LBG-like algorithm mentioned on page 109 was used to rearrange phone strings in clusters and produce their final versions: *acoustic morphemes*. The 3D-plot in Figure 6.19 shows the intra- and inter-cluster distances for and between the created acoustic morphemes. It illustrates the compactness of the created clusters in terms of the acoustic and semantic distance metrics, which is especially striking for the simple Levenshtein acoustic distance D_{SL}. The plot also justifies our decision to use both semantic distances D_{WSS} and D_{KL} to form clusters, since there is no convincing correlation between the two (cosine measure only about 0.7). ,

PAY_BILL	DISCONNECT	UNRECOGNIZED_NUMBER
ey k ax p ey m ax n t	ae n s ax l m ay s er v ih s >	d ih n ax f ay z s ah m
m ey k ax p ey m ih t >	< k ae n s ax l s er v ih s >	
m ey k ax p ey m ax n >	k ae n s ax l m ay ax k aw n t >	
m ey k ax p ey m ax n t	k ae n s ax l m ay s er v ih s >	
m ey k ax p ey m ax n t >	t uw k ae n s ax l m ay s er v ih s >	
m ey k ax p ey m ax n		

Table 6.16: Examples of constructed acoustic morphemes.

The number of salient phone strings in the created clusters varied from one to fifteen. More than half of them (174) consisted of just one string, another 20% had only two strings in them. Similarly uneven turned out the degree of calltype coverage: in Figure 6.20 we graphically depict which acoustic morphemes (rows) are positively correlated with which calltypes (columns). Here, the calltypes are ordered by their prior frequencies in the corpus (cf. Figure 6.1) and the black dash at an intersection of a calltype and an acoustic morpheme signifies a positive correlation of the two. The unsurprising outcome is that more frequent calltypes are covered by more acoustic morphemes[13].

Next, we list some representative examples for the acoustic morphemes created in our algorithm[14]. All examples from Table 6.16 are unambiguously associated with only one calltype (as are all but five of the 301 created acoustic morphemes). The first column presents one of the acoustic morphemes accounting for the calltype PAY_BILL (customer wants to make a payment over the phone). We see that all of the six strings that constitute this acoustic morpheme relate to the word sequence *"make a payment"*. This example confirms that the ASR phone misrecognition errors are the main source of noise in our task, and clustering is intended to compensate for it. In the second column there is an acoustic morpheme for calltype DISCONNECT (customer wants to cancel his contract with the company). Here, clustering threw together phone strings originating from various wordings (*"[to] cancel [my] service"*, and *"cancel my account"*). While these word sequences are certainly different from the linguistic point of view, they are virtually indistinguishable from the application perspective, and therefore become part of one acoustic morpheme despite some acoustic differences.

Finally, one acoustic morpheme created for calltype UNRECOGNIZED_NUMBER is shown in the third column of the table. It consists of just one phone string and is one of those rare cases

[13]The measured cosine between the prior distribution of calltypes and their coverage by acoustic morphemes was 0.93.

[14]In these examples we use symbols "<" and ">" to denote begin and end of the utterance respectively.

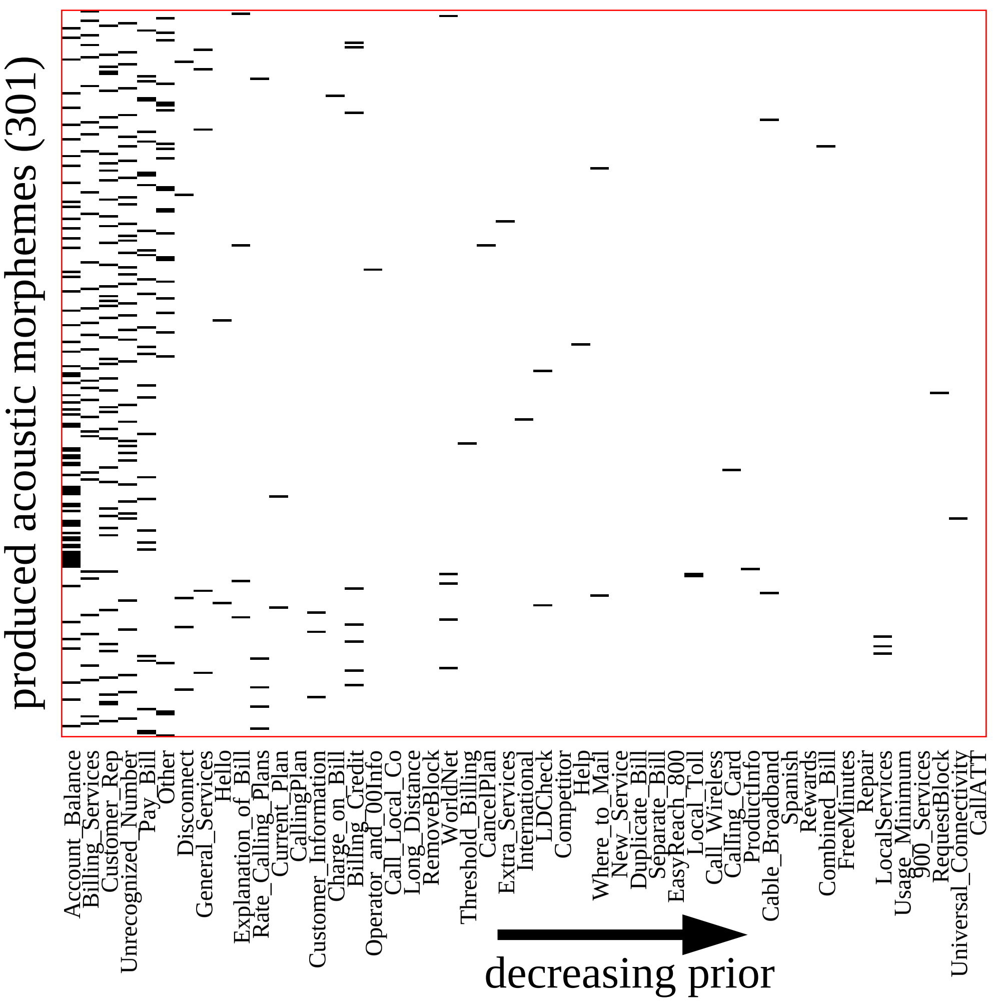


Figure 6.20: Acoustic morphemes and calltypes they are associated with; black dashes mean positive significant correlation.

where it is extremely difficult to identify the word or word sequence behind it, just by looking at it. Yet, this phone string (which stems from a word pair *"identify some"*) fulfilled the word-defining criteria in our experimental setup and was therefore selected as an independent acoustic morpheme.

On the whole, we observed that less than 4% of all created acoustic morphemes contained phone sequences that couldn't be easily recognized by a human observer as an English word or a sequence thereof. For more examples of acoustic morphemes created in our experiments see Appendix D.

Our last result in this chapter concerns the practical utility of acoustic morphemes for calltype

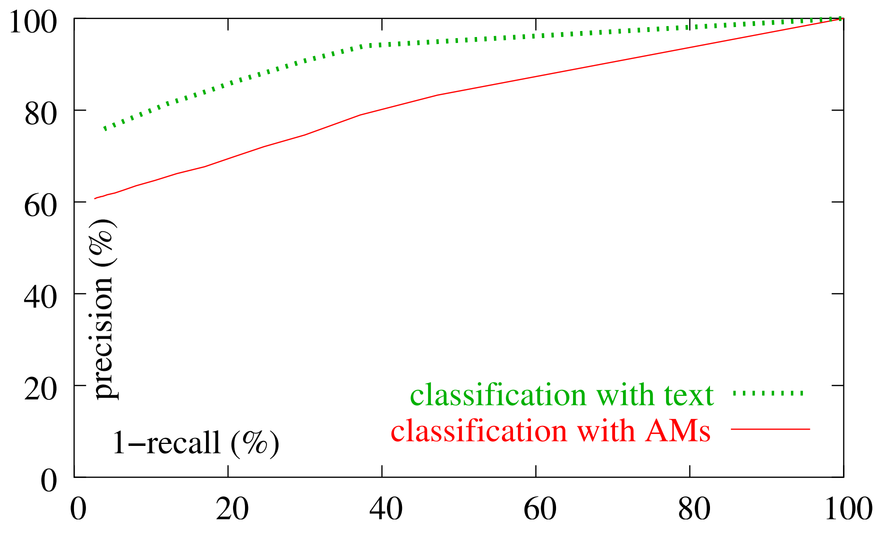


Figure 6.21: Using acoustic morphemes for calltype classification; comparison with the calltype classification results from Section 6.3.4.

classification. We took 301 acoustic morphemes generated as shown above and employed them as features for calltype classification in the following manner: using an approximate matching mechanism similar to the one described in Section 4.3.4 we parsed the training and test corpora (both obtained as an output of the phone recognizer), keeping record of all found instances of acoustic morphemes in both corpora [Lev02]. Then we trained a support vector machine classifier with classification features reflecting presence (and parsing score) of the acoustic morphemes in the utterance. In a similar manner the results of the acoustic morpheme instantiation on the test corpus were used to extract feature vectors for the test stage. The resulting ROC-curve of this classification experiment on the HMIHY-3CI corpus, along with the ROC-curve of the corresponding baseline experiment with the classification features extracted from text (cf. Section 6.3.4), are shown in Figure 6.21.

As it turns out, it is still possible to achieve reasonable calltype classification accuracy using only acoustic-morpheme-based features; however, this accuracy remains clearly inferior to the one achieved when the entire ASR-output is taken into account (F-measures in the operating points: 60% vs. 73%). We deem the insufficient number of the selected acoustic morphemes related to the strict pruning criteria to be the reason for these relatively weak classification results. In fact, by relaxing the thresholds on salience, normalized mutual information and required number of phrases occurrences in the training corpus, we were able to obtain much better classification results (F-measure ca. 68%), but many of the selected acoustic morphemes lost their apparent resemblance to words (or word sequences).

6.7 The Experiments of this Chapter Revisited

In this last section we review all the conducted experiments once again, presenting their results in a succinct form. Overall, series of experiments pertaining to several topics of the spoken language understanding in a call center scenario have been carried out under the difficult condition of **missing manual transcriptions for the domain of interest**:

- **calltype classification**
- **named entity processing**
  - **named entity detection**
  - **named entity localization**
  - **named entity value extraction**
- **extraction of acoustic morphemes**.

Our first goal was to extricate the understanding mechanisms from the traditional heavy dependency on the manually created in-domain transcriptions. Primarily, this dependency is voiced in large transcribed corpora that are normally used to train the language model of an ASR-system. Instead we decided to use a simple method of an unsupervised language model adaptation, iteratively fitting an off-the-shelf background language model in the domain of interest (in our case it was *"How May I Help You?"*-task) using only audio data from this domain. In a similar way, we also tuned up the phone-level language model starting with a simple phone-loop. Table 6.17 summarizes the positive effects of this adaptation on the word and phone accuracy, calltype classification and the three tasks of the named entity processing. The HMIHY-3CI-test corpus was used to assess the recognition accuracy and calltype classification, and HMIHY-3N-test to evaluate our algorithms for the named entity processing tasks. The unsupervised word-level language model adaptation resulted in compensating of 50 to 75 percent of the losses that one gets when using the background language model instead of the model trained on the in-domain transcriptions, while some of the phone-level algorithms required several extentions summarized below to render them practical.

As far as the calltype classification is concerned, we used two corpora to assess the performance of our algorithms: initial turn utterances (HMIHY-3CI) as in Table 6.17 but also all utterances (HMIHY-3CA). Also, two alternative classification algorithms, SVM and boosting, were tested. In Table 6.18 the classification results are put together. Apart from the conclusive

language model criterion	background	adapted (words)	adapted (phones)	transcr.-trained	relevant section(s)
word accuracy	63%	70%	—	75%	6.2
phone accuracy	77%	83%	70%	85%	6.2
calltype classif.	70%	73%	74%	76%	6.3.2
NE detection	0.81	0.83	0.78	0.84	6.4.2,6.4.4
NE localization	0.60	0.69	—	0.73	6.5.2
NE val. extract.	0.51	0.62	—	0.68	6.5.2

Table 6.17: Summarized effects of the unsupervised language model adaptation on the quality of the automated spoken language understanding; correct decision rate is the success criterion for calltype classification, and F-measure for the named entity processing tasks.

experiment	correct decision rate	relevant section
HMIHY-3CI, boosting	68%	6.3.2
HMIHY-3CI, SVM	73%	6.3.2
HMIHY-3CA, SVM	82%	6.3.2
HMIHY-3CA, SVM + nonterminals + prompt	83%	6.3.3
HMIHY-3CI, SVM (phones)	74%	6.3.4

Table 6.18: Calltype classification experiments summarized.

superiority of the SVM, one can also see an additional improvement as a result of the introduction of special nonterminals (such as “digit” and “month”) and using system prompts for classification (Section 6.3.3); besides one can see that calltype classification can be carried out at the phone-level without loss of performance (shown on the HMIHY-3CI corpus).

The results of the experiments dedicated to named entity detection, localization and value extraction are collected in Tables 6.19 (detection) and 6.20 (localization and value extraction). Unlike Table 6.17, at this point we provide F-measures for each named entity individually instead of just specifying their overall average.

Table 6.19 shows once again that support vector machines outperform boosting also for the NE-detection task, no matter if working on phone- or word-level ASR-output. At the same time, we see that word-based detection apparently is capable of a better performance. In this task also, an additional improvement can be expected from a filtering of the recognized word sequences with nonterminals and from using system prompt as yet another classification feature.

experiment	named entity			relevant section
	DATE	ITEM_AMOUNT	PHONE_NUMBER	
boosting	0.66	0.64	0.89	6.4.2
SVM	0.77	0.76	0.93	6.4.2
SVM + nonterminals + prompt	0.77	0.79	0.93	6.4.3
phones, boosting	0.55	0.64	0.83	—
phones, SVM	0.65	0.74	0.91	6.4.4

Table 6.19: Named entity detection experiments summarized (F-measure).

experiment	named entity			relevant section
	DATE	ITEM_AMOUNT	PHONE_NUMBER	
ML-parsing	0.63 / 0.59	0.56 / 0.50	0.86 / 0.77	6.5.2
+ oracle detection	0.70 / 0.65	0.76 / 0.67	0.92 / 0.81	6.5.3
+ SVM detection	0.67 / 0.62	0.60 / 0.53	0.86 / 0.76	6.5.3
+ SVM detection + lattices	0.65 / 0.61	0.58 / 0.52	0.88 / 0.79	6.5.4
+ approximate matching	0.66 / 0.62	0.63 / 0.57	0.88 / 0.79	—
+ SVM detection + approximate matching	0.69 / 0.64	0.64 / 0.58	0.88 / 0.80	6.5.4
phones + SVM detection	0.08 / 0.08	0.49 / 0.34	0.03 / 0.02	—
phones + SVM detection + lattices	0.31 / 0.26	0.50 / 0.38	0.67 / 0.44	6.5.4
phones + SVM detection + context-independent approximate matching	0.39 / 0.34	0.60 / 0.52	0.76 / 0.47	6.5.4
phones + SVM detection + approximate matching	0.36 / 0.33	0.62 / 0.52	0.79 / 0.55	6.5.4
phones + SVM detection + approximate matching (interpolated)	0.46 / 0.42	0.61 / 0.55	0.79 / 0.59	6.5.4

Table 6.20: Named entity localization / value extraction experiments summarized (F-measure).

Looking at the named entity localization and value extraction results from Table 6.20, we conclude that pre-filtering by detection is crucial for improving the localization and value extraction rates, but it also can lead to a significantly lower real-time factor in the batch mode (see Table 6.12). Also important is the mechanism of approximate matching, which produced significantly better results than, for instance, exact matching on the ASR-lattices. While phone-based strategy is clearly inferior to the word-based one for this task, some substantial improvements can yet be achieved by using approximate matching, especially when context-dependent approximate matching along with the interpolation smoothing technique is employed.

Finally, after witnessing the superiority of many word-level algorithms to their phone-based counterparts, we considered an academic problem of automatic extraction of word-like units from a continuous phone stream. Directing the search by the entropy-reducing property of words and their expected salience for the calltype classification task, we were able to extract phone strings that reduced the unigram-based parsing costs of the test corpus expressed in terms of these sequences (see Table 6.15), and also the semantic cost of the utterances obtained as an average over all salient phone strings found in them (Section 6.6.2). In the same section, we demonstrated that the number of the selected salient phone strings can be controlled by the semantic significance test that would prune away all the phrases with salience estimates that are not reliable enough. In Section 6.6.3 we then showed how the selected salient, entropy-reducing strings can be organized in a set of well separated clusters (acoustic morphemes) that are usually easily identifiable with the words that gave rise to them and also reflect the semantics of our calltype classification task.

6.7.1 Recommendations

We would like to conclude the experiment chapter with a brief itemization of the recommendations that originated from our practical experiences gained while working on this thesis.

- **Unsupervised language model adaptation:**
  if the conditions of the task are such that there is no access to the word transcriptions for the target domain, a simple unsupervised language model adaptation described in Sections 2.4.2, 2.4.3 can help out for the calltype classification and NE-processing tasks. This iterative scheme depicted in Figure 2.4 requires only a sufficient amount of audio data for the target domain and suggests alternating recognition of this data and re-estimation of the language model from the collected statistics.

- **Calltype classification:**
  use n-grams (with n up to 5) extracted from the ASR utterances transcripts as classification features for large margin classifiers (Section 3.2). If the low real-time factor is crucial, boosting can be used; otherwise, support vector machines is the preferable choice. To obtain our best results on the HMIHY-3CA corpus (fourth row of Table 6.18), augment the classification features by the system prompt and pre-filter the utterances with special nonterminals (e.g. replacing all the digit words in the transcripts by the special nonterminal

token $digit); refer to Section 6.3.3 for the description. The exact same algorithms of call-type classification (except for the pre-filtering) applied to the output of a phone recognizer is another viable alternative (last row of Table 6.18 for the HMIHY-3CI corpus).

- **Named entity detection:**
  Look at this task as a multiclass multi-label classification task, with each class corresponding to the presence of one named entity type (Section 4.2). Then, the methods for its solution are essentially the same as for the calltype classification. To replicate our best results (third row of Table 6.19) use SVM on word n-grams, with pre-filtering and system prompt as an additional classification feature. The only difference here is that the phone-level NE-detection (last row of the table) works clearly worse than the word-based one.

- **Named entity localization:**
  Carry out ML-parsing of the utterances with handcrafted generic named entity grammars (Section 4.3.2), using FSM-representation. The negative impact of the ASR-errors on the parsing, can be alleviated using approximate matching (Section 4.3.4). Here, the typical ASR-errors that the approximate matching will compensate for, are learned using formulae (4.10) and (4.11) from a pairing of the manual transcriptions and ASR-output of an out-of-domain corpus. Use the results of the upstream detection to parse only those utterances that are likely to contain named entities (Section 6.5.3, formula (6.9)). This will reduce the real-time factor of parsing (boosting) and/or improve localization results (SVM). The best NE-localization (and also value extraction) results at the word level (sixth row in Table 6.20) were achieved using such a combination of approximate matching and pre-filtering by detection with SVMs. The last raw of the same table showing the highest localization rates at the phone level, was obtained in the same manner, but with context-dependent interpolated approximate matching (see the algorithm in Figure 4.7 and formulae (4.14)–(4.16)).

- **Named entity value extraction:**
  Replace finite state acceptors representing named entities for localization by transducers that back-translate they wording into meaning (Section 4.4.2). After that, perform simple normalization of the values and represent them in the format supported by the attended application.

- **Extraction of acoustic morphemes:**
  Acoustic morphemes are intended to represent an application-dependent data-driven alter-

native to words (and common word phrases) in speech. To obtain them, use the *multipass algorithm* from Section 5.1.3. In the first phase (Section 6.6.1) select entropy-reducing phrases applying thresholds on the number of phrase occurrences and its mutual information. Then, prune away all non-salient phrases with a threshold on salience (Section 6.6.2) and semantic significance test described in Section 5.2. Group the selected phrases into acoustic morphemes with the clustering algorithm from Section 5.3 using semantic and acoustic distance metrics as described in Section 6.6.3.

Chapter 7

Further Research

In this chapter we would like to define possible agenda for further research along the lines set up by this thesis.

We will start with the most important general direction for prospective research. We have seen how the problem of automated spoken language understanding in a call center scenario can be tackled from the behavioristic positions, namely by viewing it as a search problem for a causality-bound model delivering an appropriate *reaction* for input *stimuli*. Even though we deployed this strategy in Chapter 6 only for one particular application, porting it to other application domains represents an interesting topic for further research. The changes necessary for such a transition comprise developing new semantic labeling scheme at the utterance level, as well as designing new named entity fragments. While the former has to be done from scratch each time a new domain must be served, one can often reuse the existing named entity fragments, or at least adjust them to fit the new domain needs. For instance, the date grammars we used in our call center application, would have to be adjusted to account for relative dates like *"this Friday"* when used for the ATIS task; at the same time, the cent amount could be excluded from the monetary expressions at all.

The most important question is: where do the limits of the functional understanding in the automated dialog lie? It is clear that solving tasks that involve "real" understanding (like, for instance, consulting services) is not feasible with this approach, but it should be possible to extend it to handle more complex tasks, where the scope of semantics isn't easily exhausted by a linear list of semantic categories. Some steps in this direction have been taken already. So, the latest version of HMIHY supported a hierarchical organization of the calltypes, although the main adjustments that were required to accommodate these changes were made on the part of the dialog manager [Wri02]. The main difficulty to overcome while transferring our entirely non-rationalistic calltype classification technology to a more complex scenario is the exponentially growing number of semantic categories, so that a straightforward classification as we used it in Chapter 3 wouldn't be feasible anymore.

There are several ways to alleviate this problem. For instance, we might introduce elements of hierarchical classification in the task, so that at the first step only a rough categorization took place (does the caller have a question? does he want to complain? or to provide some information?), and then a finer gradation were facilitated by a subsequent classification within each category, whereby an optional specific re-prompt [Gol99] could be used to ensure the correctness of the first-level choice. Another alternative is to consider the action domain as a Cartesian product of simpler spaces [Mil93]. For instance, an account balance inquiry can be decomposed as a product of the question type "information query" and question subject "account balance". Certainly, the growing number of supported actions comes along with an increasing multitude of the relevant types of semantic attributes that also must be taken care of.

One natural extension that is related to the increasing complexity of the application semantics and that we've been trying to avoid so far, is the acknowledgment of the fact that different parts of sentence convey different portions of information. According to the *compositionality principle* [Sis96, Tho98], the meaning of a sentence (and — in the behavioristic approach — the definition of the action to take) is a combination of these portions. While a finer annotation level is necessary to make this principle work, we must be aware of the growing human effort required to create it. Remember that we even relinquished the idea of using manual word transcriptions to minimize these efforts, but in the extreme case where each word from the training sentences is to be provided with its own meaning, the exact opposite would occur. Thus, a reasonable trade-off between complexity of the understanding task and the amount of effort we are willing to invest in this task must be strived at. We would like to conclude this line of discussion with the remark that the finer semantic labeling is provided for a continuous phone stream, the better meaning granularity can be expected from the acoustic morphemes extracted from it.

Luckily, making spoken language understanding systems act as humanlike as possible does not imply that the requirements will increase while the selection of the operational means must remain confined to the output of a recognizer. It is a well known fact that there is much more to the inter-human communication than a simple sentence exchange between two parties. Even in the acoustic channel there is a lot of extra-linguistic information related to such interaction nuances as speaker emotions, his dialectal and sociolectal affiliations, physical conditions etc. An important keyword here is *prosody*, a term that stands for the multitude of supra-segmental phenomena in speech such as intonation, accentuation, speech pauses and others. Taking advantage of the prosodic information to explore the extra-linguistic communicational aspects of speech is one of the challenges the modern spoken language understanding faces, but also a potential source for its improvement [Bat03]. For instance, it has been observed that spotting anger and frustration during the person-to-machine interaction and providing an affect-support is decisive for helping the user to bear with the shortcomings of the system [Kle99]. Alternatively, noticing anger in customer speech is an important indicator that the call should be forwarded to a human operator, even before an attempt of automated dialog is made. It is especially fortunate for the subject of this thesis that the characteristics like speaker conditions, emotions and others can be obtained from the audio signal without using word-level transcriptions [Lev00b, Zei05].

The extraction of acoustic morphemes is one task that we think could profit from incorporating prosodic features into the extraction mechanism. It is known that prosody marks syntactic boundaries in the sentence [Bat98], therefore one would expect that using prosodic information such as pauses, loudness and pitch might help extracting common meaningful words (and word phrases) from audio signal.

Another very important source of information for the automated dialog is the input from the visual sensors. A wide range of multimodal extensions from lip-reading to facial expression and gesture tracking has been shown to improve quality of speech recognition and person-to-machine communication in general [Duc94, Shi03]. Think of the query *"I want to get from here to there."*. Without employing a special gesture analyzer to find out what the user was pointing at as he was uttering words *"here"* and *"there"*, there is no chance for the system to correctly understand the query. For other examples of the verbal utterances being completed or rendered more precise by gestures, see, for instance, [Ken00]. There has been already a solid body of research on combining all these information streams into a one coherent multimodal interface scheme for the automated meaning extraction and — on a higher level — for person-to-machine communication (see, for instance, German SMARTKOM project [Rei03]). The specifics of the integration of different modalities in the context of the goal-oriented behavioristic understanding approach still remain one topic for future research.

In the rest of this chapter we will talk about some concrete technical enhancements that we consider advantageous for the performance of the algorithms presented in this thesis. As far as the calltype classification is concerned, we see an improvement potential in optimizing parameters of support vector machines, such as the selected kernel, its parameters and weights of individual classification errors. Furthermore, LLAMA offers a possibility of automatic selection of the best kernel for each class independently [Lla], albeit that the procedure is extremely time consuming. The second extension possibility stems from the information flowing in with the speech, such as the caller's phone number. Once this number is extracted, through ANI (Automated Number Identification) or an explicite inquiry, the information about the caller can be used to adjust classification parameters. For instance, it has been noticed that caller's monthly expenditure have a great influence on the scope of topics he would likely want to discuss [Gor03]. Analogous extensions are certainly applicable to the task of named entity detection.

For named entity localization and value extraction, we consider it a promising idea to elaborate on the approximate matching mechanism. In particular, we think that a selective context modeling can be integrated in the word-based understanding approach, i.e. we could also model word misrecognitions in context, given that this context is frequent enough to ensure reliable mapping distribution estimates. Another important issue about the localization task concerns the nature of the named entities looked for. We have stressed early in this thesis that our main interest is with the semantic attributes that consist of several words and therefore are prone to frequent (but partial) misrecognitions. On the other hand, especially in the case of proper names, the problem is often the absence of one of the words in the lexicon of the ASR. If the names are present at the latest at runtime, a dynamic modification of the language grammar can take place in the decoder, where generic class-tokens were used during the training stage [Mas04]. Otherwise, at least for name localization, we might, for instance, integrate the out-of-vocabulary (<OOV>) symbol in the model (*"Hello, here is Doctor <OOV> speaking"*), and consider the OOV-possibility at those locations where some word is recognized with a low confidence score. We also would like to keep exploring the chances of an effective integration of parsing language models of higher order from Section 4.3.3 in formula (4.6), despite the first negative results.

Finally, there is a need for more sophisticated language model adaptation. There are many parameters in the basic adaptation scheme in Figure 2.4 that we had to set up intuitively, but that can be optimized explicitely on a validation set. Besides, more intricate language model adaptation schemes could also be considered [Bac03], also in the cases where there *is* some limited transcribed in-domain material.

Chapter 8

Summary

In this thesis we addressed the problem of spoken language understanding without transcriptions in a call center scenario. We started by demonstrating and explaining the difficultly of the automatic understanding task, while stressing its eminent importance for the future scientific progress at the same time. Some of the understanding systems of the past two decades were described and roughly divided in two categories: statistically and linguistically motivated ones. We then presented our choice of the application field, namely the task where the system is to handle spoken language requests like, for instance: *"I don't recognize the number one two three… on my bill"*, under the difficult circumstances of missing in-domain transcriptions for language model training. Based on this choice, we suggested a suitable behaviorist methodology and outlined its individual components.

While discussing the ontology of the understanding systems in the second chapter, we argued that a behaviorist approximation is sufficient for an efficient fulfillment of quite many practical tasks, when facilitating the transfer of the concept of "understanding" to a person-to-machine dialog. This simplistic approach asserts that the measure of understanding is the correctness of the action the hearer takes upon receiving the message, and thus, stipulates a mapping from the space of the (audio) input stimuli into the space of the expected actions. In practice, this mapping is realized by the semantic function of the action or (what is nearly equivalent when dialog and world knowledge are factored out) the semantic category of the spoken utterance that this action is supposed to be elicited by. The semantic functions can optionally be augmented by a number of specific parameters to enhance the modeling capabilities of the approximation. Applied to a typical call center scenario, the task is reduced to a calltype classification plus extraction of semantic attributes which in the application domain of this work are represented by named entities: semantic attributes like names, dates, phone numbers etc. So, in our example above, the system must extract the calltype UNRECOGNIZED_NUMBER from the spoken request, and augment it by a named entity PHONE_NUMBER with a value of *"123... "*. The state of the art in both research fields (calltype classification and named entity processing) was also presented in Chapter 2.

One special issue we specifically addressed in this thesis was imposing an additional restriction on the availability of the transcribed training material for the understanding tasks. Compliant with our global goal of cost minimization for setting up and maintaining the computerized call centers, this restriction strived to evade the need for manual transcription of a large corpus of in-domain data to train a language model optimally representing this domain. In particular, we assumed that no word transcriptions were available for the domain of interest, but rather a sufficient amount of audio data from this domain was accessible, as well as a background language model (of English). To mitigate the additional difficulty caused by the restriction above, we suggested an iterative scheme of unsupervised language model adaptation which, starting from this background language model, gradually readjusted this model to the in-domain language. For the situations where not even a background language model at the word level was available (the case of rare languages or of a discourse domain with the expected lexicon very different from the common lexicon of the language), this adaptation scheme could be employed at the phone level, running on the output of a phone recognizer.

We started Chapter 3 dedicated to the calltype classification task by placing this task in the hierarchy of classification tasks under the category of topic classification and illustrating the differences to the word spotting techniques this task can avail itself of. We explained the notion of salience, as a measure of relevance of some part of a signal for task semantics and showed some of its practical realizations. Then, the use of the large margin classifiers for the calltype classification task was justified. In particular, we focused our attention on support vector machines and boosting: distribution-free classifiers that allow for a very large number of classification features. After explaining the theoretical foundations of these classifiers we presented the toolkits LLAMA (SVM) and BOOSTEXTER (boosting) that we selected for our experiments. Finally, an intuitive training and classification scheme was presented, while taking into account the peculiarity of the classification task in the absence of manual transcriptions for the domain of interest.

In Chapter 4 we were talking about processing of semantic attributes, and named entities in particular. First, we suggested a discrimination between three subtasks in this field, namely detection, localization and value extraction of named entities. We justified the individual importance of each subtask by specifying practical applications that require their solutions. At the same time we also stressed the interconnections between them, one example being the localization procedure preceded by a detection filtering. We suggested to implement detection as a classification (between "present" and "not present" classes) and use SVMs also for this task. For the localization task, named entities were manually modeled as regular grammars and translated into finite state machines. Then, a maximum-likelihood parsing procedure was devised that aimed to represent the corpus in terms of the named entities and words from the lexicon. Several issues pertaining to the localization were addressed, like language model used for parsing, introducing named entity detection scores into the parsing procedure, as well as using approximate matching in it. For the latter, we showed how individual costs of word and phone recognition errors could be estimated from two aligned representations of a validation corpus (reference and ASR-output), and how these costs, encoded as distortion transducers, could then be integrated in the approximate matching framework. For the phone-level case, we also explored the advantages of a context-specific approximate matching. As far as the named entity value extraction is concerned, we grounded its importance in the need for normalization and control of semantic legitimacy of the putative values. To incorporate the value extraction mechanism in our strategy, we extended named entity fragments from finite state acceptors to transducers that back-translate all acceptable surface forms of a named entity into a normalized sequence of meaningful units. Thus, the main part of our localization and value extraction algorithms was realized as the search for the path with the lowest cost through a composition of several finite state transducers.

One issue of automatic spoken language understanding without transcriptions that we addressed in Chapter 5 of this thesis was extraction of word-like units from a continuous phone stream. The problem is critical in the situations where no prior information at all is provided about the language used in the application, or when the language doesn't have a written form. Yet, we have seen in earlier chapters that words play a central role in the language and are a preferable means for solving many of the understanding tasks. We demarcated two defining characteristics of words in the language, namely their utility for the within-channel modeling (entropy-reducing property) and also their innate discriminative meaningfulness in the task (salience). In effect, these information-theoretical qualities form connections between lexicon and syntax, and lexicon and task semantics. After reviewing some of the recent research in this area, we turned to our approach for searching for the word-like units in speech. We designed an iterative algorithm that out of all phone sequences in the training corpus selects the subsequences increasing the likelihood of this corpus when the latter is maximum-likelihood-parsed using them. Employing calltype classification in a call center scenario as a reference task, we then suggested to select only those phone strings (linear sequences) from the extracted set that reveal strong semantic associations with one or several calltypes, whereby special attention was paid to the reliability of the high salience estimates. The final step in our methodology was taking care of not significant variations among the selected phone strings. Using acoustic and semantic distances to assess similarities between individual phone strings, we showed how these strings can be clustered into *acoustic morphemes*, word-like grammar fragments which are robust with respect to misrecognition errors and semantically irrelevant modifications.

Most of the experiments described in this thesis were conducted on the AT&T *"How May I Help You?"* data, a number of corpora of recorded customer utterances made over the phone line. After having introduced these corpora and presented their various statistics, we, firstly, demonstrated the positive impact of our unsupervised language model adaptation scheme on the recognition. Not only were we able to cut the real-time factor in half (from 1.8 to 0.9) but we also increased the word accuracy from 63.1% (with the background language model) to 70.1% (as opposed to 74.7% when the language model was trained on the in-domain transcriptions).

Next, calltype classification was addressed. The first experiment series carried out on the initial dialog utterances provided evidence for the positive effect of the unsupervised language model adaptation on calltype classification. For instance, using SVMs we obtained correct decision rates of 70%, 73% and 76% for background, adapted and transcription-trained language models respectively. Analog experiments conducted using boosting corroborated this conclusion while also witnessing the superiority of the SVMs whose ROC-curves were lying on the average 5% higher. When performing classification on all dialog utterances (instead of just the initial ones), all ROC-curves lay on the average 10% points higher respectively with the correct decision rate of 82% for the adapted language model, because of the lower semantic entropy and fewer introductory parts in the utterances from this corpus. The calltype classification performed using features extracted from the output of a phone recognizer produced equally high classification rates which proved the utility of the phone-based approach for this task. We see here an important potential implication on the future of the call center understanding technology.

For named entity (NE) experiments we focused on three NE-types: DATE, ITEM_AMOUNT, PHONE_NUMBER. Using large margin classifiers we again witnessed the utility of the unsupervised language model adaptation, this time for named entity detection. For instance, with SVM, the detection F-measure (success criterion widely used in the information-retrieval community) for PHONE_NUMBER rose from 0.90 (with background language model) to 0.93 which is the same as when using language model trained on the in-domain transcriptions. Other named entities also showed some improvement but of a lesser extent. However, the phone-based strategy turned out to be less successful for this task; especially for the short named entity DATE where it could only achieve F-measure of 0.65 as opposed to 0.77 with the adapted word language model.

Named entity localization and value extraction could also benefit from the language model adaptation, except for the named entity DATE, because there was a total mismatch in the time stamps of the adaptation and test corpora. For phone numbers, an increase in the localization F-measure from 0.61 to 0.86 was observed, which is as high as the transcription-trained language model could deliver. Furthermore, we proved the importance of pre-filtering by detection by not only reducing the collective parsing time by a factor of up to 3.5, but also improving localization and value extraction results. For instance, the value extraction F-measure grew for named entity ITEM_AMOUNT from 0.50 to 0.53 and for DATE from 0.59 to 0.62. Our techniques could profit from the approximate matching of grammar fragments as well. For example, the value extraction F-measures for DATE, ITEM_AMOUNT and PHONE_NUMBER rose from 0.62 to 0.64, from 0.53 to 0.58 and from 0.76 to 0.80 respectively. Even though, named entity extraction at the phone level came out much weaker than the word-based strategy, it still could produce reasonably nice results, especially when context-dependent approximate matching was used; for instance, we obtained value extraction F-measures of 0.42, 0.55 and 0.59 for DATE, ITEM_AMOUNT and PHONE_NUMBER respectively.

Regarding the extraction of acoustic morphemes, we demonstrated how entropy-reducing salient phone strings could be extracted from continuous phone stream in practice. In terms of the average within-channel utterance logarithmic cost, extracted phone strings caused a reduction from 5.0 to 2.1 (which is close to the average cost of the utterances expressed at the word-level, 1.5), whereby further reduction was possible if one agreed to a larger dictionary of selected phone strings. By varying the enforced salience threshold of the selected phone strings, we recorded an inverse dependency between the average conditional semantic utterance cost and the number of utterances containing such strings. Finally, we have observed that almost all created acoustic morphemes in fact did articulately represent some words or word phrases characteristic for the language and the particular task.

We concluded this thesis with an overview of the future work that can arise from the research that has been done so far. Possible porting of the system to other application domains was considered, and the limits of the presented strategy stated. We also suggested several technical improvements to guide further development of our system, such as employing the compositionality principle and using other extra-linguistic sources of information.

Bibliography

[Abe99] A. Abella and A. L. Gorin. Construct algebra: Analytical dialog management. In *37th Annual Meeting of the Association for Computational Linguistics*, Washington D.C., June 1999.

[Aiz64] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[All95] J. Allen. *Natural Language Understanding*. Benjamin Cummings, Redwood City, California, second edition, 1995.

[Als03] H. Alshawi. Effective utterance classification with unsupervised phonotactic models. In *Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*, Edmonton, Canada, May 2003.

[Alv03] M. Alvarez. HMIHY release 4.0 call type labels DRAFT. Internal technical memo, AT&T Laboratories, June 2003.

[And89] J. A. Anderson and E. Rosenfeld. *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, Massachusetts, 1989.

[App93] D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Meyers, and M. Tyson. SRI international FASTUS system: MUC-6 test results and analysis. In *Proceedings of the 6th Message Understanding Conference*, Columbia, Maryland, 1993.

[App99] D. E. Appelt and D. Martin. Named entity extraction from speech: Approach and results using the Textpro system. In *DARPA Broadcast News Workshop*, pages 51–54, 1999.

[ARP] ARPAbet phonetic alphabet. `http://www.billnet.org/phon/arpabet.html`.

[Bac03] M. Bacchiani and B. Roark. Unsupervised language model adaptation. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 224–227, 2003.

[Bag99] P. Baggia, J. L. Gauvain, A. Kellner, G. Perennou, C. Popovici, J. Sturm, and F. Wessel. Language modelling and spoken dialogue systems – the ARISE experience. In *Proceedings of European Conference on Speech Communication and Technology*, volume 4, pages 1767–1770, Budapest, Hungary, September 1999.

[Bah75] L. R. Bahl and F. Jelinek. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Transactions on Information Theory*, 21, 1975.

[Bal65] G. H. Ball and D. J. Hall. ISODATA : a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, California, 1965.

[Bar01] C. Barras, E. Geoffrois, Z Wu, and M. Liberman. Transcriber: Development and use of a tool for assisting speech corpora production. *Speech Communication*, 33(1):5–22, 2001.

[Bat98] A. Batliner, R. Kompe, A. Kießling, M. Mast, H. Niemann, and E. Nöth. M = syntax + prosody: A syntactic-prosodic labelling scheme for large spontaneous speech databases. *Speech Communication*, 25:193–222, 1998.

[Bat03] A. Batliner and E. Nöth. Prosody and automatic speech recognition - why not yet a success story and where to go from here. In *2nd Plenary Meeting and Symposium on Prosody and Speech Processing*, pages 357–364, Tokyo, Japan, 2003.

[Béc02] F. Béchet, J. Wright, A. Gorin, and D. Hakkani-Tür. Named entity extraction from spontaneous speech in How May I Help You? In *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado, September 2002.

[Béc03] F. Béchet, G. Riccardi, and D. Hakkani-Tur. Multi-channel sentence classification for spoken dialogue language modeling. In *Proceedings of European Conference on Speech Communication and Technology*, Geneva, September 2003.

[Béc04] F. Béchet, A. Gorin, J. Wright, and D. Hakkani-Tur. Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken diologue context: How May I Help You? *Speech Communication*, 42(2):207–225, February 2004.

[Ben67] J. G. Bennett, A. Blake, H. Bortoft, K. Egan, and A. Hodgson. Structural communication. *Systematics*, December 1967.

[Ben97] S. W. Bennett, C. Aone, and C. Lovell. Learning to tag multilingual texts through observation. In *Conference on Empirical Methods in Natural Language Processing*, 1997.

[Ber00] N. O. Bernsen. Speech-related technologies: Where will the field go in 10 years? In *ELSNET Brainstorming Workshop on Speech and Language Research 2000-2010*, Katwijk aan Zee, Netherlands, November 2000. Position Statement.

[Beu99] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal. The AT&T next-gen TTS system. In *Joint Meeting of ASA, EAA and DAGA*, Berlin, Germany, March 1999.

[Big01a] B. Bigi, A. Brun, J.-P. Haton, K. Smaï li, and I. Zitouni. A comparative study of topic identification on newspaper and e-mail. In *8th International Symposium on String Processing and Information Retrieval - SPIRE'01*, Laguna de San Rafael, Chili, 2001.

[Big01b] B. Bigi, A. Brun, K. Smaï li, and J. P. Haton. A hierarchical approach for topic identification. In *International Workshop "Speech and Computer"*, 2001.

[Bik97] D Bikel, S. Miller, R. Schwartz, and R. Weischedel. NYMBLE: a high-performance learning name-finder. In *5th Conference on Applied Natural Language Processing*, pages 194–201, Washington D.C., 1997.

[Bik99] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine Learning; Special Issue on Natural Language Learning*, 34(1–3):211–231, 1999.

[Bob68] D. Bobrow. Natural language input for a computer problem-solving system. In M. Minsky, editor, *Semantic Information Processing*, pages 133–215. MIT Press, Cambridge, Massachusetts, 1968.

[Bor96] M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 1009–1012, Philadelphia, Pennsylvania, 1996.

[Bos92] B. E Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, Pennsylvania, 1992.

[Bre96] M. R. Brent and T. A. Cartwright. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125, 1996.

[Bur98] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[BV03] C. Bousquet-Vernhettes and N. Vigouroux. Recognition error handling by the speech understanding system to improve spoken dialogue systems. In *ISCA workshop on Error Handling in Spoken Dialogue Systems*, Chateau-D'Oex, Switzerland, 2003.

[Che96] S. F. Chen. *Bayesian Grammar Induction for Language Modeling*. PhD thesis, Harvard University, 1996.

[Chi97] N. Chinchor. MUC-7 named entity task definition. In *Proceedings of the 7th Message Understanding Conference*, 1997.

[Cho57] N. Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957.

[Cho59] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[Chu93] K. Church and R. Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, 1993.

[Chu03] K. Church. Speech and language processing: Where have we been and where are we going? In *Proceedings of European Conference on Speech Communication and Technology*, Geneva, Switzerland, 2003.

[Col99] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, College Park, Maryland, June 1999.

[Cor95] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[Cor03] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. *Advances in Neural Information Processing Systems (NIPS 2002)*, 15, March 2003.

[Cov91] T. Cover and J. Thomas. *Information Theory*. Wiley, New York, New York, 1991.

[Cut91] A. Cutler. Segmentation problems, rhythmic solutions. In L. Gleitman and B. Landau, editors, *The Acquisition of the Lexicon*, pages 81–104. MIT Press, Cambridge, Massachusetts, 1991.

[Dav52] S. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *Journal of the Acoustical Society of America*, 24:637–642, 1952.

[Del95] S. Deligne and F. Bimbot. Language modelling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 169–172, Detroit, Michigan, 1995.

[Del97] S. Deligne and F. Bimbot. Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, 23:223–241, 1997.

[Dem77] A. P. Dempster, N. M Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[Den97] M. Denecke and A. Waibel. Dialogue strategies guiding users to their communicative goals. In *Proceedings of European Conference on Speech Communication and Technology*, volume 3, pages 1339–1342, Rhodes, Greece, September 1997.

[dM94] C. de Marcken. The acquisition of a lexicon from paired phoneme sequences and semantic representations. In *International Colloquium on Grammatical Inference*, pages 66–77, Alicante, Spain, 1994.

[dM96] C. de Marcken. The unsupervised acquisition of a lexicon from continuous speech. Technical report, Massachusetts Institute of Technology, 1996.

[Dow93] J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. Gemini: a natural language system for spoken-language understanding. In *31st Annual Meeting of the ACL*, pages 54–61, Columbus, Ohio, June 1993.

[DP92] S. Della Pietra, V. Della Pietra, R. L. Mercer, and S. Roukos. Adaptive language modeling using minimum discriminant estimation. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 633–636, San Francisco, California, March 1992.

[Dre92] H. L. Dreyfus. *What Computers Still Can't Do: a Critique of Artificial Reason*. MIT Press, Cambridge, Massachusetts, October 1992.

[Duc94] P. Duchnowski, U. Meier, and A. Waibel. See me, hear me: Integrating automatic speech recognition and lip-reading. In *Proceedings of International Conference on Spoken Language Processing*, pages 547–550, Yokohama, Japan, September 1994.

[Dud01] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, second edition, 2001.

[Eck96] W. Eckert, F. Gallwitz, and H. Niemann. Combining stochastic and linguistic language models for recognition of spontaneous speech. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 423–426, Atlanta, Georgia, 1996.

[Ehr90] U. Ehrlich. *Bedeutungsanalyse in einem Sprachverstehenden System unter Einbeziehung Pragmatischer Faktoren*. PhD thesis, University Erlangen-Nuremberg, 1990. in German.

[Eps96] M. Epstein, K. Papineni, S. Roukos, T. Ward, and S. Della Pietra. Statistical natural language understanding using hidden clumpings. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 176–179, Atlanta, Georgia, May 1996.

[Fer89] A. Fernald, T. Taeschner, J. Dunn, M. Papousek, B. De Doysson-Baries, and I. Fukui. A cross-linguistic study of prosodic modifications in mothers' and fathers' speech to preverbal infants. *Journal of Child Language*, 16(3):477–502, 1989.

[Fir68] J. Firth. A synopsis of linguistic theory 1930–1955. In F. R. Palmer, editor, *Selected Papers of J. R. Firth 1952–1959*. Longman, London, UK, 1968.

[FJ96] G. D. Forney Jr. The forward-backward algorithm. In *34th Annual Allerton Conference on Communication, Control and Computing*, pages 432–446, Monticello, Illinois, October 1996.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, New York, 1987.

[Fod75] J. A. Fodor. *The Language of Thought*. Thomas Y. Crowell, New York, 1975.

[Fre97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[FSM] FSM-toolkit. `http://www.research.att.com/sw/tools/fsm/`.

[Fur00] S. Furui. Steps toward natural human-machine communication in 21st century. In *ISCA Workshop on Voice Operated Telecom Services*, pages 17–24, Ghent, Belgium, 2000.

[Fur03] S. Furui. Robust methods in automatic speech recognition and understanding. In *Proceedings of European Conference on Speech Communication and Technology*, Geneva, Switzerland, 2003.

[Gal98] F. Gallwitz, M. Aretoulaki, M. Boros, J. Haas, S. Harbeck, R. Huber, H. Niemann, and H. Nöth. The erlangen spoken dialogue system EVAR: A state-of-the-art information retrieval system. In *1998 International Symposium on Spoken Dialogue (ISSD-98)*, pages 19–26, Sydney, Australia, November 1998.

[Ger90] M. A. Gernsbacher. *Language Comprehension as Structure Building*. Lawrence Erlbaum, Hillsdale, New Jersey, 1990.

[Gil98] J. Gillett and W. Ward. A language model combining trigrams and stochastic context-free grammars. In *Proceedings of International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

[Gle91] H. Gleitman. *Psychology*. Norton and Company, New York, New York, third edition, 1991.

[God92] J. Godfrey, E. Holliman, and J. McDaniel. SWITCHBOARD: telephone speech corpus for research development. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 517–520, San Francisco, California, 1992.

[Gol99] J. Golden, O. Kimball, M. Siu, and H. Gish. Automatic topic identification for two-level call routing. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 509–512, 1999.

[Gol00] B. Gold and N. Morgan. *Speech and Audio Signal Processing*. Wiley, New York, New York, 2000.

[Gor91] A. L. Gorin, S. E. Levinson, A. N. Gertner, and E. Goldman. Adaptive acquisition of language. *Computer Speech and Language*, 5(2):101–132, April 1991.

[Gor94a] A. L. Gorin, H. Henck, R. Rose, and L. Miller. Spoken language acquisition for automated call routing. In *Proceedings of International Conference on Spoken Language Processing*, volume 3, pages 1483–1485, Yokohama, Japan, September 1994.

[Gor94b] A. L. Gorin, S. Levinson, and A. Sankar. An experiment in spoken language acquisition. *IEEE Transactions on Speech and Audio Processing*, 2(1):224–240, 1994.

[Gor95] A. L. Gorin. On automated language acquisition. *Journal of the Acoustical Society of America*, 97(6):3441–3461, 1995.

[Gor97] A. L. Gorin, G. Riccardi, and J. H. Wright. How may I Help You? *Speech Communication*, 23:113–127, 1997.

[Gor99] A. L. Gorin, D. Petrovska-Delacrétaz, G. Riccardi, and J. H. Wright. Learning spoken language without transcriptions. In *Proceedings of Automatic Speech Recognition and Understanding*, Keystone, Colorado, December 1999.

[Gor03] A. L. Gorin. Personal communication, 2003.

[Got99] Y. Gotoh, S. Renals, and G. Williams. Named entity tagged language models. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 513–516, Phoenix, Arizona, March 1999.

[GRM] GRM-library. `http://www.research.att.com/sw/tools/grm/`.

[Haa00] J. Haas. *Probabilistic Methods in Linguistic Analysis*. PhD thesis, University Erlangen-Nuremberg, Erlangen, Germany, 2000.

[Hab94] K. Haberlandt. Methods in reading research. In M. A. Gernsbacker, editor, *Handbook of Psycholinguistics*. Academic Press, San Diego, California, 1994.

[Hac01] K. Hacioglu and W. Wardm. Dialog-context dependent language modeling using n-grams and stochastic context-free grammars. In *Proceedings International Conference on Automatic Speech and Signal Processing*, Salt Lake City, Utah, May 2001.

[Haf03a] P. Haffner. Personal communication, 2003.

[Haf03b] P. Haffner, G. Tur, and J. Wright. Optimizing SVMs for complex call classification. In *Proceedings International Conference on Automatic Speech and Signal Processing*, Hong Kong, China, April 2003.

[Hal80] P. A. V. Hall and G. R. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

[Hay91] P. Hayes and S. Weinstein. Construe-TIS: A system for content-based indexing of a database of news stories. In A. Rappaport and R. Smith, editors, *Innovative Applications of Artificial Intelligence 2*, pages 49–64. AAAI Press / MIT Press, Cambridge, Massachusetts, 1991.

[Hei79] H.-W. Hein. Automatische sprachverarbeitung. Arbeitsberichte des IMMD. Technical report, University Erlangen-Nuremberg, Erlangen, Germany, 1979. in German.

[Hob96] J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. FASTUS: a cascaded finite-state transducer for extracting information from natural language text. In Roche and Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1996.

[Hog95] R. V. Hogg and A. T. Criag. *Introduction to Mathematical Statistics*. Prentice-Hall, New Jersey, fifth edition, 1995.

[Hol01] B. Hollister. HMIHY-0300 annotations: History, process, statistics. Technical memo, AT&T Laboratories, 2001.

[Hop79] J. E Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, Massachusetts, 1979.

[Hua94] E.-F. Huang, H.-C. Wang, and F. Soong. A fast algorithm for large vocabulary keyword spotting application. *IEEE Transactions on Speech and Audio Processing*, 2(3):449–452, 1994.

[Hua01] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, New Jersey, 2001.

[Hua02] F. Huang and A. Waibel. An adaptive approach to named entity extraction for meeting applications. In *Human Language Technology Conference*, San Diego, California, 2002.

[Iye98] R. Iyer. *Improving and Predicting Performance of Statistical Language Models in Sparse Domains*. PhD thesis, Boston University, Boston, Massachusetts, 1998.

[Jel90] J. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann Publishers, San Mateo, California, 1990.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98*, pages 137–142, Chemnitz, Germany, 1998.

[Joh87] M. Johnson. *The Body in the Mind: Bodily Basis of Meaning, Imagination, and Reason*. University of Chicago Press, Chicago, Illinois, 1987.

[Kau90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data - an Introduction to Cluster Analysis*. Wiley, New York, New York, 1990.

[Ken00] A. Kendon. Language and gesture: Unity or duality? In D. McNeill, editor, *Language and Gesture*, pages 47–63. Cambridge University Press, Cambridge, UK, 2000.

[Kle99] J. Klein, Y. Moon, and R. W. Picard. This computer responds to user frustration. In *Conference on Human Factors in Computing Systems*, pages 242–243, Pittsburgh, Pennsylvania, 1999. Special Interest Group on Computer-Human Interaction.

[Klö03] H. Klöter. *Written Taiwanese*. PhD thesis, Leiden University, Leiden, Netherlands, 2003.

[Kne93] R Kneser and V. Steinbiss. On the dynamic adaptation of stochastic language modeling. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 2, pages 586–589, Minneapolis, Minnesota, 1993.

[Kni96] K. M. Knill and S.J. Young. Fast implementation methods for viterbi-based word-spotting. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 522–525, Atlanta, Georgia, 1996.

[Kub98] F. Kubala, R. Schwartz, R. Stone, and R. Weischedel. Named entity extraction from speech. In *DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, February 1998.

[Kuh90] R. Kuhn and R. De Mori. A cache-based natural language model for speech reproduction. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

[Kum91] F. Kumert. *Flexible Steuerung eines Sprachverstehenden Systems mit Homogener Wissensbasis*. PhD thesis, University Erlangen-Nuremberg, 1991. in German.

[Lan95] K. Lang. Newsweeder: Learning to filter netnews. In *12th International Conference on Machine Learning*, pages 331–339, 1995.

[Lev00a] E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Fabbrizio, W. Eckert, S Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. The AT&T Darpa Communicator mixed-initiative spoken dialogue system. In *Proceedings of International Conference on Spoken Language Processing*, Beijing, China, 2000.

[Lev00b] M. Levit. Benutzung von sprachcharakteristika zur klassifikation von sprachvarietäten und sprecherzuständen. Master's thesis, University Erlangen-Nuremberg, Erlangen, Germany, 2000. in German.

[Lev01] M. Levit, A. L. Gorin, and J. H. Wright. Multipass algorithm for acquisition of salient acoustic morphemes. In *Proceedings of European Conference on Speech Communication and Technology*, Aalborg, Denmark, September 2001.

[Lev02] M. Levit, A. L. Gorin, and E. Nöth. Using EM-trained string-edit distances for approximate matching of acoustic morphemes. In *Proceedings of International Conference on Spoken Language Processing*, pages 1157–1160, Denver, Colorado, 2002.

[Lev03] M. Levit, H. Alshawi, A. Gorin, and E. Nöth. Context-sensitive evaluation and correction of phone recognition output. In *Proceedings of European Conference on Speech Communication and Technology*, pages 925–928, Geneva, Switzerland, 2003.

[Lev04] M. Levit, P. Haffner, A. Gorin, H. Alshawi, and E. Nöth. Aspects of named entity processing. In *Proceedings of International Conference on Spoken Language Processing*, Jeju Island, South Korea, 2004.

[LIB] LIBSVM-toolkit. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[Lin80] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, January 1980.

[Lip97] R. P. Lippmann. Speech recognition by machines and humans. *Speech Communication*, 22:1–15, 1997.

[Ljo00] A. Ljolje, D. Hindle, M. Riley, and R. Sproat. The AT&T LVCSR-2000 system. In *NIST LVCSR Workshop*, 2000.

[Lla] Llama-toolkit. `http://www.research.att.com/~haffner/llama`. AT&T internal link.

[Mas04] S. Maskey, M. Bacchiani, B. Roark, and R. Sproat. Improved name recognition with meta-data dependent name networks. In *Proceedings International Conference on Automatic Speech and Signal Processing*, Montreal, Canada, May 2004.

[McD94] J. McDonough, K. Ng, P. Jeanrenaud, H. Gish, and J. R. Rohlicek. Approaches to topic identification on the Switchboard corpus. In *Proceedings International Conference on Automatic Speech and Signal Processing*, volume 1, pages 385–388, Adelaide, Australia, April 1994.

[McQ67] J. B. McQueen. Some methods of classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematics Statistics and Probability*, volume 1, pages 281–297, Berkeley, California, 1967.

[Med] Media mining system. `http://www.sail-technology.com`.

[Mer] Merriam-Webster English dictionary online. `http://www.m-w.com/`.

[Mil93] L. G. Miller and A. L. Gorin. Spoken language acquisition in an almanac data retrieval task. Technical memo, AT&T Bell Laboratories, 1993.

[Mil99] D. Miller, R. Schwartz, R. Weischedel, and R. Stone. Named entity extraction from broadcast news. In *DARPA Broadcast News Workshop*, pages 37–40, 1999.

[Moh97] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, June 1997.

[Moh02] M. Mohri and M. Riley. Weighted finite-state transducers in speech recognition; tutorial. In *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado, September 2002.

[Mor86] J. Morgan. *From Simple Input to Complex Grammar*. MIT Press, Cambridge, Massachusetts, 1986.

[Mur83] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26(4):354–359, 1983.

[Mye00] K. Myers, M. Kearns, S. Singh, and M. A. Walker. A boosting approach to topic spotting on subdialogues. In *7th International Conference on Machine Learning*, pages 655–662, Stanford, California, 2000.

[Nat02] P. Natarajan, R. Prasad, B. Suhm, and D. McCarthy. Speech-enabled natural language call routing: BBN Call Director. In *Proceedings of International Conference on Spoken Language Processing*, pages 1161–1164, Denver, Colorado, 2002.

[Nel73] K. Nelson. Structure and strategy in learning to talk. *Monographs of the Society for Research in Child Development*, 38, 1973.

[New90] E. Newport. Maturational constraints on language learning. *Cognitive Science*, 14:11–28, 1990.

[Nie] H. Niemann. Klassifikation von mustern. `http://www5.informatik.uni-erlangen.de/MEDIA/nm/klassifikation-von-mustern/m00links.html`. second edition, in German.

[Nie90] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: a semantic network system for pattern understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:883–905, 1990.

[NIS] NIST named entity definition. `http://www.itl.nist.gov/iad/894.02/related_projects/muc/index.html`.

[Oli68] D. Olivier. *Stochastic Grammars and Language Acquisition Mechanisms*. PhD thesis, Harvard University, Cambridge, Massachusetts, August 1968.

[Pal99] D. Palmer, M. Ostendorf, and J. Burger. Robust information extraction from spoken language data. In *Proceedings of European Conference on Speech Communication and Technology*, pages 1035–1038, Budapest, Hungary, 1999.

[Ped96] T. Pedersen. Fishing for exactness. In *South Central SAS User's Group (SCSUG-96) Conference*, Austin, Texas, October 1996.

[Pie92] R. Pieraccini, E. Tzoukermann, Z. Gorelov, E. Levin, C. H. Lee, and J.-L. Gauvain. Progress report on the Chronus system: ATIS benchmark results. In *5th DARPA Speech and Natural Language Workshop*, Harriman, New York, February 1992.

[Pie95] R. Pieraccini and E. Levin. A learning approach to natural language understanding. In A. J. Rubio Ayuso and J. M. López Soler, editors, *New Advances and Trends in Speech Recognition and Coding*, volume 147 of *NATO ASI*. Springer-Verlag, Berlin Heidelberg, Germany, 1995.

[Pin99] S. Pinker. *Words and Rules. The Ingredients of Language*. Weidenfeld and Nicolson, New York, New York, 1999.

[Pra04] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*, 2004.

[Pri90] P. Price. Evaluation of spoken language systems: the ATIS domain. In *DARPA Speech and Natural Language Workshop*, Hidden Valley, Pennsylvania, June 1990.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Rea88] T. Read and N. Cressie. *Goodness-of-fit Statistics for Discrete Multivariate Data*. Springer Series in Statistics. Springer-Verlag, New York, New York, 1988.

[Ree96] B. Reeves and C. Nass. *The Media Equation: How People Treat Computers, Television and New Media Like Real People and Places*. Cambridge University Press, Cambridge, UK, 1996.

[Rei03] N. Reithinger, J. Alexandersson, T. Becker, A. Blocher, R. Engel, M. Löckelt, J. Müller, N. Pfleger, P. Poller, M. Streit, and V. Tschernomas. SmartKom - adaptive and flexible multimodal access to multiple applications. In *5th International Conference on Multimodal Interfaces*, pages 101–108, Vancouver, Canada, 2003.

[Ric97] G. Riccardi, A. L. Gorin, A. Ljolje, and M. Riley. A spoken language system for automated call routing. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 1143–1146, Munich, Germany, 1997.

[Ric03] G. Riccardi and D. Hakkani-Tür. Active and unsupervised learning for automatic speech recognition. In *Proceedings of European Conference on Speech Communication and Technology*, Geneve, Switzerland, September 2003.

[Rie94] C. K. Riesbeck and W. Fitzgerald. Language understanding is recognition, not construction. *Psycholoquy: Language Comprehension*, 5(38), 1994.

[Ris89] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15 of *Series in Computer Science*. World Scientific, 1989.

[Ris98] E. S. Ristad and P. N. Yianilos. Learning string edit distance. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(5):522–532, May 1998.

[Ros95] R. Rosenfeld. Exploiting remote domains via data bleaching. Technical report, Johns Hopkins Workshop on Language Modeling, 1995. Language Modeling of Spontaneous Speech Summer Research Project.

[Roy99] D. Roy. *Learning Words from Sights and Sounds: A Computational Model*. PhD thesis, Massachusetts Institute of Technology, September 1999.

[Roy03] D. Roy, K-Y. Hsiao, N. Mavridis, and P. Gorniak. Ripley, hand me the cup: Sensorimotor representations for grounding word meaning. In *Proceedings of Automatic Speech Recognition and Understanding*, 2003.

[Rut00] P. Rutten, G. Coorman, J. Fackrell, B. Van Coile, and “Lernout & Hauspie Speech Products”. Issues in corpus based speech synthesis. In *Symposium on State-of-the-Art in Speech Synthesis*, pages 1–16, London, UK, April 2000. IEE.

[San93] A. Sankar, A. L. Gorin, J. G Wilpon, S. Y. Lee, R. Venkararamani, and D. Bock. Language acquisition for automated call routing in the telephone network. Technical memo, AT&T Bell Laboratories, 1993.

[Sch85] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 1205–1208, Tampa, Florida, 1985.

[Sch90] R. J. Scholes and B. J. Willis. Prosodic and syntactic functions of punctuation: A contribution to the study of orality and literacy. *Interchange*, 21(3):13–20, 1990.

[Sch96] M. Schmidt. Identifying speakers with support vector networks. In *Interface'96*, Sydney, Australia, 1996.

[Sch98] Y. Schapire, R. E. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[Sch00] R. E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

[Sch02] R. E. Schapire. The boosting approach to machine learning: an overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[Sek02] S. Sekine, K. Sudo, and C. Nobata. Extended named entity hierarchy. In *International Conference on Language Resources and Evaluation*, Canary Island, Spain, 2002.

[Sen92] S. Seneff. TINA: a natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86, 1992.

[Sey97] K. Seymore and R. Rosenfeld. Large-scale topic detection and language model adaptation. Technical Report CMU-CS-97-152, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379–423, July 1948.

[Sha51] C. Shannon. Prediction and entropy of written English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[Sha97] R. D. Sharp, E. Bocchieri, C. Castillo, S. Parthasarathy, C. Rath, M. Riley, and J. Rowland. The Watson speech recognition engine. In *Proceedings International Conference* *on Automatic Speech and Signal Processing*, volume 5, pages 4065–4068, Munich, Germany, 1997.

[Shi85] S. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343, 1985.

[Shi03] R. P. Shi, J. Adelhardt, V. Zeissler, A. Batliner, A. Frank, E. Nöth, and H. Niemann. Using speech and gesture to explore user states in multimodal dialogue systems. In *International Conference on Audio-Visual Speech Processing*, pages 151–156, Grenoble, 2003.

[Sik72] L. Siklóssy. Natural language learning by computer. In H. A. Simon and L. Siklóssy, editors, *Representation and Meaning: Experiments with Information Processing Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[Sis96] J. M. Siskind. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1):39–91, 1996.

[Smo00] A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans. Introduction to large margin classifiers. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, Massachusetts, 2000.

[Sto94] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California, Berkeley, California, 1994.

[SVM] SVMlight-toolkit. `http://svmlight.joachims.org`.

[Tan03] M. Tang, B. Pellom, and K. Hacioglu. Call-type classification and unsupervised training for the call center domain. In *Automatic Speech Recognition and Understanding Workshop*, St. Thomas, US Virgin Islands, December 2003. IEEE.

[Tho98] C. A. Thompson. *Semantic Lexicon Acquisition for Learning Natural Language Interfaces*. PhD thesis, University of Texas, Austin, Texas, 1998.

[Tur92] A. Turing. Intelligent machinery. In D. Ince, editor, *Collected Works of A. M. Turing: Mechanical Intelligence*, volume 3, pages 107–127. Elsevier Science Publishers, Amsterdam, Netherlands, 1992.

[VA02] Y. Vazquez Alvarez and M. Huckvale. The reliability of the ITU-T P.85 standard for the evaluation of text-to-speech systems. In *Proceedings of International Conference on Spoken Language Processing*, pages 329–332, Denver, Colorado, 2002.

[Vap82] V. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer Verlag, New York, New York, 1982.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, New York, 1995.

[vdM03] A. van der Mude, N. Gupta, and G. Tur. Developer's guide to using the AT&T NLU Toolset. Technical memo, AT&T Laboratories, 2003.

[War91] W. H. Ward. Understanding spontaneous speech: the Phoenix system. In *Proceedings International Conference on Automatic Speech and Signal Processing*, pages 365–368, Toronto, Canada, May 1991.

[War97] V. Warnke, S. Harbeck, H. Niemann, and E. Nöth. Topic spotting using subword units. In *9es Aachener Kolloquium Signaltheorie, Bild- und Sprachsignale*, pages 287–291, Aachen, Germany, 1997.

[Wei66] J. Weizenbaum. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[Whi02] S. Whittaker, J. Hirschberg, B. Amento, L. Stark, M. Bacchiani, P. Isenhour, L. Stead, G. Zamchick, and A. Rosenberg. SCANMail: a voicemail interface that makes speech browsable, readable and searchable". In *Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, 2002. Special Interest Group on Computer-Human Interaction.

[Who56] B.L. Whorf. Science and linguistics. In J. B. Carroll, editor, *Language,Thought and Reality*. MIT Press, Cambridge, Massachusetts, 1956.

[Win73] T. Winograd. A process model of language understanding. In Schank and Colby, editors, *Computer Models of Thought and Language*, pages 152–186. Freeman, 1973.

[Win80] T. Winograd. What does it mean to understand natural language. *Cognitive Science*, 4:209–241, 1980.

[Wit21] L. Wittgenstein. *Tractatus Logico-Philosophicus*. Routledge and Kegan Paul, London, 1921. translated by D. F. Pears and B. F. McGuiness.

[Wit68] L. Wittgenstein. *Philosophical Investigations*. Macmillan, New York, New York, 1968. translated by G. E. M. Anscombe.

[Woo98] P. Woodland, T. Hain, G. Moore, T. Niesler, D. Povey, A. Tuerk, and E. Whittaker. The 1998 HTK broadcast news transcription system: Development and results. In *DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[Wri97] J. H. Wright, A. L. Gorin, and G. Riccardi. Automatic acquisition of salient grammar fragments for call-type classification. In *Proceedings of European Conference on Speech Communication and Technology*, pages 1419–1422, Rhodes, Greece, 1997.

[Wri02] J. H. Wright, A. Abella, and A. L. Gorin. Unified task knowledge for spoken language understanding and dialog management. In *Proceedings of International Conference on Spoken Language Processing*, pages 1157–1160, Denver, Colorado, 2002.

[Yan97] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *14th International Conference on Machine Learning*, pages 412–420, Nashville, Tennessee, 1997.

[Yok03] Y. Yokoyama, T. Shinozaki, K. Iwano, and S. Furui. Unsupervised language model adaptation using word classes for spontaneous speech recognition. In *Workshop on Spontaneous Speech Processing and Recognition*, pages 71–74, Tokyo, Japan, 2003. IEEE-ISCA.

[Zee88] H. Zeevat. Combining categorial grammar and unification. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 202–229. Reidel, Dordrecht, Netherlands, 1988.

[Zei05] V. Zeissler. *Robuste Erkennung der Prosodischen Phänomene und der Emotionalen Benutzerzustände in einem Multimodalen Dialogsystem*. PhD thesis, University Erlangen-Nuremberg, to appear in 2005. in German.

[Zip35] G. Zipf. *The Psycho-Biology of Language*. Houghton Mifflin, Boston, Massachusetts, 1935.

[Zue92] M. Zue, J. Glass, D. Goddeau, D. Goodine, L. Hirschman, M. Phillips, J. Polifroni, and S. Seneff. The MIT ATIS system: February 1992 progress report. In *5th DARPA Speech and Natural Language Workshop*, pages 84–88, Harriman, New York, February 1992.

Index

Appendix A

Mathematical Nomenclature in this Thesis

symbol	meaning
$(\cdot)^j$	superscript indices are usually used to denote the element's number in a lexicon
$(\cdot)_i$	subscript indices are usually used to denote the position of an element in a time series
t, τ	time or iteration number (runs from 1 to T and from 1 to U)
a	acoustic unit
A	sequence thereof
w	word, there are J of the them in the lexicon
W	sequence thereof
c	semantic concept, semantic category, calltype etc. (there are M of them in the semantic lexicon)
C	time sequence thereof (where applicable)
$P(\cdot)$	probability function
$\psi(\cdot)$	semantic function of the aspired action
$\varLambda^{\psi}$	definition domain of $\psi(\cdot)$
$\boldsymbol{\nu} \in \varLambda^{\psi}$	parameter vector; each element ν_k being e.g. a named entity
L	some formal or natural language
$\varSigma^{L}$	its lexicon
κ	language model weight

Table A.1: Symbols first introduced in Chapters 1 and 2

symbol	meaning
d	dimension of the feature vectors
$\boldsymbol{x}$ (or $\boldsymbol{y}$)	d-dimensional vector of classification features (there are I of them)
y	corresponding reference label for some feature vector $\boldsymbol{x}$
$g(\boldsymbol{x})$	binary decision function used in the large margin classification
$G(\boldsymbol{x})$	real-valued linear function that yields $g(\boldsymbol{x})$
$\boldsymbol{w}$	normal vector to a separating hyperplane $G(\boldsymbol{x}) = 0$
b	translation constant that, together with $\boldsymbol{w}$ defines the hyperplane $G(\boldsymbol{x}) = 0$
ρ_G	classification margin: distance from a feature vector (or a set thereof) to the separating hyperplane
ϱ	radius of a ball containing training examples (3.11)
λ_i	Lagrange multiplies in the SVM-classification
$\Psi(\boldsymbol{x})$	transition function into a higher dimensional space $\mathbb{H}$ (SVM)
$K(\boldsymbol{x}, \boldsymbol{y})$	corresponding kernel function
n	as in n-gram
$\mathcal{L}$	(weighted) lattice, most general kind of an ASR-output
S	some word/phone sequence (or its segmentation) through the lattice
s	sequence (*string*) of (usually adjacent) words/phones w
$\#s$	number of occurrences of s
$\mathcal{D}_t$	distribution of the training examples on tth iteration (boosting)
$h(\boldsymbol{x}, y)$	weak classifier
h_t	weak classifier selected on iteration t
μ_t	weighting factor of this weak classifier
Z_t	normalization factor on iteration t
$\mathcal{Y}$	label space in the multi-label case, as opposed to a simple 2-class classification where this space is $\{-1, 1\}$

Table A.2: Symbols first introduced in Chapter 3

symbol	meaning
ν	named entity (class); there are K of them
$A^k(\cdot)$	binary detection function
θ	threshold (on NE-detection probability, distances etc.)
$\boldsymbol{\theta}$	vector of such thresholds
$(\mathcal{N}, \mathcal{T}, \mathcal{P}, N^s)$	formal grammar definition
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	sequences of terminals and nonterminals (formal grammars)
$(\mathcal{Q}, \Sigma^I, \delta, Q^s, \mathcal{Q}^F)$	finite automaton definition
$\mathbb{K}$	semiring over which a weighted finite automaton is defined
ϕ, ϕ^{NE}	fragment and, in particular, named entity grammar fragment
Σ^{NE}	lexicon for NE-localization parsing
q	similar to s but refers to intended strings as opposed to observed strings
Q	sequence of q
Φ	sequence of ϕ
$n_{\max}$	order of the language model (for parsing)
v	same as w but usually used when talking about intended strings q
ε	empty symbol
π	alignment path between two strings
$\alpha_{t,\tau}, \beta_{t,\tau}, \gamma$	see EM-algorithm
ξ	interpolation coefficients
$\mathcal{LG}, \mathcal{D}, \mathcal{S}, \mathcal{F}$	finite state machines encoding lexicon with the grammar, distortion transducer, ASR-output and the parsing result respectively

Table A.3: Symbols first introduced in Chapter 4

symbol	meaning
$D(\cdot,\cdot)$	distance between two elements (vectors, strings or clusters thereof)
$\boldsymbol{d}$	vector of several distances (of different nature)
$X^2()$	Pearson's function
χ^2	as in χ^2 distribution
N, N_m	sample size and number of time some event occurred in the sample
$\boldsymbol{n}$	vector of the occurrence statistics
p	prior probability of an event
r	partition of a sample in M classes
X, x	random variable and it's value (for significance tests)
$F(x)$	cumulative distribution function of x
α	probability threshold in significance test
$\zeta_{s/i/d}$	Levenshtein cost for substitution/insertion/deletion
O	cluster of phone strings
o	its centroid
$\mathcal{H}_0$	null-hypothesis in a statistical test
$\mathit{\Theta}$	contingency table for Fisher's significance test

Table A.4: Symbols first introduced in Chapter 5

symbol	meaning
P, R, F	precision, recall and F-measure
$\mathcal{R}, \mathcal{H}$	spaces of references and hypotheses for NE-localization evaluation
ς	sigmoid function (taking detection score into account for localization)
ω	normalized parsing cost of an utterance
$\mathit{\Omega}$	average parsing cost of a corpus

Table A.5: Symbols first introduced in Chapter 6

Appendix B

Calltypes in this Work

In this appendix we list all calltypes used in our calltype classification experiments.

B.1 Experiments on Initial Utterances

```
900_Services
Account_Balance
Billing_Credit
Billing_Services
Cable_Broadband
CallATT
Calling_Card
CallingPlan
Call_Local_Co
Call_Wireless
CancelPlan
Charge_on_Bill
Combined_Bill
Competitor
Current_Plan
Customer_Information
Customer_Rep
Disconnect
Duplicate_Bill
```

```
EasyReach_800
Explanation_of_Bill
Extra_Services
FreeMinutes
General_Services
Hello
Help
International
LDCheck
LocalServices
Local_Toll
Long_Distance
New_Service
Operator_and_00Info
Other
Pay_Bill
ProductInfo
Rate_Calling_Plans
RemoveBlock
Repair
RequestBlock
Rewards
Separate_Bill
Spanish
Threshold_Billing
Universal_Connectivity
Unrecognized_Number
Usage_Minimum
Where_to_Mail
WorldNet
```

B.2 Experiments on All Utterances

```
900_Services
```

```
Account_Balance
Billing_Credit
Billing_Services
Cable_Broadband
CallATT
Calling_Card
CallingPlan
Call_Local_Co
Call_Wireless
CancelPlan
Charge_on_Bill
Combined_Bill
Competitor
Current_Plan
Customer_Information
Customer_Rep
Disconnect
Duplicate_Bill
EasyReach_800
Explanation_of_Bill
Extra_Services
FreeMinutes
General_Services
Hello
Help
International
LDCheck
LocalServices
Local_Toll
Long_Distance
New_Service
No
Operator_and_00Info
Other
```

```
Pay_Bill
ProductInfo
Rate_Calling_Plans
RemoveBlock
Repair
Repeat
RequestBlock
Rewards
Separate_Bill
Spanish
Threshold_Billing
Universal_Connectivity
Unrecognized_Number
Usage_Minimum
Where_to_Mail
WorldNet
Yes
```

Appendix C

Named Entities in this Work

C.1 Definitions and Examples of Named Entities

Next we list the named entities that we considered in the experimental part of this thesis, explain them and give positive and negative examples. All named entity types are based on the corresponding prototype definitions from the HMIHY Labeling Guide [Alv03], although a few diversions were made.

C.1.1 Named Entity DATE

These are complete or partial date expressions. Either the year or the month of the date expression has to be specified, and no relative dates information (*yesterday*, *last month*) are accepted. The named entity can be seen as a subclass of the TIMEX named entity class from MUC-7 [Chi97].

Positive examples:

…*this call from second May*
…*my bill from June ninety nine*
…*I paid it in full last Friday, November thirty first*

Negative examples:

I may first consider other options…	(not a date at all)
…*my bill from last month…*	(relative dates are not accepted)

. . . I paid it in full on Friday	(month or year must be present)

C.1.2 Named Entity ITEM_AMOUNT

Under this named entity we understand any expression referring to a monetary amount *shown on the bill*. Thus, this named entity is context-dependent [Béc04], which means that it is not sufficient to examine only the "core"-part of the putative candidate (like, for example, *five dollar* or *dollar twenty*) to determine whether it is from this NE-type or not, but also its context must be taken into account.

Positive examples:

. . . charged me two dollars and ten cents for a call
the amount I owe you is seventy dollars

Negative examples:

. . . charge me ten cent a minute	(not an item, but rate)
I sent you a check for thirty four dollars	(not an item on the bill)

C.1.3 Named Entity PHONE_NUMBER

This NE-type stands for references to arbitrary phone numbers. Even though, we restrict our modeling grammar only to well-formed phone numbers within US, phone numbers of any kind are considered as of this type. The order in which the number and the area code are specified is not important.

Positive Examples

I don't recognize this call to 4567890 area code huh 123
my home number is 1234567
. . . and then I called 1 800 0000000. . .

Negative Examples

account number is 1234567890123456	(not a phone number)
. . . my credit card. Its number is 1234567. . .	(not a phone number)

C.2 Named Entity Grammars

In this section of the appendix we explain the format we used to create regular grammars [vdM03], and then print all of the grammars used to encode named entities explored in this thesis.

Each rule of the grammar is of the form <leading-nonterminal> = *rule body*, with the rule body consisting of one or several alternative rules (separated by "|"). The rules with different leading nonterminals are separated by semicolons. All nonterminals are enclosed in angle brackets, while the terminals are not. Missing leading "@" in the name of a nonterminal, results in an enclosure of the instantiated rule body in the special marker-parentheses, so that the processed instance then looks something like:

*<month>***june** *</month> <day-of-month>***first** *</day-of-month>*

Additionally, nonterminals can be translated into other nonterminals. To achieve that, they must be provided by a colon followed with the new symbol. Thus, special nonterminals *OpAdd* and *OpMultiply* are used to facilitate arithmetic operations at the value extraction stage (see the grammars bellow). The same holds for terminals as well. For instance, rule

< @one-dollar> = dollar:1;

will transform string *"dollar"* into string *"1"*. Square brackets mean an optional element, that can be present or not in the instance.

In the following, the grammars used to create named entity grammar fragments are presented. First, we show the baseline grammars with no syntactic context included. Next, extended versions with minimal context inclusion are shown, which we used along with the approximate matching strategy. All grammars are based on the previous work by George Kiraz and Boris Smilga.

C.2.1 Baseline Grammars

DATE-specific part

```
<START> = <cls_DATE>;
```

```
<cls_DATE> =
   [<day-of-week>] <month> [the] <day-of-month> [ [of] <year>] |
   [<day-of-week>] [ [the] <day-of-month> of ] <month> [ [of] <year>] |
   <@year_full:year> ;


<year> =
   <@year>;

<month> =
   <@month>;

<day-of-month> =
   <@day-of-month>;

<day-of-week> =
   <@day-of-week> ;

// *********************
// *** Year (1991+)  ***
// *********************

<@year> =
   <@year_2000s> |
   <@year_1990s> ;

<@year_full> =
   <@year_2000s>; //|   <@year_1990s_full> ;

<@year_2000s> =
   <@year_2000s_aux:OpAdd> ;

<@year_2000s_aux> =
   <@year_2000s_aux2:OpMultiply> [and] <@numrange_1-9> |
   <@year_2000s_aux2:OpMultiply> ;

<@year_2000s_aux2> =
   <nVal.2> <@thousand> ;

<@year_1990s> =
   <@year_1990s_aux:OpAdd> ;

<@year_1990s_aux> =
   [<nVal.19:nVal.1900>] <nVal.90> <@numrange_1-9> ;

<@year_1990s_full> =
   <@year_1990s_full_aux:OpAdd> ;

<@year_1990s_full_aux> =
   <nVal.19:nVal.1900> <nVal.90> <@numrange_1-9> ;

// ********************
// *** DAY OF MONTH  ***
// ********************

<@day-of-month> =
   <@numrange_01-09> |
   <@numrange_11-19> |
   <nVal.10> |
   <nVal.20> |
   <nVal.30> |
   <@day-of-month_aux:OpAdd> |
   <@numrange_ord_1-31> ;

<@day-of-month_aux> =
   <nVal.20> <@numrange_1-9> |
   <nVal.30> <@nVal.1> ;
```

```
// ********************
// *** MONTHS,  DAYS ***
// ********************

<@month> =
   <@month_excluding_may> |
   <@month_of_may> ;

<@month_excluding_may> =
   <nVal.00001> |
   <nVal.00002> |
   <nVal.00003> |
   <nVal.00004> |
   <nVal.00006> |
   <nVal.00007> |
   <nVal.00008> |
   <nVal.00009> |
   <nVal.000010> |
   <nVal.000011> |
   <nVal.000012> ;

<@month_of_may> =
//   month of
   <nVal.00005> ;


// ********************
// ***** TERMINALS *****
// ********************

<@day-of-week> =
   sunday    |
   monday    |
   tuesday   |
   wednesday |
   thursday  |
   friday    |
   saturday  ;


// months
<nVal.00001> = january   ;
<nVal.00002> = february  ;
<nVal.00003> = march     ;
<nVal.00004> = april     ;
<nVal.00005> = may       ;
<nVal.00006> = june      ;
<nVal.00007> = july      ;
<nVal.00008> = august    ;
<nVal.00009> = september ;
<nVal.000010> = october   ;
<nVal.000011> = november  ;
<nVal.000012> = december  ;
```

ITEM_AMOUNT-specific part

```
<START> = <cls_CURRENCY> ;

<cls_CURRENCY> =
   <@dollars> [ <@simple-cents> | ( [and] <@cents> ) ]
   | <@one-dollar:dollar-amount> <@simple-cents>
   | <@cents>
   | <@simple-dollars> <@simple-cents>
;



/* ******************************
```

```
   local context
   ***************************** */
<@one-dollar> = dollar:1;

<@dollars> =
   <dollar-amount> (dollar | dollars);

<@simple-dollars> =
   <dollar-amount> [dollar | dollars];

<@cents> =
   <cent-amount> (cent | cents) ;

<@simple-cents> =
   <@simple-cent-amount:cent-amount> [ cent | cents ] ;

<@simple-cent-amount> =
   <nVal.10> |
   <@numrange_11-19> |
   <@numrange_20-99> ;

<dollar-amount> =
  <@numrange_01-999> ;

<cent-amount> =
   <@numrange_01-99> ;
```

PHONE_NUMBER-specific part

```
<START> = <cls_PHONENUMBER> ;

<cls_PHONENUMBER> =
   ( [ [<@my_area_code>] <areacode> ] <phoneno> ) |
   ( <phoneno> [and] <@my_area_code> <areacode> )
   ;
```

```
<phoneno> =
   <@3digits_nlz> <@4digits> ;

<areacode> =
   [ <nVal.1> ] <@3digits_nlz> ;


/* ******************************
   local context
   ****************************** */

<@my_area_code> = [area] code | area [code] ;
```

Common part

```
// *************************
// ***** NUMBER RANGES *****
// *************************

// for months

<@numrange_ord_1-31> =
   <@numrange_ord_1-10> |
   <@numrange_ord_11-19> |
   <nVal.00020> |
   <nVal.00030> |
   <@numrange_ord_1-31_aux:OpAdd> ;

<@numrange_ord_1-31_aux> =
   <nVal.20> <@numrange_ord_1-9> |
   <nVal.30> <nVal.0001>;
```

```
// 10s

<@numrange_ord_1-99> =
   <@numrange_ord_1-10> |
   <@numrange_ord_11-19> |
   <@numrange_ord_20-99> ;

<@numrange_ord_20-99> =
   <@tens_ord> |
   <@numrange_ord_20-99_aux:OpAdd>;

<@numrange_ord_20-99_aux> =
   <@tens> <@numrange_ord_1-9> ;

<@numrange_ord_11-19> =
   <nVal.00011> |
   <nVal.00012> |
   <nVal.00013> |
   <nVal.00014> |
   <nVal.00015> |
   <nVal.00016> |
   <nVal.00017> |
   <nVal.00018> |
   <nVal.00019> ;

<@numrange_ord_1-10> =
   <@numrange_ord_1-9> |
   <nVal.00010>     ;

<@numrange_ord_1-9> =
   <@numrange_ord_1-2> |
   <nVal.0003>     |
   <nVal.0004>     |
   <nVal.0005>     |
   <nVal.0006>     |
   <nVal.0007>     |
   <nVal.0008>     |
   <nVal.0009>     ;

<@numrange_ord_1-2> =
   <nVal.0001> |
   <nVal.0002> ;

// tens

<@tens_ord> =
   <nVal.00020> |
   <nVal.00030> |
   <nVal.00040> |
   <nVal.00050> |
   <nVal.00060> |
   <nVal.00070> |
   <nVal.00080> |
   <nVal.00090> ;

// ********************
// ***** TERMINALS *****
// ********************

// terminals

<nVal.0001> = first  ;
<nVal.0002> = second ;
<nVal.0003> = third  ;
<nVal.0004> = fourth ;
<nVal.0005> = fifth  ;
<nVal.0006> = sixth  ;
<nVal.0007> = seventh;
```

```
<nVal.0008> = eighth ;
<nVal.0009> = ninth  ;
<nVal.0000> = zeroth ;

// teens

<nVal.00011> = eleventh   ;
<nVal.00012> = twelfth    ;
<nVal.00013> = thirteenth ;
<nVal.00014> = fourteenth ;
<nVal.00015> = fifteenth  ;
<nVal.00016> = sixteenth  ;
<nVal.00017> = seventeenth;
<nVal.00018> = eighteenth ;
<nVal.00019> = nineteenth ;

// ten

<nVal.00010> = tenth ;
<nVal.00020> = twentieth  ;
<nVal.00030> = thirtieth  ;
<nVal.00040> = fortieth   ;
<nVal.00050> = fiftyieth  ;
<nVal.00060> = sixtieth   ;
<nVal.00070> = seventieth ;
<nVal.00080> = eightith   ;
<nVal.00090> = ninetieth  ;

// *************************
// ***** 1-4 DIGITS ********
// *************************

<@4digits> =
   <@2digits> <@2digits> |
   <@4digits_21_100:OpMultiply>;

<@4digits_21_100> =
   <@numrange_01-99> <nVal.100>  ; // e.g., twenty one hundred

<@3digits_nlz> =
   <@2digits_nlz> <@1digit> |
   <@1digit_nlz> <@2digits> |
   <@hundreds>;

<@3digits> =
   <@2digits> <@1digit> |
   <@1digit> <@2digits> |
   <@hundreds>;

<@2digits_nlz> =
   <@1digit_nlz> <@1digit> |
   <nVal.10> |
   <@numrange_11-19>   |
   <@numrange_20-99>   ;

<@2digits> =
   <@1digit> <@1digit> |
   <nVal.10> |
   <@numrange_11-19>   |
   <@numrange_20-99>   ;

<@1digit_nlz> =
   <@numrange_1-9>;

<@1digit> =
   <@numrange_0-9>;

// *************************
// ***** NUMBER RANGES *****
// *************************

<@numrange_00-9999> =
   <nVal.0>             |
   <@numrange_01-9999>  ;

// 1000s

<@numrange_01-9999> =
   <@numrange_01-999>   |
   <@numrange_1000-9999> ;

<@numrange_1000-9999> =
   <@numrange_1000-9999_aux:OpAdd>;

<@numrange_1000-9999_aux> =
   <@few_thousands> [[<@fillers>] <@numrange_01-999>] ;

<@few_thousands> =
   [<filler_one:nVal.0>] <@thousand> |
   <@few_thousands_aux:OpMultiply> ;

<@few_thousands_aux> =
   <@numrange_1-9> <@thousand> ;

<@thousands> =
   [<filler_one:nVal.0>] <@thousand> |
   <@thousands_aux:OpMultiply> ;

<@thousands_aux> =
   <@numrange_01-9999> <@thousand> ;

// 100-999
```

```
<@numrange_01-999> =
   <@numrange_01-99>   |
   <@numrange_100-999> ;

<@numrange_100-999> =
   <@numrange_100-999_aux:OpAdd>;

<@numrange_100-999_aux> =
   <@hundreds> [[<@fillers>] <@numrange_01-99>] ;

<@hundreds> =
   <@hundred> |
   <@hundreds_aux:OpMultiply> ;

<@hundreds_aux> =
   <@numrange_1-9> <@hundred> ;

// 10s

<@numrange_00-99> =
   <nVal.0> |
   <@numrange_01-99>;

<@numrange_01-99> =
   <@numrange_01-10> |
   <@numrange_11-19> |
   <@numrange_20-99> ;

<@numrange_20-99> =
   <@tens> |
   <@numrange_20-99_aux:OpAdd>;

<@numrange_20-99_aux> =
   <@tens> <@numrange_1-9> ;

<@numrange_01-12> =  // useful for months
   <@numrange_00-10> |
   <nVal.11>         |
   <nVal.12>         ;

<@numrange_11-19> =
   <nVal.11> |
   <nVal.12> |
   <nVal.13> |
   <nVal.14> |
   <nVal.15> |
   <nVal.16> |
   <nVal.17> |
   <nVal.18> |
   <nVal.19> ;

<@numrange_00-10> =
   <nVal.0>       |
   <@numrange_01-10> ;

<@numrange_01-10> =
   <@numrange_01-09> |
   <nVal.10>     ;

<@numrange_01-09> =
   [<nVal.0>] <@numrange_1-9>;

<@numrange_01-02> =
   [<nVal.0>] <@numrange_1-2>;

<@numrange_0-9> =
   <@numrange_1-9> |
   <nVal.0>        ;

<@numrange_1-9> =
   <@numrange_1-2> |
   <nVal.3>     |
   <nVal.4>     |
   <nVal.5>     |
   <nVal.6>     |
   <nVal.7>     |
   <nVal.8>     |
   <nVal.9>     ;

<@numrange_1-2> =
   <nVal.1> |
   <nVal.2> ;

<@numrange_0> =
   <nVal.0>;

// ********************
// ***** TERMINALS *****
// ********************

// ones

<nVal.1> = one;
<nVal.2> = two;
<nVal.3> = three;
<nVal.4> = four;
<nVal.5> = five;
<nVal.6> = six;
<nVal.7> = seven;
<nVal.8> = eight;
<nVal.9> = nine;
```

```
<nVal.0> = zero | oh | o;

// useful if you don’t want to mark up (e.g. in lists of Zip codes)
<@nVal.1> = one;
<@nVal.2> = two;
<@nVal.3> = three;
<@nVal.4> = four;
<@nVal.5> = five;
<@nVal.6> = six;
<@nVal.7> = seven;
<@nVal.8> = eight;
<@nVal.9> = nine;
<@nVal.0> = zero | oh | o;

// ten

<nVal.10> = ten;

// teens

<nVal.11> = eleven;
<nVal.12> = twelve;
<nVal.13> = thirteen;
<nVal.14> = fourteen;
<nVal.15> = fifteen;
<nVal.16> = sixteen;
<nVal.17> = seventeen;
<nVal.18> = eighteen;
<nVal.19> = nineteen;

// tens

<@tens> =
   <nVal.20> |
   <nVal.30> |
   <nVal.40> |
   <nVal.50> |
   <nVal.60> |
   <nVal.70> |
   <nVal.80> |
   <nVal.90> ;

<nVal.20> = twenty;
<nVal.30> = thirty;
<nVal.40> = forty;
<nVal.50> = fifty;
<nVal.60> = sixty;
<nVal.70> = seventy;
<nVal.80> = eighty;
<nVal.90> = ninety;

// > 100

<@hundred> =
   <nVal.100>;

<@thousand> =
   <nVal.1000>;

<nVal.100>  = hundred;
<nVal.1000> = thousand;

// fillers
<@fillers> = and;

// ordinals

<@oVal.1> = first  ;
```

```
<@oVal.2> = second ;
<@oVal.3> = third  ;
<@oVal.4> = fourth ;
<@oVal.5> = fifth  ;
<@oVal.6> = sixth  ;
<@oVal.7> = seventh;
<@oVal.8> = eighth ;
<@oVal.9> = ninth  ;
<@oVal.0> = zeroth ;

<oVal.10> = tenth  ;

// teens

<oVal.11> = eleventh   ;
<oVal.12> = twelfth    ;
<oVal.13> = thirteenth ;
<oVal.14> = fourteenth ;
<oVal.15> = fifteenth  ;
<oVal.16> = sixteenth  ;
<oVal.17> = seventeenth;
<oVal.18> = eighteenth ;
<oVal.19> = nineteenth ;

<oVal.20> = twentieth  ;
<oVal.30> = thirtieth  ;
<oVal.40> = fortieth   ;
<oVal.50> = fiftyieth  ;
<oVal.60> = sixtieth   ;
<oVal.70> = seventieth ;
<oVal.80> = eightith   ;
<oVal.90> = ninetieth  ;

<filler_one> =
   one;         // e.g., one thousand, one hundred
```

C.2.2 Extended Grammars

DATE-specific part

Only the main rule was changed in the extended version:

```
<cls_DATE> =
   [<day-of-week>] <month> [the] <day-of-month> [<year>] |
   [<day-of-week>] [the] <day-of-month> of <month> [<year>] |
   <@month_excluding_may:month> [ [of] <year>] |
   (in | from | by | my | our | of | for | the | this)
                              <@month_of_may:month> [ [of] <year>] |
   (in | from | by | my | our | of | for) <@year_full:year> ;
```

ITEM_AMOUNT-specific part

This is the only grammar where, instead of context integration, we found it useful to remove one of the rule alternatives, so that the main rule here now is:

```
<cls_CURRENCY> =
   <@dollars> [ <@simple-cents> | ( [and] <@cents> ) ]
   | <@one-dollar:dollar-amount> <@simple-cents>
// | <@cents>
   | <@simple-dollars> <@simple-cents>
;
```

PHONE_NUMBER-specific part

The NE-specific rules of this named entity have been changed to:

```
<START> = <cls_PHONENUMBER> ;

<cls_PHONENUMBER> =
   ( <@my_phone_number_is> [ [<@my_area_code>] <areacode> ] <phoneno> )
   |
   ( [<@my_phone_number_is>] <@my_area_code> <areacode> <phoneno> )
   |
   ( <@my_phone_number_is> <phoneno> [and] <@my_area_code> <areacode> )
   ;

<phoneno> =
   <@3digits_nlz> <@4digits> ;

<areacode> =
   [ <nVal.1> ] <@3digits_nlz> ;


/* ******************************
   local context
   ****************************** */
<@my_phone_number_is> =
     ( [my] (phone | telephone) [number] [is] )
      |
     ( ( my | our | the ) number [is] ) ;

<@my_area_code> = [area] code | area [code] ;
```

Appendix D

Generated Acoustic Morphemes

Here, we present more examples of acoustic morphemes automatically generated by our experiments from Section 6.6. Phonetic alphabet ARPABET is employed:

```
AM #0
 bos b ih l eos
 bos b ih l ih ng eos
 ih s b ih l ih ng eos
AM #1
 ax b eh t er
AM #10
 ey t iy a n d t iy w er l d n ih t
AM #100
 s s er v ih s eos
AM #101
 s ah m th ih ng ax b aw t m ay
AM #102
 dh ey w er m ey d t uw v
AM #103
 w ih th ey
AM #104
 s iy hh aw m ah ch
AM #105
 ax r ey n jh m ax n
 ax r ey n jh m ax n s eos
```

```
AM #106
 a n s ax l m ay s er v ih s eos
 bos k a n s ax l s er v ih s eos
 k a n s ax l m ay ax k aw n t eos
 k a n s ax l m ay s er v ih s eos
 t uw k a n s ax l m ay s er v ih s eos
AM #107
 b ih l ih ng ih n f er m ey sh ax n eos
 bos b ih l ih ng f er m ey sh ax n eos
 bos b ih l ih ng ih n f er m ey sh ax n eos
 m ay b ih l ih ng ih n f er m ey sh ax n eos
AM #108
 n ow hh aw m ah ch m ay b ih l
AM #109
 s er v ih s d ih s k ax n eh k t ih d eos
AM #11
 ch ey n jh ah v ax d r eh s t eos
 ch ey n jh m ay a d r eh s
 ch ey n jh m ay a d r eh s t eos
AM #110
 ih ng ih n k w ay er iy eos
AM #111
 m ay b ih l ih ng a d r eh s t eos
AM #112
 bos ay d l ay k t uw s p iy k t uw k ah s t ax m er s er v ih s eos
AM #113
 ch eh k aa n ax
AM #114
 bos ay n iy d t uw s p iy k t uw
 bos ay n iy d t uw s p iy k t uw ax
 bos ay n iy d t uw s p iy k t uw ah
 bos ay n iy d t uw s p iy k t uw ey
 bos ay n iy d t uw s p iy k w ih th
 bos ay n iy d t uw t ao k t uw
 bos ay w aa n ax s p iy k t uw
 bos ay w aa n t uw s p iy k w ih th ax
 bos n iy d t uw s p iy k w ih th
 n iy d t uw s p iy k t uw
AM #115
 bos ax k aw n t ih n f er m ey sh ax n eos
 bos ay d l ay k t uw m ih n f er m ey sh ax n
AM #116
 bos w aa n ax n ow
 bos w aa n ax n ow w ah t
AM #117
 t ih l ih ng k aa r d eos
AM #118
 w iy m ey d eos
AM #119
 ax k ah s t ax m er r eh p r ih z eh n t ax t ih v p l iy z eos
 ax l ay v r eh p r ih z eh n t ax t ih v eos
 ax r eh p r ih z eh n t ax t ih v eos
 ax r eh p r ih z eh n t ax t ih v p l iy z eos
 ax s er v ih s r eh p r ih z eh n t ax t ih v p l iy z eos
 ah ax r eh p r ih z eh n t ax t ih v eos
 th p r eh p r ih z eh n t ax t ih v eos
 bos r eh p r ih z eh n t ax t ih v p l iy z eos
 r eh p r ih z eh n ih v p l iy z eos
 r eh p r ih z eh n t ax t ih v p l iy z eos
 s er v ih s r eh p r ih z eh n t ax t ih v p l iy z eos
AM #12
 bos ay n iy d k ah s t ax m er s er v ih s eos
 bos k ah s t ax m er s er v ih s eos
 bos k ah s t ax m er s er v ih s p l iy z eos
 bos l ao ng d ih s t ax n s er v ih s eos
AM #120
 bos ay d l ay k t uw s p iy k t uw ax k ah s t ax m er
 bos ay w uh d l ay k t uw s p iy k t uw ax k ah s t ax m er
```

```
AM #121
 bos ay d l ay k t uw s p iy k t uw ax p er s ax n p l iy z eos
 bos ay d l ay k t uw s p iy k t uw a n aa p ax r ey t er p l iy z eos
 bos ay n iy d t uw s p iy k t uw a n aa p ax r ey t er p l iy z eos
 bos ay w uh d l ay k t uw s p iy k t uw a n aa p ax r ey t er eos
AM #122
 p ey m ax n ax r ey n jh
AM #123
 ch aa r jh d ah ^
AM #124
 b ay k r eh d ih t k aa r d eos
 k r eh d ih t k aa r d eos
 k r eh d ih t k aa r d p l iy z eos
 t m ay k r eh d ih t k aa r d eos
 w ih th ax k r eh d ih t k aa r d eos
 w ih th m ay k r eh d ih t k aa r d eos
AM #125
 er m ax n
AM #126
 d ih s k ah s m ay b ih l eos
AM #127
 ao n m ay b ih l p
AM #128
 ax l ay v eos
AM #129
 dh iy ax m aw n t ah
 dh iy ax m aw t ah v m ay
AM #13
 ch aa r jh ao n m ay b ih l eos
 ch aa r jh ih z aa n m ay b ih l eos
 ch aa r jh aa n m ay b ih l eos
 bos ay hh a v ch aa r jh ih z ao n m ay b ih l
AM #130
 b ih l p ey m ax n
```

```
AM #131
 bos ay w aa n ax p ey ax
AM #132
 d ih s k ax n eh k t
 d ih s k ax n eh k t ih t eos
AM #133
 k a n s ax l m ay ey t iy a n d t iy l ao ng d ih s t ax n s er v
                                  ih s eos
AM #135
 bos hh aw m ah ch ax n
 bos hh aw m ah ch ih t
AM #136
 k a n s ax l m ay l ao ng d ih s t ax n s s er v ih s eos
AM #137
 t uw ax n uw a d r eh s
AM #138
 bos ch eh k ax f ow n n ah m b er
AM #139
 s p iy k t uw ax p er s ax n eos
 s p iy k t uw ey hh y uw m ax n eos
 s p iy k w ih th ax p er s ax n eos
AM #14
 dh ax k ey b ax l
 k ey b ax l
AM #140
 bos ay d l ay k t uw d ih s k ah s m ay b ih l eos
AM #141
 bos b a l
AM #142
 bos ay d l ay k t uw s p iy k t uw ax k ah s t ax m er r eh p r
                            ih z eh n t ax t ih v eos
 bos ay d l ay k t uw s p iy k t uw ax k ah s t ax m er s er v ih
                              s r eh p r ih z eh n t ax t ih v eos
 bos ay d l ay k t uw s p iy k t uw ax r eh p r ih z eh n t ax t
```

```
                                    ih v eos
 bos ay d l ay k t uw s p iy k t uw ax r eh p r ih z eh n t ax t
                                    ih v p l iy z eos
 bos ay n iy d t uw s p iy k t uw ax k ah s t ax m er s er v ih s
                                    r eh p r ih z eh n t ax t ih v eos
 bos ay n iy d t uw s p iy k t uw ax r eh p r ih z eh n t ax t
                                    ih v eos
 bos ay n iy d t uw s p iy k t uw ey r eh p r ih z eh n t ax t ih
                                    v eos
 bos ay n iy d t uw t ao k t uw ax r eh p r ih z eh n t ax t ih
                                    v eos
 bos ay w aa n ax s p iy k t uw ax r eh p r ih z eh n t ax t ih
                                    v eos
 bos ay w aa n ax t ao k t uw ax r eh p r ih z eh n t ax t ih v eos
 bos ay w uh d l ay k t uw s p iy k t uw ax k ah s t ax m er s er
                           v ih s r eh p r ih z eh n t ax t ih v eos
 bos ay w uh d l ay k t uw s p iy k t uw ax r eh p r ih z eh n t
                                 ax t ih v eos
AM #143
 ch ey n jh m ay s er v ih s eos
AM #144
 b ih l b ay k r eh d ih t k aa r d eos
 p ey m ay b ih l b ay k r eh d ih t k aa r d eos
AM #145
 ax p r aa b l ax m w ih th m ay b ih l eos
 bos ah ay hh a v ax p r aa b m w ih th
 bos ay hh a v ax p r aa b l ax m w ih th m ay b ih l eos
 bos hh a v ax p r aa b l ax m w ih th m ay b ih l eos
 bos hh a v ax p r aa b m w ih th m ay b ih l eos
 p r aa b l ax m w ih th m ay b ih l eos
 p r aa b m w ih th m ay b ih l eos
AM #146
 d ih n ax f ay z s ah m
AM #147
```

```
 p ey m ih t w ah z
AM #148
 bos p ey m ax n t
AM #149
 ey t s er v ih s eos
AM #15
 dh ax n ah m b er eos
AM #150
 bos ay d l ay k t uw d ih s k ax n t ih n y uw s er v ih s eos
 bos ay w uh d l ay k t uw d ih s k ax n eh k t m ay s er v ih s eos
AM #151
 b ih l ih ng k w eh sh ch ax n eos
 bos ay hh a v ax b ih l ih ng k w eh sh ch ax n eos
 bos b ih l ih ng k w eh sh ch ax n eos
 bos b ih l ih ng k w eh sh ch ax n s eos
 bos b ih l ih ng k w eh s ch ax n eos
 bos hh a v ax b ih l ih ng k w eh sh ch ax n eos
 bos hh a v ax b ih l ih ng k w eh s ch ax n eos
```