Institut für Neuroinformatik
Universität Ulm
Prof. Dr. Günther Palm

Word Recognition using Hidden Markov Models and Neural Associative Memories

Dissertation zur Erlangung des Doktorgrades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Ingenieurwissenschaften
und Informatik der Universität Ulm

vorgelegt von

Zöhre Kara Kayikci

aus Istanbul, Türkei

2008

Amtierender Dekan	: Prof. Dr. Michael Weber
1. Gutachter	: Prof. Dr. Günther Palm
2. Gutachter	: Prof. Dr. Dr.-Ing. Wolfgang Minker
Tag der mündlichen Prüfung	: 24.02.2009

Acknowledgements

First of all I would like to thank my PhD supervisor, Prof. Dr. Günther Palm, head of the "Institute of Neural Information Processing" at the University of Ulm, for his support, suggestions and his great interest.

I would also like to express my gratitude to my second PhD supervisor, Prof. Dr. Dr.-Ing. Wolfgang Minker, head of the "Institute of Information Technology" at the University of Ulm, for his interest in this thesis and his valuable advice.

I thank my mentor Dr. Friedhelm Schwenker for his reading and his helpful recommendations. My thanks go also to Dr. Muhamed Qubbati and David Bouchain for a critical reading and for their useful suggestions.

I also have to thank the Graduate School, University of Ulm whose doctoral scholarship financed this thesis. Further thanks go to my colleagues at the Institute of Neural Information Processing of the University of Ulm for their assistance.

Kym Nguyen deserves special thanks for her support in proof-reading during the last phase of my work.

Thanks also go to my mentor Dr. Zafer Kücük from the Karadeniz Technical University, Turkey, for his support and encouragement.

Finally, I would like to thank my family and my husband Aziz for their understanding, their support, encouragement, tolerance and for their patience.

Ulm, November 2008 Zöhre Kara Kayikci

Zusammenfassung

In dieser Arbeit wird ein neuer hybrider Ansatz für die automatische Spracherkennung vorgestellt. Dieser Ansatz basiert auf Hidden-Markov-Modellen (HMM) auf dem Subword-Unit-Level und neuronalen Assoziativspeichern auf der Wort- und Sprachebene. Der Schwerpunkt der Arbeit ist, ein flexibles und robustes Spracherkennungssystem unter realen Bedingungen zu entwickeln und die Erkennungsleistung entsprechend zu verbessern. Der entwickelte Ansatz besteht aus zwei Teilen: der Subword-Unit-Erkennung, die auf HMM basiert, und der Worterkennung, die mit neuronalen Assoziativspeichern realisiert ist. Die Worterkennung besteht aus Einzelworterkennungs- und Sprachmodell-Netzwerken.

Das System ist Teil von einem Sprachverarbeitungssystem, das in einen autonomen mobilen Roboter eingebettet ist. Für eine gegebene Sprachäußerung erkennt das hybride System die Wörter und leitet sie an das Satzverständnis-Modul weiter, wobei es zufällig generierte Wortrepräsentationen verwendet. Um Subword-Unit- und Word-Strings in den neuronalen Assoziativspeichern zu verarbeiten, werden binäre spärliche neuronale Repräsentationen benutzt. Sie sind auch nützlich, um die Mehrdeutigkeiten auf dem Subword-Unit-Level und Word-Level zu repräsentieren.

Im Rahmen dieser Dissertation werden verschiedene Aspekte des entwickelten hybriden Systems untersucht. Diese Aspekte beinhalten die Repräsentation und die Behandlung der Mehrdeutigkeiten auf verschiedenen Ebenen und die inkrementelle Erweiterung des Wörterbuchs um neue Wörter. Wegen ggf. falsch ausgesprochener Wörter, Homophone, der Wortgrenzen-Mehrdeutigkeiten, des Hintergrundgeräusches oder ungenügend vieler Trainings-Daten ergeben sich zwei Typen von Mehrdeutigkeiten während der HMM Vorverarbeitung. Diese können als die Subword-Units, die falsch oder nicht von HMM erkannt werden können, definiert werden. Die Fehlertoleranz der neuronalen Assoziativspeicher ermöglicht es dem System, die Mehrdeutigkeit auf dem Subword-Unit-Level zu lösen. Wenn das System die Mehrdeutigkeit nicht lösen kann, repräsentiert es sie auf der Wortebene, indem es eine Superposition von allen alternativen Wörtern für die Subword-Unit-Sequenz generiert. Um diese Mehrdeutigkeit auf der Wortebene zu lösen, wurde das Einzelworterkennungsnetzwerk um ein anderes Netzwerk aus

neuronalen Assoziativspeichern (das Sprachmodellnetzwerk) erweitert. Dieses verwendet die zusätzliche a-priori-Information über die Wortsequenzen, die erkannt werden sollen. Für die kontinuierliche Spracherkennung mit großem Wörterbuch, steigert das Sprachmodellnetzwerk die Erkennungsperformanz des hybriden Systems erheblich.

Ein wichtiger Aspekt des Hybrid-Systems, das im Rahmen dieser Arbeit untersucht wird, ist das inkrementelle Hinzufügen neuer Wörter im Wörterbuch. Auf der HMM-Ebene des präsentierten Systems wurden zwei verschiedene Ansätze für die Verbesserung der Lernperformanz eingesetzt.

Das hybride Spracherkennungssystem wird erfolgreich bei verschiedenen Erkennungsaufgaben angewendet. Im Vergleich zu anderen auf HMM basierenden Spracherkennungssystemen sind die Erkennungsergebnisse konkurrenzfähig.

Summary

In this thesis a novel hybrid approach to automatic speech recognition (ASR) has been proposed. This hybrid system is based on hidden Markov models (HMMs) on the subword-unit level and neural associative memories (NAMs) on the word and language levels. The focus of the work is to develop a flexible and robust speech recognition system against real-world environments and to augment the recognition performance. The developed hybrid system consists of two parts: HMM-based subword-unit recognition and NAM based word recognition, which is also composed of single word recognition network and language model network.

The developed hybrid system is also a part of a language processing system embedded in a mobil robot. For a given speech utterance the developed hybrid system recognizes words and forwards them to the NAM based sentence understanding module in the language processing system using randomly generated binary neural word representations. In order to process subword-unit and word strings in the NAM based recognition networks, binary sparse neural representations were utilized, which were also useful to represent ambiguities on the subword-unit and word levels.

Within the scope of this thesis different features of the developed hybrid system were investigated. These features include representation and handling of ambiguities on different levels, and incremental extension of task vocabulary with novel words. Due to the pronunciation ambiguity, homophones, word boundary ambiguity, background noise or insufficient training data two types of ambiguities rose during the HMM preprocessing. These can be defined as the subword-units that are wrongly recognized or can not be recognized by HMMs. The fault tolerance ability of NAMs usually enables the hybrid system to solve the ambiguity on the subword-unit and words levels. If the hybrid system can not solve the ambiguity then it represents the ambiguity on the word level generating a superposition of all alternative words for the subword-unit sequence. In order to solve the ambiguity on the word level, the single word recognition network was extended by adding another network of NAMs (language model network) utilizing the additional priori information on the word sequences to be recognized. For large vocabulary continuous speech recognition, the NAM based language model

network considerably increases the recognition performance of the hybrid system.

An important feature of the hybrid system examined in the context of this work is the incremental adding of novel words to the task vocabulary during runtime. On the HMM level of the presented hybrid system two slightly different approaches for improving the learning performance of HMMs have been proposed and deployed.

The proposed hybrid speech recognition system has been successfully applied to various recognition tasks. Compared to other HMM-based speech recognition systems in the literature, competitive recognition results were achieved.

Contents

List of Tables

List of Figures

Chapter 1

Overview

This chapter gives a motivation of the thesis at hand. Research goals are formulated and the solution approach is outlined. At the end of this chapter the structure of this work is summarized.

1.1 Motivation

Speech recognition is the process of automatically extracting and determining linguistic information conveyed by a speech wave using computers. Linguistic information is also called phonetic information such as phonemes (the minimal unit of sound that has a semantic content) or words. In typical speech recognition systems, the input speech is compared to stored units of phonemes or words, and the most likely sequence of units is selected as an output sequence of phonemes or words of input speech.

With the progress in the development of new algorithms and improved modeling techniques, speech recognition has started to facilitate human-machine interfaces and has evolved from laboratory systems for isolated speech and small vocabularies to commercial applications for continuous speech and large vocabularies. Speech recognition is used in a variety of applications, from computer control, and speaker verification to dictation systems.

Despite the advances achieved throughout the last decades, existing speech recognition technologies are still behind the capabilities of the human brain to understand speech. One popular method dealing with speech recognition is a statistical approach called *hidden Markov models* (HMMs). In particular, recognition systems based on HMMs [80] are effective and allow for good recognition performance under many circumstances, but suffer from some limitations such as robustness to variability from speakers and environmental conditions [96], and, increasing dictionary size.

For example, a word may be uttered differently by the same speaker because of illnesses or emotions. Typically, females sound different from males. As

do children from adults. Also, there is variability due to dialect and foreign accent. The speech produced in noise is different from the speech produced in a silent environment because of the change in speech production.

Starting from the late 1980s, HMMs and artificial neural networks (ANNs) have been combined within a hybrid architecture and a variety of different approaches have been proposed in the literature in order to overcome these limitations [14, 71, 27, 32, 62, 95, 12, 68].

The thesis at hand focuses on the automatic speech recognition system utilising a novel hybrid HMM/neural associative memory (NAM) approach, which is a single layer ANN, different from the approaches in the literature. The proposed speech recognition system is a part of a language processing system for understanding simple command sentences. The hybrid system processes the speech waveform hierarchically. The speech waveform is first transformed into speech vectors (e.g. mel frequency cepstral coefficients) and HMMs use these speech vectors to recognize subword-units. The outputs of HMMs (subword-units) are then forwarded to NAMs to extract words from the stream of subword-units. Afterwards, in the language processing system, the stream of words is processed by an NAM-based language understanding module responsible for extracting the semantics from the stream of words with respect to a set of grammatical rules [66, 63].

The reason for extending HMM-based systems with NAMs is to improve ASR performance by benefiting from the capabilities of NAMs, such as pattern matching, fault-tolerance and learning. NAMs are very useful in real-world applications that deal with noisy, often incomplete data. Because of the difficulties mentioned above, for a given speech utterance, the outputs of HMMs can contain ambiguities, e.g., missing or spurious subword-units. NAMs might be suitable for solving such ambiguities that occur in the HMM output. In the presented approach the binary *Willshaw model* [100] is used for the NAMs because it is simple to implement. It is highly efficent and useful in technical applications due to its rapid and fault-tolerant retrieval of stored patterns.

Within the scope of this research we also address the expansion of the task vocabulary with novel words. In real-world scenarios it is likely that the hybrid speech recognition systems have to learn previously unknown-words. Thus the capability of learning new words is very useful.

One well-established application for the proposed hybrid speech recognition system is the language understanding system embedded in a robot. For robots it is essential to be able to recognize the spoken commands and understand them to fulfill the tasks corresponding to the spoken commands.

1.2 Research Goals

Speech recognition is difficult due to speaker variability, pronunciation variability, large set of acceptable words and environmental conditions. These speech characteristics greatly reduce the performance. Acoustic data are strongly speaker dependent and this dependence makes the ASR problem more difficult. The pronunciation variability means that the same word can be uttered in different ways, i.e. with different phonemes. For most ASR applications the number of acceptable words is large. Therefore, the search space of the recognition network is large and complex. The last of these characteristics that reduces recognition accuracy is noise in the acoustic environment. All these characteristics are big challenges for designing a flexible and robust ASR system.

The research presented in this thesis was basically motivated by the aim of developing a speech recognition system that is flexible for speaker and pronunciation variabilities and also robust against environmental noise. Therefore, the HMM-based speech recognition system is combined with NAM-based recognition networks in the proposed approach, where HMMs are used for subword-unit (e.g. phonemes, demi-syllables or syllables) recognition and NAMs are utilized for word recognition. A word recognition architecture is developed, which is composed of various interconnected NAMs.

The proposed speech recognition system is also a part of a language processing system for understanding simple command sentences. It provides recognized words to the NAM-based sentence understanding module in the language processing system using binary neural representations.

An important focus in the presented approach is the handling of ambiguities and their representation for further processing in language recognition and sentence understanding modules. During subword-unit recognition, for a given speech utterance, it is possible that missing or (and) (additional) spurious subword-units happen in the HMM output sequence. The task of the NAM-based recognition networks is to retrieve the correct words from the HMM output stream. If the address pattern generated from the HMM output transcription does not strongly address one unique word pattern in NAMs, then the ambiguity in the address pattern is handled by representing a superposition of all words in question. The superposition of words can be later handled on upper levels, such as language model or sentence levels to resolve the ambiguity on the word level using additional inputs,such as a priori information on the word sequence to be recognized, contextual information or syntactical information.

Another focus within the scope of this thesis is the application of the presented approach to large vocabulary speech recognition systems, where the number of acceptable words is large. For this kind of systems language

model is a very important in order to increase recognition performance. Because of this reason the proposed approach is then extended with an NAM-based language model network.

The last focus in this work is on learning novel words during runtime. The neural network architecture gives an additional benefit if new word representations should be added. Namely the underlying HMMs on the subword-unit level are reused with minimal or no retraining. If a new word representation is learned, the NAMs learn which HMM output transcription belongs to which word, and they can learn this associative relationship even if the HMM output transcription does not yield the subword-units that a linguist might expect for a "correct" transcription of the novel word. This eliminates the requirement of changing the HMMs, which saves a lot of processing power.

1.3 Overview of the Thesis

This thesis is divided into four parts. The first part (chapter 2) describes an introduction of the relevant fields, like ASR, subword-unit-level sentence representation, and the theory of associative memories. The second part (chapters 3 and 4) gives theoretical foundations for this work: fundamentals of statistical speech recognition and the classical theory of binary Willshaw associative memory. In the third part (chapters 5, 6 and 7) the new hybrid speech recognition approach developed within the scope of this thesis, its extended form and the learning aspect of the proposed hybrid system are defined. Finally, the last part (chapter 9) shows different applications of the hybrid system and the results of the evaluation of the proposed approach.

A typical chapter is composed of an introduction part, which gives information on the related work in the literature, a main part, which is concerned with the relevant or newly developed methods, and a discussion part, which summarizes the whole chapter.

Chapter 2 is an introduction to ASR, the different types of subword-units used to represent sentences, and the theory of associative memory. It also briefly addresses the relevant works in the literature.

Chapter 3 reviews the statistical speech recognition. The fundamentals of speech recognition are given [81, 80].

Chapter 4 examines the binary Willshaw associative memory [100, 73, 76]. The storage and retrieval processes are explained and the aspect of fault tolerance is analyzed.

Chapter 5 presents the novel hybrid HMM/NAM speech recognition approach. The hybrid approach is composed of two parts: the HMM-based subword-unit recognition and the NAM-based single word recognition network. These

parts are explained individually. The aspect of representing ambiguity in the single word recognition network is analyzed.

Chapter 6 concerns the extension of the hybrid system with an NAM-based language model network. The aspect of handling ambiguity in the language model network is given.

Chapter 7 investigates the learning of novel words during runtime.

Chapter 8 introduces the speech corpora which are used during application and evaluation of the hybrid system.

Chapter 9 first gives an overview of the methods that are used to measure the performance of the proposed hybrid speech recognition system. Then it shows different applications of the hybrid system and the results of the evaluation of the proposed approach. The proposed hybrid system is also compared to the HMM-based research in the literature in terms of the recognition performance (word error rate (WER)).

Chapter 10 gives the main contributions of this work.

Chapter 11 discusses the main results of this work.

Appendix A gives the sentences involved in the MirrorBot speech corpus.

Appendix B reviews the basic information theory.

Chapter 2

Introduction

Automatic speech recognition (ASR) [80, 46, 21] is the process by which a computer converts spoken words into written words. One major application area of ASR is human-computer interaction. While many tasks are more efficiently solved with visual or pointing interfaces, speech has the potential to be a user-friendly interface than the keyboard, for tasks where natural language communication is useful, or for which keyboards are not appropriate (e.g. hands-busy applications).

Speech recognition tasks can be classified in terms of a set of directions. A first direction of classification is the vocabulary size. Speech recognition is easier if the number of distinct words to be recognized, such as the *digit* tasks, are small. On the other hand, tasks with large vocabularies of roughly $20,000$ to $60,000$ words are much more difficult.

Another direction in speech recognition is how fluent, natural, or conversional the speech is. *Isolated word* recognition, where there is a small pause between words, is much easier than *continuous speech* recognition, where words run into each other. On the other hand, while reading out loud in *read speech* (e.g. the dictation task) is relatively easy, recognizing the speech of two humans talking to each other, *conversational speech*, is much harder. When humans talk to machines, they talk more slowly and more clearly.

A third direction is speaker and accent. A *speaker dependent* system is a system where the speech patterns are adapted to a single speaker. Speaker independent systems are able to recognize speech from people whose speech it has never been exposed to before and it is harder than speaker dependent recognition systems.

A final direction is channel and noise. Commercial dictation systems, and much of the laboratory research in speech recognition, is done with high quality, head mounted microphones. Head mounted microphones eliminate the distortion that occurs in a table microphone as the speaker's head moves around. Noise of any kind also makes recognition harder. Thus, recognizing

a speaker dictating in a quiet office is much easier than recognizing a speaker dictating in a noisy car on the highway with the window open.

In the last two decades, speech recognition and understanding technology have undergone major changes. ASR has evolved from laboratory systems for isolated speech with small vocabularies to systems for continuous speech from different speakers with large vocabularies.

State-of-the-art speech recognition systems are usually based on the use of hidden Markov models (HMMs) [80]. This means that observable events in the real word (e.g. acoustic observations) are modeled with probability distributions. However, HMMs suffer from several difficulties; concerning increasing dictionary size, different speaking styles of speakers, and weaknesses to environmental conditions [96]. On the other hand, artificial neural networks (ANNs) are well known for one of the most powerful methods of pattern recognition and optimization. In order to overcome these difficulties, ANNs are applied to various ASR applications (e.g. classification of phonemes or words) [18, 19, 28, 84, 35, 98, 99, 60, 105].

An ANN operates by creating connections between many processing elements like neurons. These neurons can be simulated by a digital computer. Each neuron takes many input signals and produces a single output signal that is typically sent as input to other neurons. The neurons can be fully or partially interconnected and are typically organized into different layers. The input layer receives the input and the output layer produces the output. Usually one or more hidden layers are used in between input and output layers. An ANN realizes a mapping between an input space and an output space, which can be specified by learning from a finite set of patterns. Due to their pattern-matching and learning capabilities, ANNs have proved useful in a variety of real-world applications that deal with complex, often incomplete data. The first of these applications were in pattern recognition and speech recognition, in particular.

In spite of their ability to classify short-time acoustic-phonetic units (e.g. phonemes), ANNs were not suitable for ASR, especially with long sequences of acoustic observations which are required to represent words from a dictionary or whole sentences. This is mainly due to the lack of ability to model long-term dependencies in ANNs.

Starting from the late 1980s, a variety of hybrid approaches based on HMMs and ANNs have been introduced to improve the flexibility and performance of speech recognizers. The goal in these hybrid systems for ASR is to take advantage of the properties from both HMMs and ANNs. The hybrid systems, proposed in the literature, were investigated by Trentin and Gori in 1991 [96], and some of these hybrid systems are briefly given below:

Early approaches were based on ANN architectures that attempted to em-

ulate HMMs [14, 71]. These approaches strengthened the idea that ANNs could be effectively used for ASR, but the straight emulation of standard HMMs did not allow for them to overcome these difficulties.

In the early 1990s, HMMs and ANNs were then combined to take advantage of both approaches [27, 62, 67, 71, 8, 105]. Hybrid HMM/ANN systems usually delegate to ANN the computation of emission probabilities. It also inherits from HMM the modeling of words with left-to-right automa and the Viterbi decoding. In some ANN/HMM hybrids, the output of an ANN is sent to an HMM for speech recognition [27, 32, 62, 95]. Each output unit of the ANN is trained to perform a non-parametric estimate of the posterior probability of a context-dependent HMM state given the acoustic observations [12, 68]. This approach had a strong influence over a number of following approaches.

In other approaches [5, 6, 8], the ANN is used as a feature extractor for an HMM, with the goal to transform a raw input sequence into a low dimensional observation sequence. The ANN can also be effectively used as vector quantizers for discrete HMMs which assume that a finite alphabet of input symbols has to be modeled [42, 52].

In this work we will present a new hybrid HMM/NAM approach which is based on HMMs on the elementary phonetic level and NAMs on the higher levels, such as word and language levels. An NAM is the realization of an associative memory in a single layer artificial neural network. The proposed approach possesses both the advantages of HMMs and NAMs, and enables us to develop a flexible and robust hybrid speech recognition system. The rest of this chapter introduces the basics of the relevant fields like automatic speech recognition, subsymbolic (subword-unit) representation of sentences and the theory of associative memory.

2.1 Automatic Speech Recognition

An ASR process may be divided into several steps. First, a speaker speaks some words, generating an audio signal. This signal, which originates in the analog domain, is received by an automatic speech recognition system. Then the analog audio signal is converted to a digital representation for processing by a computer.

In order to characterize the essential information present in the audio signal, an acoustic processor converts the audio signal in a digital format into acoustic features needed to decode the signal into words. For the given acoustic features X, a recognizer will choose the most probable word sequence $\hat{W}$, by searching over possible word sequences W:

$$\hat{W} = \underset{W}{\text{argmax}} p(W)p(X|W). \tag{2.1}$$

The recognizer depends on two components, the language model $P(W)$ and the acoustic model $P(X|W)$.

The language model, $P(W)$, assigns probabilities to sequences of words. To estimate the probability of a sequence of words $W = w_1, w_2, ..., w_n$ Bayes'rule is applied:

$$p(w_1 w_2 .....w_n) = \prod_{i=1}^{n} p(w_i|w_1 ...w_{i-1}). \tag{2.2}$$

This allows to estimate the probability of a sequence by iteratively estimating the probability of the sequences as each new word is introduced in turn. In practice, this is simplified by recognizing only a few words prior to the word in question which are needed to estimate the probability, usually two or three.

The acoustic model, $P(X|W)$, characterizes the relationship between the observed acoustic features and the associated words. The acoustic model is parameterized by a certain number of parameters and acoustic data is used to estimate these parameters. To train an acoustic model, a sufficient amount of transcribed speech is required.

2.2 Subword-Units

For large vocabularies containing tens of thousands of words it is not feasible to provide a separate acoustic model for each word in the vocabulary because a large amount of training data is needed to build a better word-based system. A word based recognizer requires a much larger memory due to storage of every parameter for every state in every word. Instead, each word is decomposed into several sub-word units like phonemes or syllables. Then, the acoustic model of a word is given by the concatenation of the acoustic models which represent the sub-word units. The mapping between the words and the corresponding sequences of sub-word units is given by the so-called pronunciation lexicon (see section 3.2.1).

The pronunciation of a word can be given as a series of symbols that correspond to the individual units of sound that make up a word. The individual units can be phonemes, context-dependent phonemes, demi-syllables or syllables.

A phoneme is the smallest contrastive unit in the sound system of a language. Typically, there are about 50 phonemes for English. In this thesis, phonemes are encoded using TIMIT phonetic symbols [79]. Although the *International*

Phonetic Alphabet (IPA) is very popular, there is a problem with this alphabet: the IPA symbols are difficult to type on computers. It can be done using IPA symbols, but this is very inconvenient. Below are tables 2.1 and 2.2 with the phonetic symbols of the TIMIT.

Table 2.1: *The phonetic symbols of the TIMIT for Vowels.*

TIMIT phonetic symbols	Example
aa	father
ae	bat
ao	bought
aw	bout
ay	bite
ey	bait
iy	beat
ow	boat
oy	boy
uw	boot
ax	about
ix	attribute
eh	bed
ih	bit
uh	book
ah	but
axr	butter
er	bird
ux	suit

Example: The word "bot" is composed of three phonemes "b aa t".

On the other hand, context-independent subword-units are inadequate in representing the spectral and temporal properties of the speech unit in all contexts because the pronunciation of a sound is affected by the preceding and following sounds. Therefore, the set of subword-units is extended to include context-dependent units (diphones and triphones). A context-dependent diphone can be defined as

p_L-p left context diphone
or

p+p_R right context diphone,

where p_L is the phoneme immediately preceding p, p_R is the phoneme immediately following p. In English there are about $1,500$ to $2,000$ diphones. Triphones differ from diphones in that they include a complete central phone

Table 2.2: *The phonetic symbols of the TIMIT for Consonants.*

TIMIT phonetic symbols	Example
hh	hi
hv	ahead
l	lead
el	bottle
r	roof
w	wall
y	yacht
m	mom
em	bottom
n	new
en	button
ng	sing
eng	tossing
f	frank
v	very
th	think
dh	that
s	silly
z	zoom
sh	shelf
zh	azure
p	pool
b	bite
t	tip
d	dog
ch	child
jh	judge
k	clip
g	good

and represent a sequence of three phonemes. Similarly, a set of context-dependent triphones can be defined as

p_L-p+p_R left-right context triphone.

In English there are approximately $10,000$ triphones.

Demi-syllables consist of either the initial consonant cluster and some part of the vowel nucleus, or the remaining part of the vowel nucleus and the final (optional) consonant cluster [85]. For English there are about $2,000$ demi-syllables.

Example: The demi-syllabic representation of the word "bot" consists of two demi-syllables "b_aa aa_t".

The linguistic definition of a syllable is given as a vowel nucleus plus the optional initial and final consonants or consonant clusters. In English there are approximately 10,000 syllables.

Example: The syllabic representation of the word "bot" is "b_aa_t".

Figure 2.1 shows the segmentation of the speech signal containing the sentence "bot show pepper" into various subword-units.

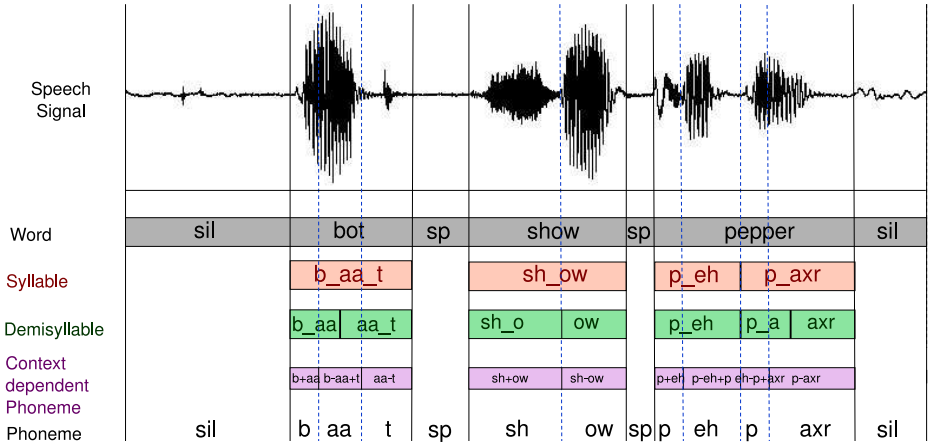

Figure 2.1: The segmentation of the speech signal into various subword-units. "sp" denotes small pause between words.

2.3 Associative Memory

An *associative memory* is a system which stores patterns or associations between patterns (figure 2.2), which are usually represented as vectors of a fixed length. After patterns $u^1, u^2, ..., u^M$ are stored during learning, the stored patterns can be retrieved by addressing the associative memory using the previously stored patterns. In the case that the input pattern is a noisy or incomplete version of the previously stored patterns, the output pattern may be the superposition of the stored patterns related to the noise in the input pattern.

There are two types of associative memory, namely *hetero-associative* and *auto-associative* memory. In *hetero-association* the memory stores associations $x^\mu \mapsto y^\mu$ between two sets, address patterns X and content patterns Y. This is called *pattern mapping*. In this case the stored content patterns can be retrieved by addressing the associative memory by the previously stored address patterns.

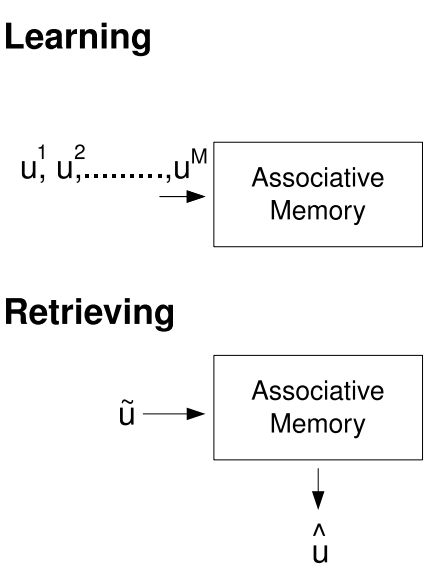

Figure 2.2: An associative memory. During learning M patterns $u^1, u^2, ..., u^M$ are stored. For a retrieval, an address pattern $\tilde{u}$, which may be a noisy version of the previously stored address pattern, is applied to the associative memory. As a retrieval result the associative memory generates an output pattern $\hat{u}$ which should be equal to the previously stored content pattern.

In the case of *autoassociation* the content pattern y^μ is assumed to be equal to the corresponding address pattern x^μ. These memories can be used for *pattern completion*. After learning a noisy or incomplete version of a stored pattern $\hat{x}$ can be completed to a previously stored pattern, which is close to x.

Autoassociation can also be created from *heteroassociation*. For the heteroassociation of pattern pairs $x^\mu \mapsto y^\mu$ where the patterns x^μ and y^μ are binary vectors of lengths m and n, one can concatenate the binary vectors (x^μ, y^μ) of length $m+n$ and the binary patterns can be autoassociatively stored in the resulting $(m+n)\times(m+n)$ matrix (figure 2.3). In the same way, *heteroassociation* is a special case of *autoassociation*. For example, an autoassociative memory matrix can be divided into two quadratic autoassociative sub-matrices A_1 and A_2 and two heteroassociative sub-matrices H and H^T (figure 2.3).

Associative memories can be implemented in various ways [54, 55, 56, 100]. One of these implementations is neural implementation of associative memories [100, 73, 94], which has fault tolerance and the possibility of parallel implementation.

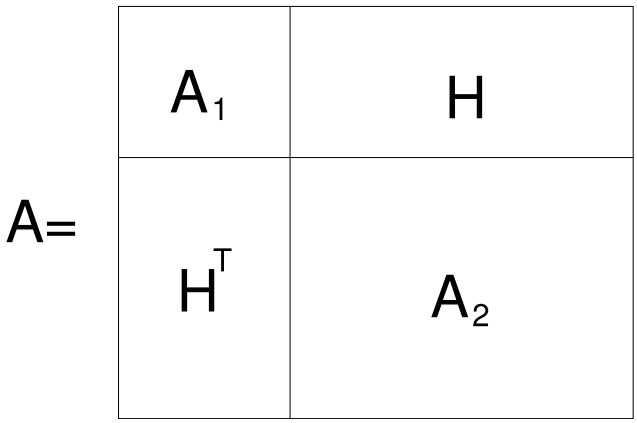

Figure 2.3: Heteroassociation is a special case of autoassociation, and vice versa.

2.3.1 Neural Associative Memory

A neural associative memory (NAM) is the realization of an associative memory in a single layer artifical neural network [75, 56]. Typical representation of an NAM is a matrix W where w_{ij} is the synaptic weight, connecting neuron i of the address population to neuron j of the content population. The patterns are subsets of a neuron population and stored in synaptic connections between two neuron populations namely address and content populations. These correspond to the input patterns x^μ and the output patterns y^μ, respectively.

As shown in figure 2.4, the rows correspond to axons, the columns to dendrites, and the cross-points to modifiable synapses. The information that neurons belong to one pattern is stored in the synapses by the Hebbian learning rule [36]. The output is computed by summing up the products of presynaptic activities and weights of the synapses in each column and comparing the sum against a threshold.

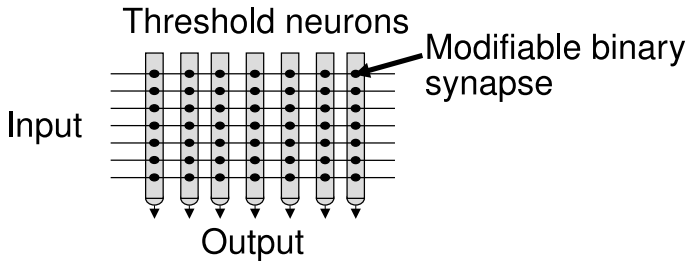

Figure 2.4: The architecture of an NAM.

When one compares neural associative memories with other pattern retrieval algorithms like look-up tables, where each pattern is explicitly represented

in a list or table, the advantage of neural associative memories is that the memory access can be fast if many patterns are stored. In look-up tables, when an address (input) pattern is given it has to be compared to each stored pattern to find the most matching pattern. If n^2 patterns have been stored, the number of steps needed for comparing two patterns is n^2. In contrast, a binary NAM can store a large set of pattern vectors (almost n^2 pattern vectors for a vector length n) in a fault tolerant way and the retrieval process contains only $n\log n$ steps [53]. Even the number of necessary steps would decrease to $\log n$ by a parallel implementation using n processors.

Another advantage of neural associative memories is fault tolerance with respect to variation of the input pattern. This means that an output (content) pattern can be retrieved for a set of input patterns that are closest to the input pattern presented during learning.

2.3.2 The Willshaw Model

The binary **Willshaw model** of neural associative memory uses binary neurons and synapses [100, 73, 94, 76, 17, 89, 93]. The working principle of the binary Willshaw model is given in figure 2.5 for hetero-association. The patterns are hetero-associatively stored in the binary memory matrix $W \in \{0,1\}^{m\times n}$ corresponding to synaptic connections between the address population (for the patterns x^μ) and content population (for the patterns y^μ). The pattern pairs are binary vectors (x^μ, y^μ) where x^μ is the address (input) vector of length m corresponding to a population of m neurons and y^μ is the content (output) vector of length n corresponding to a population of n neurons. The neurons can be active (1) or silent (0). Similarly, each of the binary synaptic connections can be either active or inactive. Usually all the input (or output) pattern vectors contain the same number k of active neurons, called the *pattern activity*:

$$k = \sum_{j=1}^{M} x_j. \tag{2.3}$$

If the pattern activity k is small compared to the vector length m the pattern is called "sparse". A pattern is stored by activating the pattern neurons and Hebbian learning activates all synapses connecting two pattern neurons.

A pattern is stored by activating the pattern neurons and Hebbian learning activates all synapses connecting two pattern neurons. After storing M patterns the memory matrix W of synaptic weights can be expressed as the superposition of outer products of the pattern vectors (see section 4.1). The memory load p_1, a significant parameter of the memory matrix, is defined as the relative number of active synapses. In general p_1 increases with the number of stored patterns M.

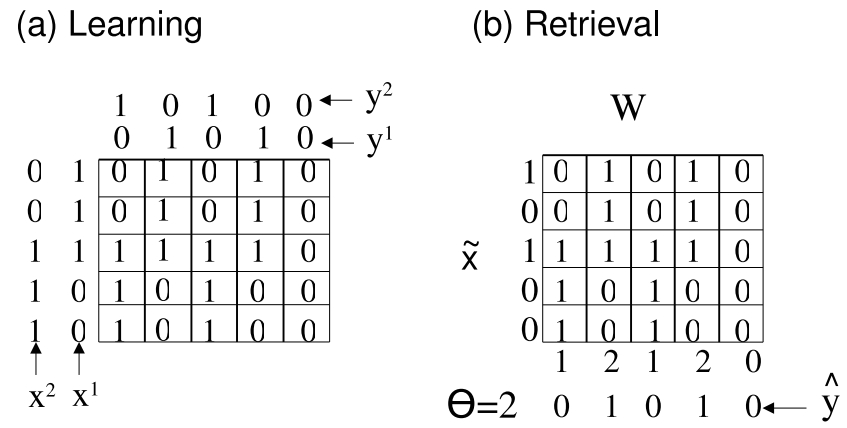

Figure 2.5: Working principle of the binary Willshaw model for hetero-association. During learning phase (left) M binary vector pairs (x^1, y^1),(x^2, y^2),....,(x^M, y^M) are stored sequentially in the binary memory matrix W representing synaptic connections between neurons. Initially all synaptic connections are inactive. After learning of all patterns, each stored vector pair activates the connections corresponding to its outer product. For a retrieval (right) an address pattern $\tilde{x}$ is used for vector matrix multiplication. To obtain the retrieval result $\hat{y}$ a threshold Θ is applied.

Retrieval is performed by activating a number of output neurons corresponding to an address pattern $\tilde{x}$ which can be a noisy version of the stored pattern. The dendritic potentials $y = \tilde{x}W$ of the output neurons are computed by a vector matrix multiplication of the input vector $\tilde{x}$ and the memory matrix W. Then an appropriate threshold Θ is chosen and the output neurons equal to or larger than the threshold are activated. The Willshaw threshold strategy simply chooses the threshold equal to the number of one-entities in the address pattern. Note that if the address pattern is an incomplete or noisy version of the stored pattern, the Willshaw strategy will also activate additional neurons.

The theoretical analysis of the binary Willshaw model was initially carried out in 1969 by David Willshaw [100]. In 1980 it was refined by Günther Palm [73]. It was found that high memory capacities can be obtained if the patterns are sparse.

Chapter 3

Statistical Speech Recognition

Speech recognition technology has been developed significantly since the middle of the last century [20]. Availability of substantial computational rescources and the application of statistical modeling techniques lead to the impressive performance of today's recognizers.

Hidden Markov Models (HMMs), which were introduced in 1975 [3, 45], had significant contributions to the success of the probabilistic approach and are now very popular in speech recognition systems.

The speech recognition problem can be defined as the task of converting a speech signal into a sequence of written words. This problem is solved in three steps: First, the relation between the sound of spoken words and their associated text representation is determined, which is called modeling. Then, given a speech signal, the model is used to hypothesize the words, which is called decoding. And finally, the decided word hypothesis is evaluated to determine how good it is.

Modern speech recognition system works by searching through a large set of text representations to determine the hypothesis which has the highest probability of generating the speech utterance. To do this, acoustic signals first need to be pre-processed to generate a sequence of acoustic feature vectors. Then, given a sequence of acoustic observation vectors $O = o_1, ..., o_T$, a speech recognition system decides on the word sequence $W = w_1, ..., w_N$ that maximizes the *a-posteriori* probability $p(W|O)$. The word sequence which maximizes this posterior probability also minimizes the probability of a sentence error, i.e., at least one word in the recognized sentence is wrong. In mathematical terms, *Bayes' Decision Rule* is given as:

$$\hat{W} = \underset{W \in \Omega}{\operatorname{argmax}} p(W|O) \tag{3.1}$$

$$= \underset{W \in \Omega}{\operatorname{argmax}} \frac{\{p(O|W) \cdot p(W)\}}{p(O)} \tag{3.2}$$

$$= \underset{W \in \Omega}{\operatorname{argmax}} \{p(O|W) \cdot p(W)\}. \tag{3.3}$$

Here, Ω denotes the set of all possible word sequences. The first term $p(O|W)$ in equation 3.3 is generally called the *acoustic model* as it estimates the probability of a sequence of acoustic observations. The second term $p(W)$ is generally referred to as the *language model* since it describes the probability associated with a postulated sequence of words. Note that the denominator $p(O) = \sum_{W'} p(O|W')p(W')$, the probability of acoustic observation, can be neglected because it is the same for all hypotheses Ω and will not have an effect on the decision. Given the acoustic model and language model probabilities, the probabilistic model can be operationalized in a search algorithm so as to compute the maximum word probability for a given acoustic waveform.

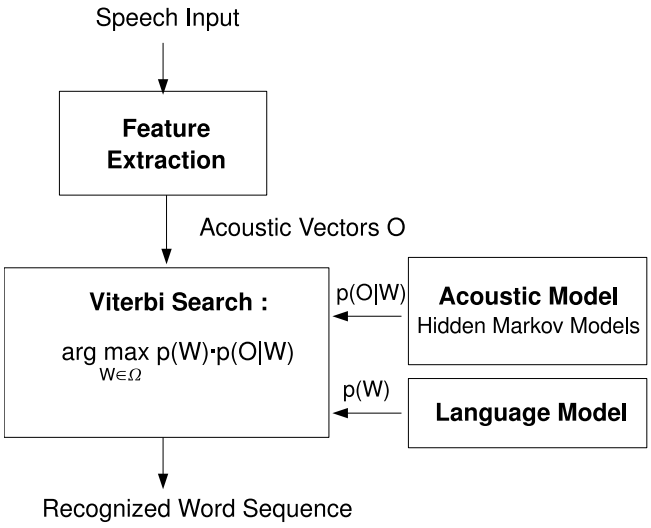

Figure 3.1: Architecture of an automatic speech recognition system.

As shown in figure 3.1, the architecture of an automatic speech recognition system consists of the following four main components:

- The feature extraction module which generates the acoustic feature vectors. The extraction of the acoustic features is usually based on a short-time spectral analysis of the acoustic signal.
- The acoustic model which gives the probability $p(O|W)$ to observe a sequence of acoustic feature vectors O for a given word sequence W. For medium and large vocabulary systems, the acoustic models are usually not defined on a word level but on a subword-unit level (e.g. phoneme level). In the case of a phoneme model, a pronunciation lexicon which contains the phonetic transcription of each word is used to concatenate the phonemes to word models.
- The language model which is trained on large collections of written text, e.g., transcriptions of acoustic data or newspaper articles.

- The search procedure which combines all stochastic models and determines the word sequence with the highest sentence posterior probability for a given speech signal.

These components will be explained in more detail in the following sections:

3.1 Feature Extraction

Acoustic feature extraction aims at generating a parametric representation of the speech waveform (e.g. figure 3.2), usually as a sequence of feature vectors. These acoustic features have to be robust enough to discriminate between the different basic speech units.

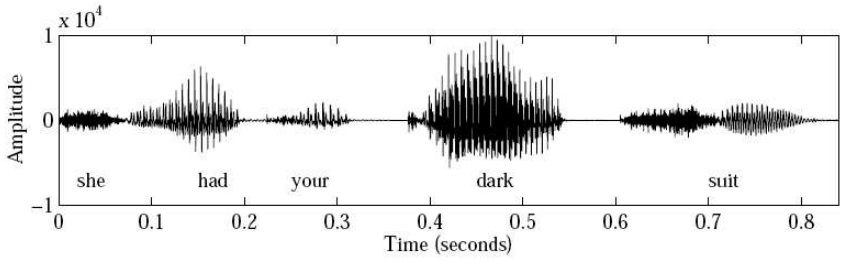

Figure 3.2: An example of speech waveform.

In the first step of the feature extraction process, the analog microphone signal is converted into a digital signal. Although possessing relevant information, high frequencies have smaller amplitude with respect to low frequencies. Therefore, a preemphasis of high frequencies is required to obtain similar amplitude for all frequencies [4]. The digital signal is then processed using a first order *pre-emphasis* filter to obtain similar amplitude for low and high frequencies:

$$x'(n) = x(n) - a \cdot x(n-1), \tag{3.4}$$

where a is the preemphasis parameter and a typical value for a is 0.95.

Speech is a non-stationary signal, i.e. its statistical charateristics are invariant with respect to time. Therefore, a short time analysis can be performed to hold articulatory stability. The spectral features are extracted from a small window of speech that characterizes a particular subphone and for which it can be assumed that the signal is stationary. Typically, every 10 milliseconds (ms) a frame of 25 ms duration, $x_t(n), t = 1, ..., T$, is extracted from the speech signal and multiplied by a *Hamming* window:

$$x_t(n) \equiv w(n) \cdot x'_t(n), \quad 1 \leq t \leq T \tag{3.5}$$

$$w(n) = 0,54 - 0,46 \cdot cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, ..., N-1. \tag{3.6}$$

The multiplication of the speech wave by the window function gradually attenuates the amplitude at both ends of the extraction interval to prevent an abrupt change at the end points.

Then, for each frame, a low-dimensional feature vector is generated by using several methods. The most frequently used methods for signal analysis are the *mel frequency cepstral coefficients (MFCC)* analysis [4] and the *perceptual linear prediction (PLP)* analysis [80]. In this work we will use the *mel frequency cepstral coefficients (MFCC)* analysis.

During the generation of MFCC features, the *Discrete Fourier Transform (DFT)* is used to know how much energy the signal contains at different frequency bands. The input to the DFT is a windowed signal, and the output, for each of N discrete frequency bands, is a complex number $x_t(k)$ representing the magnitude and phase of that frequency in the signal:

$$x_t(k) = x_t\left(e^{-j2\pi k/N}\right), \;\; k = 0, ..., N-1. \tag{3.7}$$

A commonly used algorithm for computing the DFT is the *Fast Fourier Transform (FFT)*. FFT is very efficient, but only works for values of N that are powers of two.

The results of the FFT is information about the amount of energy at each frequency band. However, human hearing is not sensitive at all frequency bands. Spectral features of speech are generally obtained as the result of filter banks, which properly integrate a spectrum at defined frequency ranges. A set of $M = 24$ band-pass filters is generally used since it simulates human ear processing. Filters are usually non-uniformly spaced along the frequency axis (see figure 3.3):

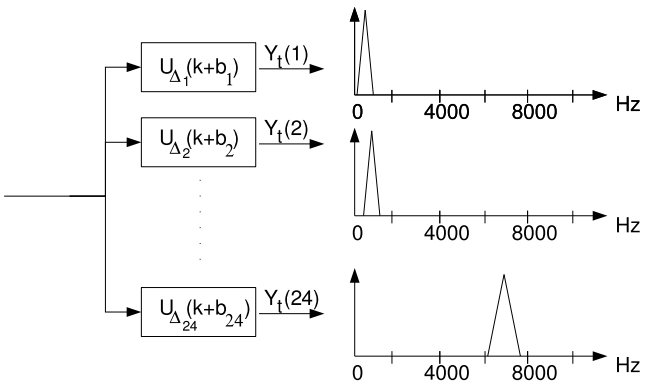

Figure 3.3: Bank of filters scaled according to Mel scale [4].

$$U_{\Delta_m}(k) = \begin{cases} |k| < \Delta_m \rightarrow & 1 - |k|/\Delta_m \\ |k| \geq \Delta_m \rightarrow & 0 \end{cases}, \tag{3.8}$$

where k is the DFT domain index, and $2\Delta_m$ is the size of the m-*th* filter bank triangular window.

The m-*th* filter bank output is given by:

$$Y_t(m) = \sum_{k=b_m-\Delta_m}^{b_m+\Delta_m} x_t(k) U_{\Delta_m}(k + b_m), \tag{3.9}$$

where $1 \leq m \leq M$, and the central frequency may be computed according to $b_m = b_{m-1} + \Delta_m$. For the frequency $f < 1$ kHz, Δ_m is chosen so that 10 uniformly spaced filters are obtained, and, for $f > 1$ kHz, the following approximation can be used: $\Delta_m = 1.2 \times \Delta_{m-1}$.

This leads to a significant reduction of the information content into a smaller number of 24 coefficients. After that, the logarithm of the filter bank coefficients is computed to make feature extraction less sensitive to variations in dynamics.

The final procedure for the Mel frequency cepstrum computation (MFCC) consists of performing the Discrete Cosine Transform on the logarithm of the magnitude of the filter bank coefficients:

$$y_t(k) = \sum_{m=1}^{M} \log\{|Y_t(m)|\} \cdot cos\left(k\left(m - \frac{1}{2}\right)\frac{\pi}{M}\right), \;\; k = 0, ..., L, \tag{3.10}$$

where L is the number of MFCC coefficients.

Cepstral coefficients are usually joined to an energy coefficient e_t taking into account the logarithm of the energy of the frame. This parameter is useful since differences in energy are seen among different phonemes. The energy is computed as the log of the signal energy:

$$e_t = \log \sum_{n=0}^{N-1} {x'}_t(n). \tag{3.11}$$

A further improvement in performance is obtained by taking into account the dynamic evaluation of the speech signal, since such evaluation carries relevant information for automatic speech recognitions. For this reason, features related to the change in cepstral features over time are also added. To do this, the first and second order derivatives of the cepstrum coefficients are added. A simple way to compute derivatives is just to compute the difference of cepstral values between frames. Given an acoustic feature vector $y_t(k)$ indexed in time t, the ith order time difference can be computed as:

$$\Delta^i\{y_t(k)\} = \Delta^{i-1}\{y_{t+1}(k)\} - \Delta^{i-1}\{y_{t-1}(k)\}, \;\; \Delta^0\{y_t(k)\} = y_t(k). \tag{3.12}$$

The feature vectors computed at time t may be composed of a set of $L+1$ acoustic features $\{e_t, y_t(1), y_t(2), ..., y_t(L)\}$, and their first and second-order differences:

$$y_t = \{e_t, y_t(k), \Delta\{e_t\}, \Delta\{y_t(k)\}, \Delta^2\{e_t\}, \Delta^2\{y_t(k)\}\}. \tag{3.13}$$

3.2 Acoustic Modeling (Hidden Markov Model)

The goal of acoustic modeling is to provide an estimation of the probability of the observed spectral feature vectors $O = o_1, ..., o_T$ given a sequence of linguistic units (phones, subparts of words, words) $W = w_1, ..., w_N$, i.e., $P(O|W)$.

The core acoustic modeling technique in ASR is HMM, which is a stochastic finite automaton that consists of a network of states, each of which models the acoustic characteristics of speech. There is a wide range of applications of HMM in literature [24, 7, 23, 44, 82, 80].

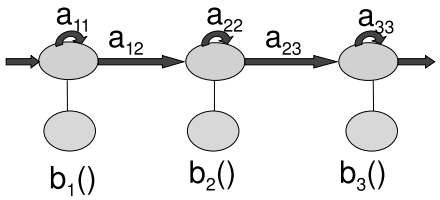

Figure 3.4: An HMM consisting of three emitting states in the left-right topology (a_{ij}: transition probability, $b_j()$: emission probability).

Figure 3.4 illustrates an HMM consisting of three emitting states in the left-right topology or the so-called Bakis topology. An HMM can readily model signals whose properties change over time in a successive manner, e.g. speech. The network of states in an HMM generates a discrete time signal by changing state in each time step according to some transition probabilities and generating in each time step a single observation according to the state dependent emission probability. The fundamental property of a left-right HMM is that no transitions are allowed to states whose indices are lower than that of the current state, as shown in figure 3.4.

An HMM is characterized by the following components:

- a set of states $Q = q_1, q_2, ..., q_N$
- a set of observations $O = o_1, o_2, ..., o_N$
- the transition probabilities $A = a_{ij} = p[q_{t+1} = j | q_t = i]$, $1 \leq i, j \leq N$, representing the probability of moving from state i to state j

- the emission probability $B = \{b_j(o_t)\}$, expressing the probability of an observation o_t being generated from state j
- initial state distribution $\pi = \{\pi_i\},\ 1 \leq i \leq N$.

Given a state sequence $X = x(1), x(2), ..., x(T)$, the probability of the observation sequence O can be computed as

$$p(O, X|W) = \pi_{x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)}. \tag{3.14}$$

If the observations were characterized as discrete symbols chosen from a finite alphabet, a discrete probability density within each state could be used. However, the observations are usually continuous signals or vectors. Hence, it would be advantageous to be able to use HMMs with continuous observation densities to model continuous signal representation directly.

The most general representation of the state output distribution is a finite mixture of the form

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} N(o_t, \mu_{jk}, \Sigma_{jk}), \tag{3.15}$$

where M is the number of mixture components, c_{jk} is the weight of the k'th component and $N(\cdot, \mu, \Sigma)$ in this thesis represents a multivariate Gaussian with mean vector μ and covariance matrix Σ, i.e.,

$$N(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(o_t - \mu_j)' {\Sigma_j}^{-1} (o_t - \mu_j)}, \tag{3.16}$$

where n is the dimensionality of the speech vector o_t.

3.2.1 Lexicon

When using an acoustic subword-unit modeling, a knowledge source is required, which describes the composition of words from the subword-units. This knowledge source is usually called lexicon. The lexicon assigns a single or multiple alternative sequences of subword-units to each word in vocabulary. Some examples, taken from a lexicon with phoneme transcriptions, are presented in table 3.1.

Table 3.1: *Examples of pronunciation lexicon entries.*

Word	Pronunciation
CAT	k ae t
IS	ih z
TABLE	t ey b el
THIS	th ih s

3.2.2 Subword-Unit Modeling

For medium and large vocabulary speech recognition systems, whole word models as acoustic elementary units are usually not possible due to the insufficient number of training samples. Thus, this possibility is typically only used for small vocabularies, like a digit recognition task.

To increase the number of available training samples for the elementary acoustic units, smaller pieces of speech may be addressed instead of whole words. The phoneme is the most popular smallest sound unit that can be used to differentiate between words. Of course, the usage of phonemes requires the incorporation of additional knowledge on how to concatenate the basic units in order to build models for whole words. This additional knowledge is given using a lexicon describing the composition of words (see section 3.2.1).

Figure 3.5 illustrates a sentence in which an acoustic model for a word is formed by concatenating the HMMs for the acoustic subword-units (e.g. phonemes) comprising the word. Each word in the sentence can be looked up in the lexicon to find its transcription in terms of acoustic subword-units. Hence, the sentence can be written in terms of acoustic subword-units (e.g. phonemes).

Since the physical articulators are not capable of performing sudden, drastic movement, the phonemes are influenced by the preceding and following phonemes. In order to capture the effects of co-articulation between adjacent phonemes, context-dependent phonemes have been introduced [87, 72]. The most important context-dependent units are *biphones* (left or right phone context) and *triphones* (left and right context).

The construction of context dependent models for the phonemes within the words (word-internal triphones) is relatively simple to implement because the context of a phoneme within a word is directly given by the corresponding entry in the pronunciation lexicon for that word. In the so-called *word-internal modeling* the dependency of the word boundary triphones on the neighboring words is neglected. For the word boundaries biphones are used. For example, the sentence “this is cat” can be represented using monophones:

th ih s / ih z / k ae t.

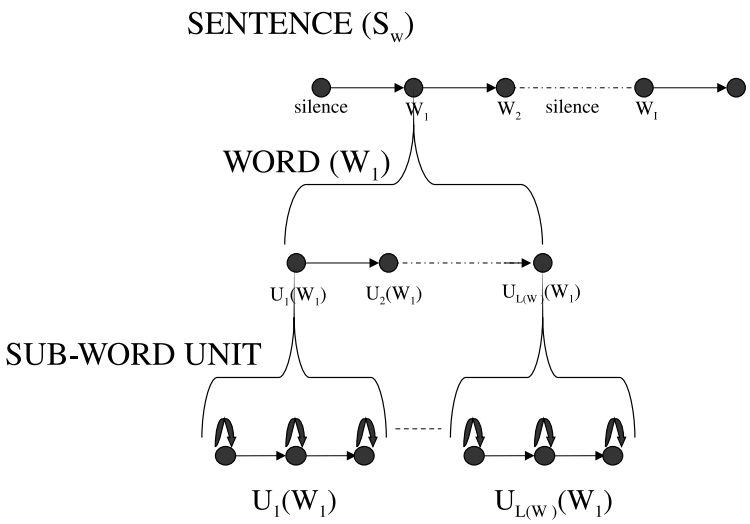

Figure 3.5: Representation of a sentence, word and acoustic subword-unit in terms of HMMs.

The same sentence can be represented using biphone and triphone models:

th+ih th-ih+s ih-s / ih+z ih-z / k+ae k-ae+t ae-t.

Across word modeling explicitly treats co-articulation effects, independent of the position of word boundaries [38]. In contrast to word-internal modeling, the ending phoneme of the preceding word and the starting phoneme of the succeeding word are used as a context for the word under consideration. This modeling technique significantly reduced the word error rates, but, at the expence of higher computational costs. When cross-word triphones are used, the example sentence is represented as

-th+ih th-ih+s ih-s+ih / s-ih+z ih-z+k / k+ae k-ae+t ae-t+,

where * denotes the beginning and the end of the sentence.

The number of triphones may become larger than the number of vocabulary words. This gives rise to the trainability problem, i.e., not enough data per model. This problem is handled by merging similar context models together (see section 3.2.5).

3.2.3 Assumptions of Acoustic Modeling

In this thesis, the acoustic modeling of automatic speech recognition is based on the following assumptions [22].

- Speech is assumed to be stationary during a short period of time (e.g. 25 ms). A feature vector is usually computed from this stationary period of speech signals.

- A Gaussian distribution is usually used to represent the distribution in a state. This means that the speech is assumed to be Gaussian. That is, the random deviation of a speech sound in a state follows Gaussian distribution.
- For the simplification of the probability computation, each speech feature vector is assumed to be independent. This assumption leads to diagonal covariance matrices, and a speeding-up in the computation.

3.2.4 Estimation of Model Parameters

In order to determine the model parameters that maximize the probability of the observation sequence, the so-called Baum-Welch re-estimation formulae is applied. We define first $\xi_t(i,j)$, the probability of being in state i at time t and state j at time $t+1$, given the model and the observation sequence, i.e.,

$$\xi_t(i,j) = \frac{p(q_t = i, q_{t+1} = j|O,W)}{P(O|W)} \tag{3.17}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}. \tag{3.18}$$

This probability is calculated using the so-called *Forward-Backward* algorithm. The forward probability can be defined as

$$\alpha_t(i) = p(o_1, o_2, ..., o_t, q_t = i). \tag{3.19}$$

That is the joint probability of observing the first t speech vectors and being in state i at time t. This forward probability can be efficiently calculated by the following recursion

- Initialization
$$\alpha_1(i) = \pi_i b_i(o_1), \ \ 1 \leq i \leq N \tag{3.20}$$
- Induction
$$\alpha_{t+1}(i) = [\sum_{j=1}^{N}\alpha_t(j)a_{ji}]b_i(o_{t+1}), \ \ 1 \leq i \leq N, \ \ 1 \leq t \leq T-1 \tag{3.21}$$
- Termination
$$p(O|W) = \sum_{i=1}^{N}\alpha_T(i). \tag{3.22}$$

Step 1 initializes the forward probabilities as the joint probability of state i and initial observation o_1. The induction step (step 2) calculates the probability that state i can be reached at time $t+1$ from N possible states. Finally, step 3 gives the total likelihood $P(O|W)$.

The backward probability $\beta_t(i)$ is defined as

$$\beta_t(i) = p(o_{t+1}, o_{t+2}...., o_T|q_t = i). \tag{3.23}$$

As in the forward case, this backward probability can be computed efficiently using the following recursion

- Initialization
$$\beta_T(i) = 1, \;\; 1 \leq i \leq N \tag{3.24}$$
- Induction
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \;\; t = T-1, T-2, ..., 1, \;\; 1 \leq i \leq N. \tag{3.25}$$

The initialization step 1 arbitrarily defines $\beta_T(i)$ to be 1 for all i. Step 2 calculates the probability of having been in state i at time t under the condition of considering all possible states j at time $t+1$.

Let the probability of being in state i at time t be defined as

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{3.26}$$

Using the above formulas 3.18 and 3.26, the model parameters a_{ij}, the mean μ_j and the variance Σ_j for the state output distribution $b_j(o_t)$ can be estimated as follows:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \tag{3.27}$$

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t}{\sum_{t=1}^{T} \gamma_t(j)}, \tag{3.28}$$

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) (o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{3.29}$$

3.2.5 Training of Hidden Markov Models

In the first step of training procedure, the phoneme model parameters are initialized using either so-called flat start procedure (i.e, each training utterance is uniformly segmented and all states of all models are made equal), or a bootstrapping method (i.e., each training utterance is divided into phonemes and the phoneme states are initialized with the corresponding data). The *Baum-Welch* algorithm based embedded training is then used to train the model parameters.

Embedded training works as follows. Every training file must have an associated label file which gives a trancription for that file, and it is processed in turn. The training method uses the associated transcription to construct a composite HMM which spans the whole utterance. This composite HMM is made by concatenating instances of the phoneme HMMs corresponding to each label in the transcription. The Forward-Backward algorithm is then applied and the sums needed to form the weighted averages are accumulated in the normal way. When all of the training files have been processed, the new parameter estimates are formed from the weighted sums and the HMM set is updated.

In order to increase the performance in speech recognition tasks the set of subword-units (phonemes) is extended to include context-dependent units (e.g. triphones). Before building a set of context-dependent models, it is necessary to decide on whether or not cross-word triphones are to be used. If they are, then word boundaries in the training data can be ignored and all phoneme labels can be converted to triphones. If, however, word internal triphones are to be used, then word boundaries in the training transcriptions must be marked in some way (either by an explicit marker which is subsequently deleted or by using a short pause).

In order to create triphone models, the set of initialized and trained context-independent phoneme models is converted to a set of context dependent models by simply cloning phonemes and then re-estimating using triphone transcriptions, e.g., a triphone model *l-p+r* denotes the context-dependent version of the phoneme *p* which is to be used when the left neighbour is the phoneme *l* and the right neighbour is the phoneme *r*.

This will lead to a very large set of models, and relatively insufficient training data for each model. Hence, the states within triphone sets are tied to share data and to be able to make robust parameter estimates. This tying can be done by two mechanisms, data-driven clustering and tree-based clustering. The first uses a similarity measure between states based on Euclidean distance. Figure 3.6 shows the clustering and the tying of the corresponding states of the triphone group for the phoneme “ih” using the data-driven clustering mechanisms.

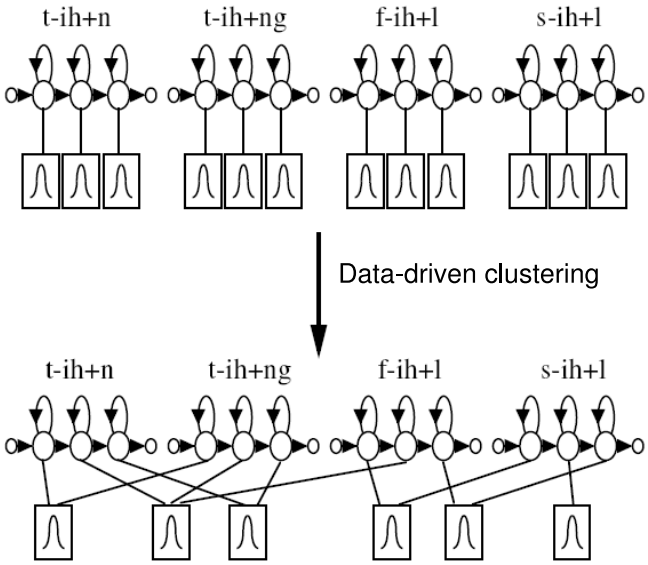

Figure 3.6: Data-driven state tying [103].

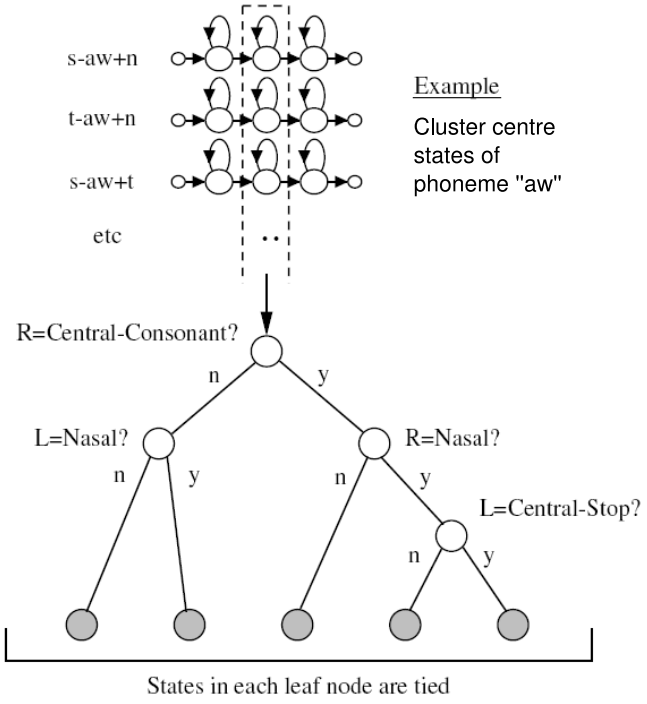

Figure 3.7: Decision-tree based state tying [103].

The second uses phonetic decision trees and is based on asking questions about the left and right contexts of each triphone. A phonetic decision tree is a binary tree in which a yes/no phonetic question is attached to each node. Figure 3.7 illustrates the case of tying the centre states of all triphones of the

phoneme "aw" (as in "out"). All of the states trickle down the tree and depending on the answer to the questions, they end up at one of the shaded terminal nodes. For example, in the illustrated case, the centre state of "s-aw+n" would join the second leaf node from the right since its right context is a central consonant, and its right context is a nasal but its left context is not a central stop. The details of these algorithms can be found in the HTK Book [103].

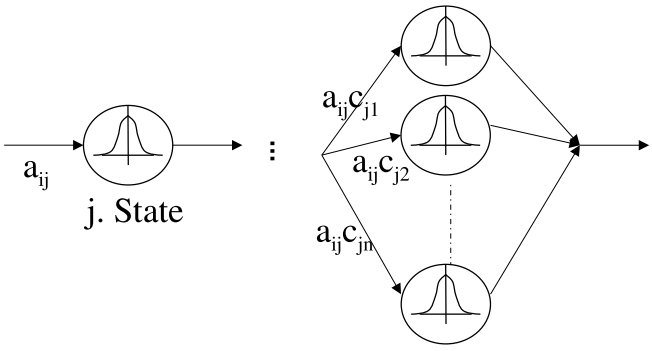

Figure 3.8: Representing a Mixture.

The final step in the building of HMMs is usually the conversion from single Gaussian HMMs to multiple mixture component HMMs (see figure 3.8). The number of mixture components is repeatedly increased by splitting the "heaviest" mixture component until the required number of components is obtained. The split is performed by copying the mixture component, dividing the weights of both copies by 2, and finally perturbing the means by plus or minus σ standard deviations.

3.3 Language Modeling

The goals of the (statistical) language model are to specify legal sequences of words and to provide probabilities of sequence of words W for a given recognition task. The speech recognizers produce a string of words, guided by their language models. When the recognition search comes to the end of a word, it uses a language model to determine which words to search for following the current word. If it is assumed that W is a specified sequence of words, i.e.,

$$W = w_1 w_2 ..... w_m \tag{3.30}$$

then $p(W)$ can be computed as

$$p(W) = p(w_1 w_2 ..... w_m) = \prod_{i=1}^{m} p(w_i | w_1 ... w_{i-1}). \tag{3.31}$$

Unfortunately, it is essentially impossible to reliably estimate the conditional probabilities for all words and all sequence lengths in a given language. Hence, in practice, it is convenient to use an N-gram word model, where the term $p(w_j|w_1....w_{j-1})$ is based on the preceding N-1 words and approximated as

$$p(w_j|w_1....w_{j-1}) \approx p(w_j|w_{j-N+1}....w_{j-1}) \tag{3.32}$$

$$= \frac{C(w_j, w_{j-1}, ..., w_{j-N+1})}{C(w_{j-1}, ..., w_{j-N+1})}, \tag{3.33}$$

where C is the count of a given word sequence in the training corpus. Even N-gram probabilities are difficult to estimate reliably, for all values of N, but for when $N = 2$ (bigram) or possibly 3 (trigram). A bigram language model assumes that the probability of a word w_n depends only on the immediate predecessor word w_{n-1}

$$p(W) \approx \prod_{i=1}^{m} p(w_n|w_{n-1}), \tag{3.34}$$

while in a trigram language model the probability is assumed to depend on the two immediate predecessor words w_{n-1} and w_{n-2}

$$p(W) \approx \prod_{i=1}^{m} p(w_n|w_{n-1}, w_{n-2}). \tag{3.35}$$

Hence, in practice, it is often convenient to use a bigram or trigram word model.

3.4 Viterbi Search

The *Viterbi* search is used to find the most likely sequence of states in a Markov chain to produce an observation sequence of feature vectors. Figure 3.9 shows the structure of a Markov chain for the generic speech model [59]. It is built by connecting word HMMs in parallel in order to recognize words. Each word HMM can be constructed of subword HMMs. In this figure, each word can jump to any word in the vocabulary. Between words, there may be a short silence depending on the isolated word or continuous speech recognition. In the case of subword-unit (e.g. phonemes or syllables) recognition the word HMMs are replaced with subword-unit HMMs. When the bigram is used as a language model, the path from W_i to W_j has the bigram probability $P(W_j|W_i)$. This bigram probability can be viewed as a state transition probability from the exit state of W_i to the entry state of W_j.

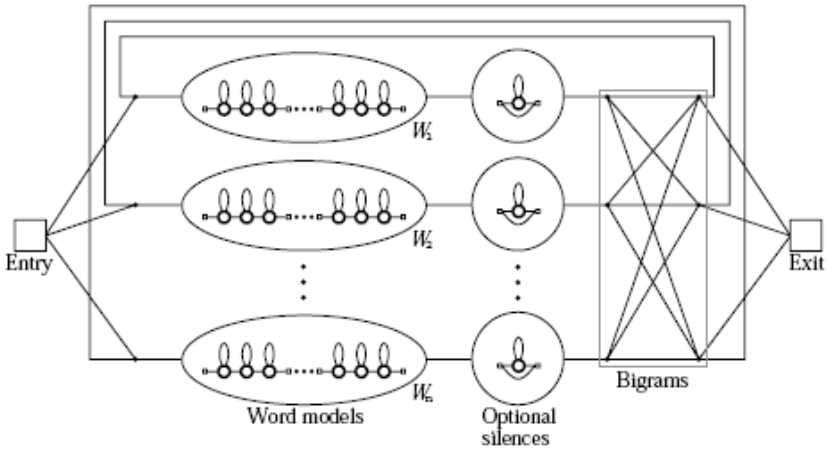

Figure 3.9: Generic HMM model.

In order to find the best state sequence, $q = (q_1 q_2 ....q_T)$, for the given observation sequence $O = (o_1 o_2 ....o_T)$, the following quantity is defined

$$\delta_t(i) = \max_{q_1,q_2,...,q_{t-1}} p(q_1 q_2 ....q_{t-1}, q_t = i, o_1 o_2 ...o_t), \tag{3.36}$$

that is, $\delta_t(i)$ is the highest score along a state sequence at time t, which takes into consideration the first t observations and ends with state i. The Viterbi algorithm for finding the best state sequence is given as follows:

- Initialization

$$\delta_1(i) = \pi_i b_i(o_1), \ 1 \leq i \leq N \tag{3.37}$$
$$\psi_1(i) = 0 \tag{3.38}$$

- Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1} a_{ij}] b_j(o_t), \ 2 \leq t \leq T, \ 1 \leq j \leq N \tag{3.39}$$
$$\psi_t(j) = arg \max_{1 \leq i \leq N} [\delta_{t-1} a_{ij}], \ 2 \leq t \leq T, \ 1 \leq j \leq N \tag{3.40}$$

- Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.41}$$
$$q_T{}^* = arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.42}$$

- State sequence backtracking

$$q_t{}^* = \psi_{t+1}(q_{t+1}{}^*), \ t = T-1, T-2, ..., 1, \tag{3.43}$$

where $\psi_t(j)$ is the previous state in the best path to state j at time t.

This algorithm is applicable when it is based on the bigram language model. If the trigram model is used, the bigram transition probabilities in figure 3.9 can no longer be used, because the trigram is second order dependent. The generic speech model becomes too complex to incorporate trigrams. In this case, it becomes a tree with height 3, where each node in the tree represents a word model and has all words in the vocabulary as its children. The Viterbi algorithm is exponential with respect to the order of the language model. In large vocabulary continuous speech recognition, higher order N-grams play an important role. However, using a more complex model is computationally expensive, especially when the size of vocabulary is large.

The Viterbi algorithm is similar to the forward-backward procedure. The big advantage of the *Viterbi* algorithm is that no sums of probabilities need to be computed. Therefore, it is possible to simply work with the logarithm of all probabilities, computing sums of logarithmic probabilities rather than products of probabilities.

To efficiently restrict the search space of Viterbi search, a number of concepts have been introduced. One of the most important is *Beam search*: at each time frame only search hypotheses whose probability is above a certain threshold, are expanded. A careful tuning of this pruning threshold may prevent search errors.

3.5 Discussion

Modern general-purpose speech recognition systems are generally based on HMMs. These are statistical models which output a sequence of symbols. One possible reason why HMMs are used in speech recognition is that a speech signal could be viewed a short-time stationary signal. That is, in a short-time in the range of 10 milliseconds, speech could be approximated as a stationary process. Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. Speech is processed in HMMs using a sequence of n-dimensional vectors (with n being a small integer), each of which is computed every 10 milliseconds. The vectors consist of cepstral coefficients. The HMM has in each state a statistical distribution that is a mixture of diagonal covariance Gaussians which will give a likelihood for each observed vector. Each acoustic model will have a different output distribution. The most widely used training technique is the forward-backward algorithm, which creates an HMM that maximizes the probability that the HMM produces the observation sequence in the training data.

For medium and large vocabulary speech recognition system, a word based

recognizer requires an HMM built for each distinct word and more training data. In addition, a word-based recognizer needs a much larger memory because it must store every parameter for every state in every word. Therefore, a phoneme-based recognizer requires a relatively small number of phonemes. An HMM for a sequence of words or subword-units is made by concatenating the individual trained HMMs for the separate acoustic models, e.g., phonemes.

Recognition accuracy of HMMs can be greatly improved by taking advantage of the possible a priori information on the sequences to be recognized. This information can be embedded in a language model. The role of a language model is to reduce the set of acceptable words of the search space. During recognition, the Viterbi algorithm finds the sequence of states of HMMs in the search space that can yield the given sequence of observations with the highest probability.

Adding novel words to the task vocabulary is usually not easy for HMMs, especially during runtime. In case of subword-unit modeling, for the novel words, involved acoustic models must be retrained. If there are unseen acoustic units in the novel word, new acoustic models must be created and trained. The lexicon must be extended with the novel transcription. In addition, the language model must be modified with respect to the new transcription.

HMMs suffer from some limitations in real-world environments. Although HMMs allow for excellent recognition performance in laboratory conditions, there is a decrease in their performance for real-world conditions.

In the scope of this work, HMMs are used for subword-unit recognition. For acoustic modeling, context-dependent phonemes are used. Depending on the size of the task vocabulary, output subword units are context-dependent phonemes or longer subword units like syllables.

Chapter 4

Willshaw Associative Memory

Associative memories are content-addressable structures that store a set of M associations between pattern pairs that are represented as binary input $x^\mu \in \{0,1\}^m$ and output $y^\mu \in \{0,1\}^n$ vectors with $\mu = 1, 2, ..., M$.

In neural associative memories, the associations between patterns are stored in the synaptic connections between two neuron populations. In 1969 David Willshaw and others introduced a binary model of neural associative memory [100]. The neurons in Willshaw's model are simple binary threshold neurons, i.e. they become active if their membrane potential (input) exceeds a given threshold, otherwise, they are inactive.

Neural implementations of associative memories can have an advantage over simple look-up tables if the number of patterns is large and parallel implementation is possible. Its other advantage is fault tolerance, when the address patterns $\tilde{x}^\mu$ differ from the previously stored original patterns x^μ.

The memory capacity of an associative memory is the amount of information that can be retrieved from the memory matrix and it can be measured theoretically by maximizing the transinformation between the stored patterns and the retrieved patterns. The memory capacity per synapse is measured by normalizing the memory capacity to the physical memory.

In 1969 Willshaw et al. [100] discovered that a high memory capacity per synapse of $\ln 2 \approx 0.7$ is possible for Steinbuch's neural associative memory using binary patterns and synapses [94]. In 1980 Palm elaborated these results [73]. This analysis revealed that the upper bounds $\ln 2$ for heteroassociation and $\ln 2/2$ for autoassociation are only approached asymtotically for very large numbers of neurons [89, 70]. High memory capacities are only possible for sparse patterns, i.e., the number k of one-entities is much smaller than the pattern size n. For k values smaller or larger than the optimal value, the capacity usually decreases rapidly to smaller values. It is also important that high capacities can only be achieved for random patterns, i.e., the stored patterns are independent.

In the mid eighties alternative neural implementations were proposed [39, 40, 41], with results proving that the capacity of the Willshaw model exceeds the capacities of other models.

In the late eighties and nineties methods were developed to improve the retrieval results under noisy conditions by bidirectional and iterative retrieval [58, 89, 93], and an early attempt to use sparse matrices was neither efficient nor very promising [9]. In 1996 Schwenker et al. [89] showed that for pattern mapping in the autoassociative memory, iterative retrieval can be used for a better retrieval of the stored information. For heteroassociation, the upper theoretical bound of the memory capacity per synapse was extended to $1/(2\ln 2) \approx 0.72$ for Hebb-like synapses which rewards coincident pre- and postsynaptic activity [97, 76, 77]. In spite of this, the bound of $\ln 2$ for the memory capacity per synapse was never exceeded for non-trivial models.

In 1997 Graham and Willshaw analyzed the capacity and efficiency of the heteroassociative memory for varying pattern coding rates, connectivity and cue noise levels and they showed that altering the connectivity is not an effective way of changing the capacity of an associative net, and generally the capacity is more sensitive to net size than to connectivity [31]. For the more realistic case of neural associative memories, i.e. a partially connected neural network using normalized *winner-takes-all* recall [16, 30], maximum capacity of 0.53 obtained at 1% connectivity approaches the theoretically approximate maximum of $\ln 2$ for a fully connected network [29], where a simple *winner-takes-all* approach chooses the required number of output units with the highest dendritic sums to activate. In 1998 Bosch and Kurfess showed that the maximum capacities for incompletely connected associative memories range from 0.53 to 0.69, depending on the connectivity of the network [10]. These capacities can be obtained for sparse input and output patterns.

In this chapter we will study the binary **Willshaw model** of neural associative memory [100, 73, 74, 94, 76, 17, 89, 93] in terms of memory capacity, retrieval efficiency, and fault tolerance. First, we review the storage and retrieval algorithms used in the Willshaw model. Then, we analyse its storage and memory capacity. Finally, we investigate the fault tolerance of the Willshaw model.

4.1 Storing and Retrieving Patterns

For the set of associations between M pattern pairs $\{(x_\mu, y_\mu) : \mu = 1, ..., M\}$ we assume that all patterns are binary vectors of length n containing k ones.

The patterns are stored heteroassociatively in the binary memory matrix $W \in \{0,1\}^{n\times n}$ corresponding to synaptic connections between two neuron populations. The address population corresponds to the patterns x^μ, and the con-

tent population corresponds to the patterns y^μ. Matrix entry w_{ij} corresponds to the synaptic weight of the connection from neuron i in the address population to neuron j in the content population. A pattern is stored by activating the pattern neurons and Hebbian learning activates all synaptic connections between two pattern neurons. After storing M patterns the matrix entry w_{ij} is obtained from the superposition of outer products of the pattern vectors.

$$w_{ij} = min\left(1, \sum_{\mu=1}^{M} {x_i}^\mu \cdot {y_j}^\mu\right) \in \{0,1\}. \tag{4.1}$$

The storage is considered distributed because many matrix entries carry information about more than one pattern pair. In this sense, storage of several pattern pairs will affect the same synapses, so that each entry in the memory matrix W may contain the superposition of several memory traces, i.e., the sum $\sum_{\mu=1}^{M} {x_i}^\mu \cdot {y_j}^\mu$ should have more than one nonzero contribution for the matrix entry w_{ij} [78].

For pattern retrieval an address pattern $\tilde{x}$ is applied to the associative memory W. The applied address pattern $\tilde{x}$ may be a noisy version of one of the original address patterns x^μ. First dendritic potentials are obtained by a vector matrix multiplication of the input vector $\tilde{x}$ and the memory matrix W. The retrieval result is then obtained by applying a threshold Θ,

$$\hat{y}_j = \begin{cases} 1, & \sum_{i=1}^{m} \tilde{x}_i w_{ij} \geq \Theta \\ 0, & otherwise \end{cases}. \tag{4.2}$$

The output unit is set to 1, i.e. the output neuron is activated, if its dendritic potential is equal to or larger than the threshold Θ. Otherwise, it is set to 0. The choice of the threshold Θ is important to obtain good retrieval results. One possibility is the *Willshaw retrieval strategy* where the threshold is set equal to the number of one-entries in the address pattern $\tilde{x}$,

$$\Theta = \sum_{i=1}^{m} \tilde{x}_i. \tag{4.3}$$

Indeed, this strategy is only possible if the applied address pattern contains no false ones. In the case that the address pattern contains false ones, an alternative strategy can be used, where the threshold Θ is set to a global value. For example, this value could be equal to the maximum value of the dendritic potentials of output units.

Pattern retrieval is designed to be fault tolerant, i.e. there might be additional or missing ones in the address pattern. This means that the address pattern

does not have to be equal to one of the previously stored patterns x^μ but might contain errors in the form of additional or missing ones. In the case of an erroneous input pattern $\tilde{x}$, associative memories will try to retrieve the output pattern y^μ that belongs to a previously stored input pattern x^μ with the closest matching address pattern $\tilde{x}$.

If the address pattern $\tilde{x}$ is part of a stored pattern, the Willshaw strategy will activate all the neurons of the content pattern, but possibly also additional neurons. Indeed, this strategy is a possible choice if the address pattern contains no false input neurons. Another strategy can be that the threshold is set to a global value.

For the auto-association where the address and content patterns are the same ($x^\mu = y^\mu$), the Willshaw model can be applied in the same way.

4.2 Storage and Memory Capacity

In the case of storing hetero-associatively M random pattern pairs (x^μ, y^μ), the key parameters are the total number of input units m, the total number of output units n, the number of active units (ones) in the input pattern l, the number of active units (ones) in the output pattern k and the number of pattern pairs to be stored M. In the case that all M patterns have size n and contain exactly k ones, the probability that a given synapse is not set by the storage of one pattern pair is

$$p(w_{ij} = 0) = \left(1 - \frac{k^2}{n^2}\right)^M, \tag{4.4}$$

because there are $\begin{pmatrix} M \\ k \end{pmatrix}$ possibilities to distribute k ones on M places.

$$p(w_{ij} = k) = \begin{pmatrix} M \\ k \end{pmatrix} \left(\frac{k^2}{n^2}\right)^k \left(1 - \frac{k^2}{n^2}\right)^{M-k} \tag{4.5}$$

$$p(w_{ij} = 0) = \begin{pmatrix} M \\ 0 \end{pmatrix} \left(\frac{k^2}{n^2}\right)^0 \left(1 - \frac{k^2}{n^2}\right)^M = \left(1 - \frac{k^2}{n^2}\right)^M \tag{4.6}$$

Therefore, the probability $p_1 = p(w_{ij} = 1)$ that a particular synaptic weight w_{ij} has been modified after the storage of all M pattern pairs is

$$p_1 = 1 - \left(1 - \frac{k^2}{n^2}\right)^M \tag{4.7}$$

$$\approx 1 - e^{-\frac{M \cdot k^2}{n^2}}. \tag{4.8}$$

From equation 4.7 the number of pattern pairs (storage capacity) can be resolved,

$$M \approx \frac{ln(1-p_1)}{ln(1-\frac{k^2}{n^2})} \tag{4.9}$$

$$\approx -\frac{n^2}{k^2} ln(1-p_1). \tag{4.10}$$

Note that the ones in the binary patterns are chosen independently with probability $\frac{k}{n}$. After storing the patterns an address pattern $\tilde{x}$ containing k ones of the pattern x^μ is applied to the memory matrix $\tilde{x}M$ to retrieve the associated content pattern y^μ. Applying the Willshaw strategy $\Theta = k$ can cause false one-entities due to overlap between patterns. The probability p_{01} of a false one can be approximated with

$$p_{01} \approx {p_1}^k. \tag{4.11}$$

Associative memories are used to store and retrieve a finite number M of patterns. This can be interpreted as transmission and storage of information. In order to obtain the channel capacity of B.9 for a given M, we have to maximize the transinformation between the source and target patterns with respect to the statistics of the source patterns. Thus we can define the *memory capacity*,

$$C := \underset{P}{sup} T(y^1, y^2, ..., y^M; \hat{y}^1, \hat{y}^2, ..., \hat{y}^M), \tag{4.12}$$

where P is the transition probability (equation B.8).

If patterns are independently generated random patterns and the retrieval result $\hat{y}^\mu$ depends only on the corresponding original pattern y^μ, then the memory capacity is simply

$$C \approx M \cdot \underset{P}{sup} T(y^\mu, \hat{y}^\mu). \tag{4.13}$$

For the binary Willshaw model, storing binary patterns $(x^\mu, y^\mu) \in \{0,1\}^n$ the storage capacity can be obtained by applying the theory of binary channels (section B.4)

$$C(k,n,M) \approx MnT(y^\mu; \hat{y}^\mu). \tag{4.14}$$

In the case of heteroassociation, the individual information stored per output pattern I^μ is defined as the difference between the information contained in the output pattern y^μ (i.e. $\log_2 \left(\begin{array}{c} n \\ k \end{array} \right)$) and the information needed to determine the false "ones" in the output pattern:

$$I^\mu = \log_2 \left(\begin{array}{c} n \\ k \end{array} \right) - \log_2 \left(\begin{array}{c} S^\mu + k \\ k \end{array} \right), \tag{4.15}$$

where S^μ is the number of false ones in the output pattern y^μ. The total information stored in the memory I (memory capacity) is

$$C(k,n,M) = I = \sum_{\mu=1}^{M} I^\mu = \sum_{\mu=1}^{M} \log_2 \left(\begin{array}{c} n \\ k \end{array} \right) - \log_2 \left(\begin{array}{c} S^\mu + k \\ k \end{array} \right) \tag{4.16}$$

$$= \sum_{\mu=1}^{M} \sum_{i=0}^{k-1} \log_2 \frac{n-i}{k+S^\mu-i}. \tag{4.17}$$

The expectation value of S^μ is determined as

$$E(S^\mu) = (n-k)p_{01} \approx (n-k){p_1}^k. \tag{4.18}$$

Inserting equation 4.18 into equation 4.17 gives

$$C(k,n,M) \approx -\sum_{\mu=1}^{M} \sum_{i=0}^{k-1} \log_2 \frac{k+E(S^\mu)-i}{n-i} \tag{4.19}$$

$$\approx -M \sum_{i=0}^{k-1} \log_2 \frac{(n-k)p_{01}+k-i}{n-i} \tag{4.20}$$

$$\approx -Mk \log_2 \frac{(n-k){p_1}^k + k}{n}. \tag{4.21}$$

To obtain good retrieval results a *high-fidelity (hifi)-requirement* $p_{01}/(k/n) \to 0$ and $p_{01} \to 0$ as $n \to \infty$ is necessary [76], i.e. the number of false ones is almost zero. This can be obtained by requiring $p_{01} \leq \epsilon k/n$ for a small positive ϵ and extremely sparse output patterns $q = k/n \to 0$.

Then the memory capacity is approximated:

$$C(k,n,M) \approx -Mk \log_2(k/n) \approx Mk \log_2(n/k). \tag{4.22}$$

In other words, for a small $q = k/n$ the information per pattern is $I(q) \approx -(k/n)\log_2(k/n)$ and the totally stored information is approximated:

$$C(k,n,M) \approx MnI(k/n) \approx Mk \log_2(n/k). \tag{4.23}$$

To maximize the memory capacity $C(k,n,M)$ for given k and n, we store as many patterns as possible so that we can still fulfill the *hifi-requirement*. The maximal matrix load is calculated from the *hifi-requirement* and equation 4.11:

$$p_{1,max} = (\epsilon k/n)^{1/k}. \tag{4.24}$$

Substituting $p_{1,max}$ in equation 4.9 we get the maximal number of stored patterns,

$$M_{max} \;=\; \frac{\ln(1-p_{1,max})}{\ln(1-k^2/n^2)} \approx -\frac{n^2}{k^2}\ln(1-p_{1,max}). \tag{4.25}$$

Using the approximation $k \approx -\log_2(n/k)/\log_2 p_{1,max}$ from equation 4.24 the maximal number of stored patterns is approximated as

$$M_{max} \;\approx\; (\log_2 p_{1,max})\cdot\ln(1-p_{1,max})\cdot\frac{n^2}{k\log_2(n/k)}. \tag{4.26}$$

The maximal stored information per synapse is obtained by normalizing the memory capacity to the physical memory:

$$C(k,n) \;:=\; \frac{C(k,n,M_{max})}{n^2} \tag{4.27}$$

$$\approx\; -\ln(1-p_{1,max})\frac{\log_2(n/k)}{k} \tag{4.28}$$

$$\approx\; \log_2 p_{1,max}\ln(1-p_{1,max}), \tag{4.29}$$

where $k \approx -\log_2(n/k)/\log_2 p_{1,max}$ is used from the equations 4.11 and 4.24 for the second approximation. It is seen from the equation 4.28 that the optimal choice of the parameter p_1 is $p_1 = 0.5$ which maximizes the memory capacity C. Therefore $C \to \ln 2$ for $n \to \infty$, which Willshaw et al. discovered in 1969 [100]. These results were refined by Palm in 1980 [73].

Figures 4.1, 4.2, and 4.3 show some results of the classical analysis of the Willshaw model of an autoassociative memory ($l = k$ and hifi-parameter $\epsilon = 0.01$). As can be seen in figure 4.1 the optimal capacity $C \approx 0.49$ is reached for $k = 21$. For larger or smaller k values the capacity decreases rapidly. For optimal k value the memory matrix is not sparse with $p_{1,max} \approx 0.5$. The memory load $p_{1,max}$ increases with k monotonically from 0 to 1. For the optimal k the memory matrix is not sparse $p_{1,max} = 0.5$. Figure 4.2 illustrates that the memory capacity $C(k,n)$ is maximal for $p_{1,max} = 0.5$ but maximal M is obtained for moderately sparse memory matrix ($p_{1,max} \approx 0.16$). Figure 4.3 shows the convergence of $C(n)$ towards $\ln 2$ for $n \to \infty$ is quite slow.

4.3 Fault Tolerance

In this section we will investigate the decrease of memory capacity when the address patterns used for retrieval deviate from the stored ones.

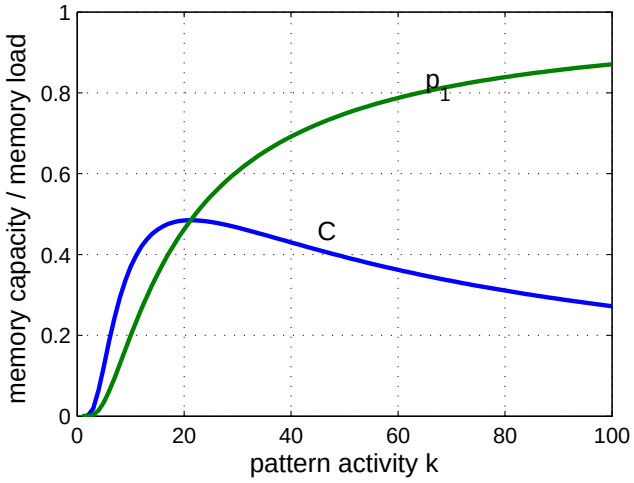

Figure 4.1: Dependence of the memory capacity $C(k,n) \approx \log_2(p_{1,max})\ln(1-p_{1,max})$ and the memory load $p_{1,max} = (\epsilon k/n)^{1/k}$ on the pattern activity k (number of one-entries in a pattern) for pattern size $n = 10^6$.

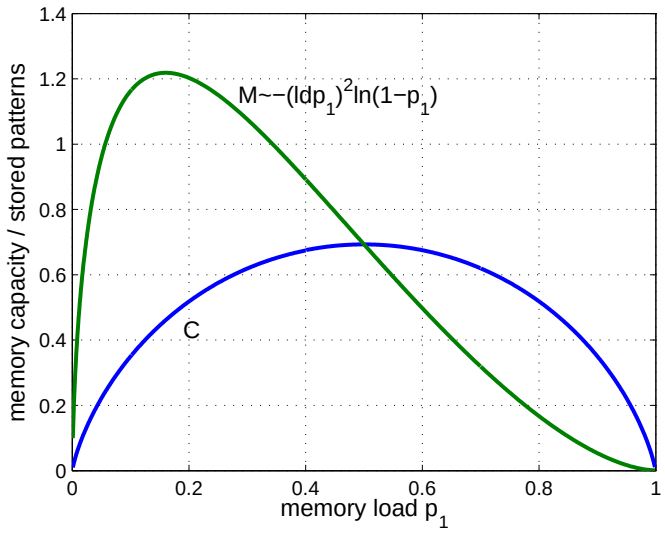

Figure 4.2: Dependence of the memory capacity $C(k,n) \approx \log_2(p_{1,max})\ln(1-p_{1,max})$ and $M \sim (\log_2 p_{1,max})^2 \ln(1-p_{1,max})$ on the memory load $p_{1,max}$ for given pattern size $n = 10^6$.

4.3.1 Missing Ones in the Address Patterns

Let us assume that an address pattern used for retrieval contains a part $\lambda \in (0,1]$ of the l ones of the original address pattern, and there are no false ones in the address pattern. To still achieve good retrieval results according to the hifi-requirements (see section 4.2), we have to store fewer patterns than

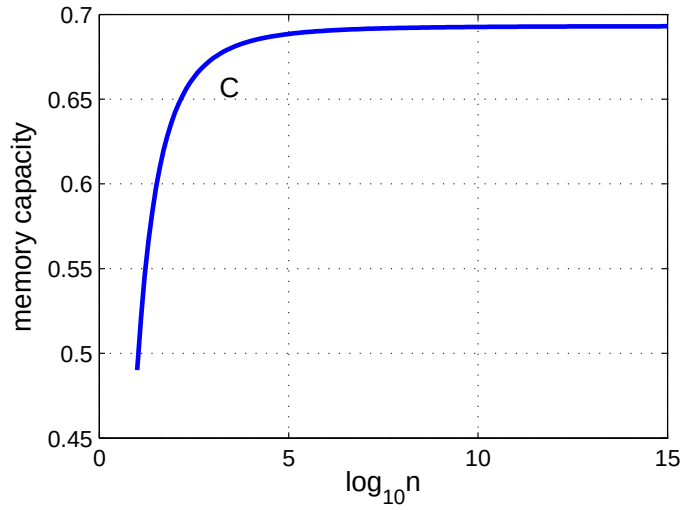

Figure 4.3: The convergence of $C(n) \rightarrow \ln 2$ for $n \rightarrow \infty$.

before. Analogously to equation 4.24 a maximal memory load $p_{1,max}(\lambda)$ can be computed in terms of hifi-requirements,

$$p_{1,max}(\lambda) = (\epsilon k/n)^{1/(\lambda l)} = {p_{1,max}}^{1/\lambda}. \tag{4.30}$$

From equation 4.25 it is seen that we can store only

$$M_\lambda \approx -\frac{mn}{lk} \ln(1 - p_{1,max}(\lambda)) \tag{4.31}$$

patterns. The fraction of the number of patterns to be stored for perfect address patterns is

$$f_\lambda := \frac{M_\lambda}{M} \approx \frac{\ln(1 - p_{1,max}(\lambda))}{\ln(1 - p_{1,max})}. \tag{4.32}$$

From equations 4.23 and 4.28 the classical memory capacity $C(k,n)$ is reduced to

$$C_\lambda(k,n) := f_\lambda \cdot C(k,n). \tag{4.33}$$

Asymptotically for large n the maximal possible memory capacity is obtained from equations 4.29, 4.32 and 4.33:

$$C_\lambda(k,n) := f_\lambda \cdot C(k,n) \approx \log_2(p_{1,max}) \cdot \ln(1 - p_{1,max}(\lambda)). \tag{4.34}$$

Substituting $p_{1,max} = p_{1,max}(\lambda)^\lambda$ from equation 4.30, we can write $C_\lambda(k,n)$

$$C_\lambda(k,n) \approx \lambda \log_2(p_{1,max}(\lambda)) \cdot \ln(1 - p_{1,max}(\lambda)). \tag{4.35}$$

We get the maximal capacity for $p_{1,max}(\lambda) = 0.5$ corresponding to $p_1 = 0.5^\lambda$, and we can write

$$C_\lambda(k,n) \to \lambda \cdot \ln 2. \tag{4.36}$$

Considering fault tolerance according to parameter λ we can generally write the asymptotic memory capacity

$$C_\lambda(k,n) = \lambda C(k,n). \tag{4.37}$$

Therefore, addressing the memory with a fraction λ of the l ones in the original pattern decreases the classical asymptotic memory capacity by factor λ.

4.3.2 False Ones in the Address Patterns

Comparing the analysis of missing ones in the address pattern to the analysis of false ones shows that the latter is more complicated. Moreover, the Willshaw retrieval strategy, where threshold is equal to the number of active units in the address patterns, can not be applied for the address pattern containing false ones.

For the classical Willshaw model Sommer and Palm [93] have introduced exact formulas for the error probabilities in the retrieved pattern given the error probabilities for numbers of false and missing ones in the address pattern and the threshold. Thus, the threshold can be chosen between l and the number of ones in the address pattern to minimize the error probabilities in the retrieved pattern.

However, if the error probabilities in the address pattern are not known, it is impossible to find an appropriate threshold. Instead, it is necessary to adjust a posteriori threshold repeatedly until the number of ones in the retrieved pattern matches the desired pattern activity.

Even if the error probabilities in the address pattern are known, it is difficult to obtain general results about the impact on number of storable patterns and memory capacity as has been done in section 4.3.1 for missing ones. This is due to the lack of appropriate approximations necessary for general results.

4.4 Discussion

The Willshaw model is a simple implementation of binary neural associative memory. It provides an efficient and useful algorithm in technical applications for rapid and fault-tolerant access to stored pattern information. Its performance can be superior to classical algorithms of computer science, e.g. look-up tables.

The Willshaw model is efficient in terms of memory capacity. For sparse patterns with $k = \log_2 n$ (where n is the length of the binary pattern vectors, and k is number of ones) a high memory capacity of $\ln 2 \approx 0.7$ is possible asymptotically for $n \to \infty$. This means that the Willshaw model can store asymptotically about 0.7 bit of information per synapse. A slightly better memory capacity of $1/2 \ln 2 \approx 0.72$ is obtained for different learning rules and retrieval strategies such as Hebb rule [76]. In 2003 Knoblauch showed that asymptotically memory capacity of 1 is possible for the Willshaw model if the binary memory matrix is optimally compressed [53].

The model is sensitive to false ones in the input patterns. In the case that the input pattern is composed of a superposition of two or more patterns, which are binary vectors of length n containing k ones, at a threshold level $\Theta = k$, the output of the memory is a superposition of patterns corresponding to the patterns the input pattern contains, if the patterns do not overlap. When the patterns overlap, using a maximum threshold strategy may activate some neurons in the output population more strongly than the other output neurons representing the corresponding output patterns.

An important parameter in the model is the pattern activity $k = \sum_{i=1}^{n} y_i$, i.e. the number of one entities in the binary pattern vector of length n. In case the memory stores patterns with different pattern activities, these generally decrease the memory capacity if there are overlaps between patterns, e.g. a pattern with a high pattern activity contains the same neurons that belong to the two patterns with low pattern activities. When a maximum threshold strategy is chosen, a superposition of patterns with both high and low activities is retrieved.

For the memories with fixed memory size, the number M of patterns to be stored in the memory is limited. If we store too many patterns, the memory capacity first increases, but after a limit, it starts to decrease. This leads to an overloading of the memory. A solution to the overloading problem can be to keep the memory size large enough, as to not cause an out-of-memory failure.

Chapter 5

Hybrid HMM/NAM System

A variety of different hybrid hidden Markov model/artificial neural network (HMM/ANN) architectures have been proposed in the literature. The hybrid HMM/ANN approach combines the advantages of both HMMs and ANNs, and often allows for significant improvements in performance in difficult automatic speech recognition (ASR) tasks compared to standard HMMs.

Between the late eighties and the beginning of the nineties ANN architectures attempted to emulate HMMs [71, 14]. The idea underlying the approach was to compute the forward and backward probabilities in HMMs [14]. The model was called *Alpha Net*, because its architecture and dynamics were calibrated to resemble the forward computation of the alphas in the Baum-Welch algorithm. The Alpha net is a recurrent neural network. A recurrent architecture was built for each word to be included in the model. The neurons were organized in order to represent the states of the HMM. Each neuron is connected with a recurrent connection to itself, and with a forward connection to the unit representing the following adjacent state in the HMM. The weights of these connections are equal to the state transition probabilities between the corresponding pairs of states.

In some ANN/HMM hybrids [62, 27, 32, 95], ANNs were used to estimate the HMM state-posterior probabilities. Bourlard et al. [13, 67, 11, 12] proposed HMM/ANN hybrids for continuous ASR. A multi-layer perceptron (MLP) was trained to estimate the posterior probabilities of HMM states, with the objective of maximizing the posterior probability of a given (left-to-right) Markov model given an acoustic observation sequence. Singer and Lippman [91] used radial basis function (RBF) [15] networks instead of MLPs as Bayesian probability estimators and the resulting hybrid was used in an isolated word recognition task.

Robinson et al. [83, 37] extended Bourlard's approach with the introduction of a recurrent network instead of an MLP to estimate state-posteriors. Their system, called ABBOT, is a continuous speech, speaker independent system

for large vocabularies. Hochberg et al. improved a system by a combination of recurrent neural networks, which were trained on different acoustic features, namely MEL and Perceptual Linear Prediction (PLP) coefficients [37]. Backward and forward recurrent networks were also introduced, resulting in four parallel probability estimators.

ANNs were also used to estimate state posteriors [102], but instead of considering only strict values "0" and "1" as output unit values, according to the Viterbi segmentation of training sequences, the output probabilities were generated in the continuous range $(0, 1)$ within the forward-backward mechanism. The training technique was evaluated on a recognition task of continuous digits collected over the telephone channel and the hybrid system achieved less word-error-rate (WER) than standard context-dependent HMMs.

In another hybrid model, the ANN was trained as a feature extractor for a context-dependent HMM in order to transform input acoustic representations into compact low-dimensional representations [8, 5, 6]. These representations are more suitable to be modeled by the emission probabilities of the HMM than standard acoustic parameters. Experimental results showed that training the ANN jointly with the HMM in the proposed hybrid system improved recognition performance over the standard context-dependent HMM [8, 6].

Starting from the late eighties, neural networks were also used as vector quantizers for discrete HMMs. Most of those works (for instance [42, 52]) relied on Kohonen's learning vector quantization (LVQ) [57] as an effective neural alternative to standard clustering algorithms. A new neural architecture was proposed to perform vector quantization on the acoustic features for a discrete HMM [86]. The novelty of the approach involved the training of the ANN, where a feedforward net, basically a 1-layer MLP, was trained with an unsupervised algorithm based on the maximal mutual information criterion.

In 1996 Jang and Un [43] presented a hybrid system for speaker independent isolated word recognition. The system was constituted by a connectionist fuzzy vector quantizer (FVQ), the output of which was fed into a discrete HMM. The proposed FVQ consisted of time-delayed neural networks which were used as phoneme classifiers.

Zavaliagkos et al. [105] developed a hybrid system that combines the advantages of neural networks and HMMs using a multiple hypothesis paradigm and rescores the hypothesis generated by an HMM which uses an N-best strategy. The network computes scores on whole segments of frames, corresponding to phonemes. The connectionist model used for this purpose was called segmental neural network (SNN). Training of the system was accomplished using a segmentation performed by the HMM, possibly considering

the first N-best hypotheses of segmentations, which from the second best down to the Nth best are used as negative examples. Another similar approach for rescoring of a standard HMM with N-best strategy was founded by Moudene et al. [69].

Thomas et al. [25] described the application of associative memory to an isolated speech recognition system. In the associative memory approach patterns are represented with quantized speech vectors (LPC) per frames. The number of processing units in the network is equal to the number of entries in the patterns to be stored. The weights are set with

$$T_{ji} = \sum (2a_{pi} - 1)(2a_{pj} - 1), \tag{5.1}$$

where a_{pi} is the ith entry in speech pattern p and the sum is taken over all speech patterns to be stored. The words are stored using the equation 5.1. Each speech vector is a state of the network corresponding to a minimum value of the network energy. That minimum is expected to correspond to one of the speech patterns of the network.

This chapter is concerned with a new hybrid HMM/NAM approach to speech recognition, which is the basis of this thesis. In the presented approach HMMs are used on the elementary subword-unit level and NAMs are used on the word level. First, the system architecture based on the presented approach is described. Afterwards the conversion of non-binary data strings into sparse binary code vectors, which are used in neural associative memories to represent data strings, is explained. After the first part of the hybrid system "HMM-based subword-unit recognition" is explained, the second part "single word recognition network" is delineated, which is a network of binary neural associative memories. Finally, the presented hybrid system is discussed in terms of computational complexity and fault tolerance.

5.1 System Architecture

An overview of the architecture of the hybrid HMM/NAM system is given in figure 5.1. The information flow in the system is as follows: A spoken sentence is first captured via a microphone and preprocessed to obtain speech vectors. The speech vectors are feature vectors of a raw audio signal, namely Mel Frequency Cepstral Coefficients (13 coefficients are used with 26 delta coefficients) (see section 3.1). The speech vectors enter the HMM module and are transformed into elementary speech components (e.g. phonemes or syllables) by HMMs. As these different types of speech components (subword-units) have different temporal resolutions, it depends on the size of the vocabulary or the amount of the available training data, as to which type of subword-unit is best to use. For example, phonemes could be appropriate for

small vocabulary, whereas large vocabulary may require the usage of demi-syllables or syllables. The HMM output sequence of speech components is then forwarded to the NAM-based single word recognition network. NAMs are used for postprocessing the output of the HMMs and generate matching word hypotheses (output words). The output words can then be further processed, e.g. to extract the semantics of the sentence (i.e., the meaning of the sentence).

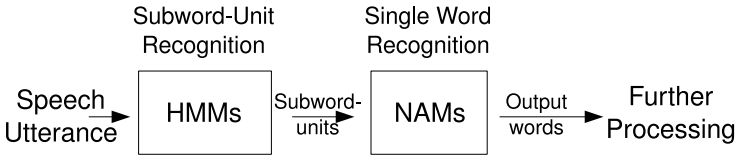

Figure 5.1: Overview of the architecture of the hybrid HMM/NAM system.

The goal of the hybrid system is to take advantage of both HMMs and NAMs in order to improve recognition performance and to generate more flexible and robust recognition systems. When HMMs incorrectly recognize subword-units or can not recognize all correct subword-units, the single word recognition network is able to extract correct words from the HMM output subword-unit stream due to the pattern completion and fault tolerance properties of NAMs.

5.2 Neural Representations

In this section we will focus on binary distributed neural representations of data strings, which is necessary when we deal with neural associative memories. Neural representations are distributed in a trivial sense because the elements, i.e., the neurons, are distributed in the brain. The sparsely distributed neural representation of a pattern means that the ratio p between the number of activated and non-activated neurons in the neuron population satisfies $p \ll 0.5$.

5.2.1 Subword-Unit Representations

The size of the neuron population for subword-unit representations is determined in terms of the number of subword-units which are required for the recognition task. Each neuron in the population is assigned to one subword-unit, i.e., when a subword-unit is present in the memory, only one neuron will be activated and the other neurons remain in an inactive state. In other words, each subword-unit is represented as 1 out of N binary sparse code vectors, where N is the number of distinct subword-units in the recognition

task. Figure 5.2 shows examples of binary sparse neural representations of subword-units.

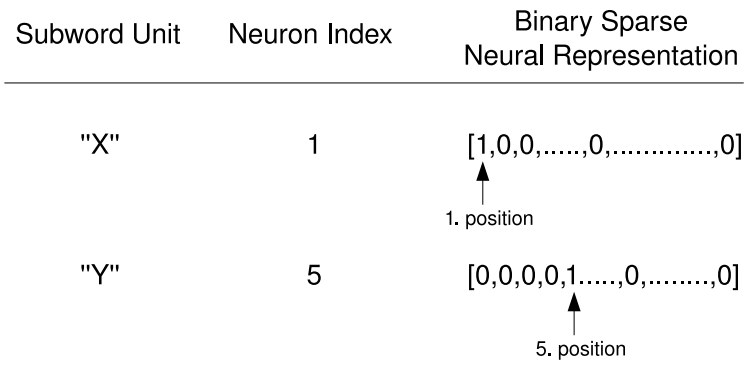

Figure 5.2: Binary sparse neural representations of subword-units. For each subword-unit only the entry in the code vector corresponding to the assigned neuron index is set to 1.

5.2.2 Word Representations

If the representation of a "word" string is defined as the set of neurons that are activated when the "word" string is present in the memory, then more than one neuron will be activated. The words are represented in associative memories using two different sparsely distributed representations.

In the first representation, binary sparse word representations reflect the phonological similarities of words. A natural similarity between different word patterns is given by the number of shared neurons in the underlying neural representations. This means that the word patterns containing the same subword-units share the same neurons which are assigned to these subword-units. The subword-units can be phonemes, demi-syllables or syllables. In figure 5.3 examples are given for phonetic transcriptions of two words "blue" and "black" and their corresponding binary sparse neural representations. In the binary sparse neural representations, the entries are set to 1 and their positions are the same as the neuron indices assigned to the subword-units in the word. The pattern activity is different for each word. This means that each binary neural word representation has a different number of active units because each word is composed of a different number of subword-units.

In the second representation, the words are represented using randomly generated code vectors. For each word a code vector is independently generated (see figure 5.4). In order to get these kinds of sparse binary code vectors we must choose the number of neurons and determine the number of active neurons in terms of the size of the vocabulary so that $p \ll 0.5$ is satisfied.

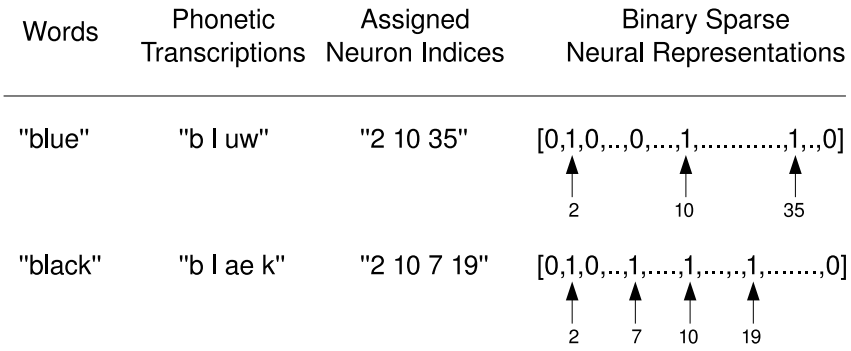

Figure 5.3: Binary sparse neural representations of words.

The advantage of this representation is that each neural word representation has the same number of active neurons (neural activation). When the hybrid system is integrated with an NAM-based module for further processes, such as language understanding, these neural word representations can also be used for the communication between the word recognition module in the proposed hybrid approach and the NAM-based language understanding module in a language processing system. However, this neural representation does not reflect any kind of similarity between words.

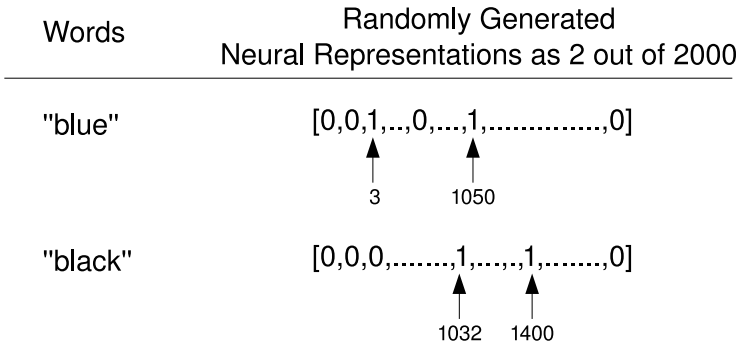

Figure 5.4: Randomly generated binary sparse neural representations of words.

5.3 Subword Unit Recognition

The first part of the presented hybrid system is *HMM-based subword-unit recognition*, where a set of speech vectors is transformed into a sequence of elementary speech components. The output subword-units are speech components such as phonemes, context-dependent phonemes, demi-syllables or syllables. As we said in section 5.1, the type of the speech component used

as output subword-units is determined in terms of the vocabulary size and its recognition accuracy. For small vocabularies a satisfactory subword-unit accuracy can be obtained using phonemes, while for large vocabularies a satisfactory subword-unit accuracy can be achieved using subword-units longer than phonemes, such as demi-syllables or syllables.

For acoustic modeling, context-dependent phonemes (triphones) are usually used because they achieve good recognition performance. The context-dependent phoneme based recognition systems follow a general strategy for acoustic model training. All phoneme models are three-state left-to-right HMMs without skip states (as shown in figure 3.4 in section 3.2). The training procedure essentially involves four steps, as follows:

- Single Gaussian monophone models are created and initialized with the global mean and variance of the training data and trained using reference transcriptions derived from the pronunciation dictionary.

- Context-dependent models that occur in the training corpus are constructed from monophone models. Before building a set of context-dependent models, it is necessary to decide whether or not cross-word triphones are to be used. If they are, then word boundaries in the training data can be ignored and all monophone labels can be converted to triphones. If, however, word internal triphones are to be used, then word boundaries in the training transcriptions must be marked in some way (either by an explicit marker which is subsequently deleted, or by using a short pause). All cross-word or word-internal triphones are created by copying the monophone models for each required triphone context and the transition matrices across all the triphones of each base phone are tied. Then, the models are retrained.

- The states are clustered using two mechanisms. The first is data-driven and uses a similarity measure between states. The second uses decision trees and is based on asking questions about the left and right contexts of each triphone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters (more details for the binary decision tree are given in [103]). The distributions of all the states in each cluster are tied to share data and to be able to make robust parameter estimates. The state-clustered triphones are then retrained.

- Single Gaussian HMMs are converted to multiple mixture component HMMs. The number of mixture components in each state is successively incremented by splitting single Gaussian distributions into mixture distributions until the required number of components is obtained.

Figure 5.5 shows an overview of the recognition process. During the recognition process, a recognition network is compiled using the language model, the pronounciation dictionary and a list of the HMMs to be used. Then it is used to recognize the input utterance.

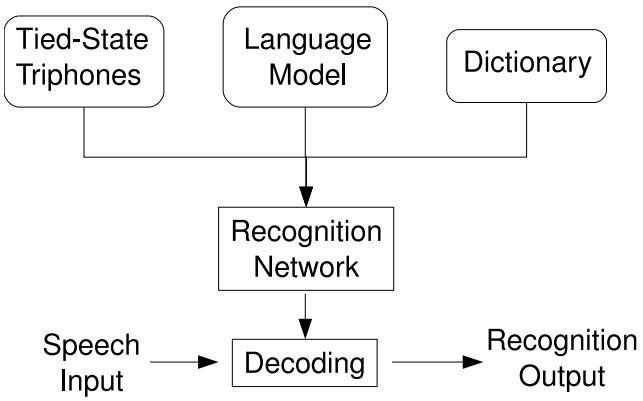

Figure 5.5: Overview of the recognition process.

For a phoneme recognizer, the dictionary contains an entry for each phoneme and the lexicon for the pronunciation might contain

ih	ih
eh	eh
iy	iy
...etc.	

A phoneme-level language model is used to provide estimates of the probabilities of phoneme transitions within the sentences of the phoneme recognition task. The recognition network will expand the simple phoneme loop into a context-dependent phoneme loop to create a cross-word context dependent network. In this case the phoneme network can not be expanded to a word-internal context dependent network.

In the case of a longer subword-unit recognizer such as syllables, the dictionary contains an entry for each subword-unit and the lexicon for the pronunciation contains

b_ay	b ay
k_ao_r	k ao r
d_ih_ng	d ih ng
...etc.	

A subword-unit-level language model is used to provide estimates of the probabilities of subword-unit transitions within the sentences of the recognition task. As in the case of phoneme recognizer, the simple phoneme loop is expanded to a context-dependent phoneme loop to create a cross-word context dependent network.

5.4 Single Word Recognition Network

The second part in the presented hybrid system is a single word recognition network. An overview of the single word recognition network is given below in figure 5.6. Each box in the single word recognition network in figure 5.6 corresponds to an associative memory. The single word recognition network is based on the Willshaw model of binary neural associative memories and consists of 5 associative memories and a representation area SWU. The memories are interconnected with each other via hetero- and autoassociative connections. The memories M1 and M3 are autoassociative memories, while M2, WRD and M4 are heteroassociative memories.

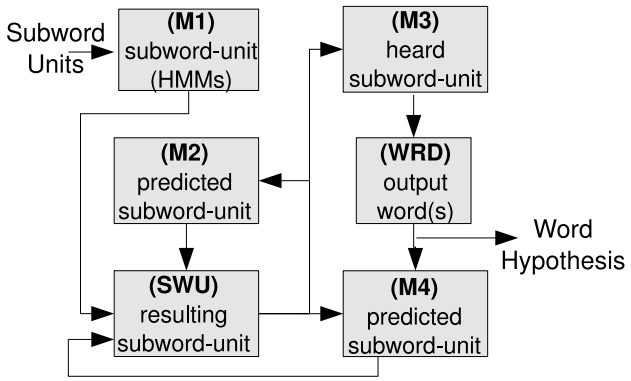

Figure 5.6: Overview of the single word recognition network.

The basic idea is that the single word recognition network generates a list of word hypotheses in terms of the subword-units processed each time a new subword-unit is read from the HMM output sequence. After processing all subword-units that belong to a possible word, a word hypothesis is activated.

For isolated word recognition tasks it is usually easy to determine word boundaries because there is a small pause between words which is used to detect the word boundaries. Figure 5.7 shows an example command sentence and its subword-unit (phoneme-level) transcription. In figure 5.7 the small pause "sp" denotes the word boundaries.

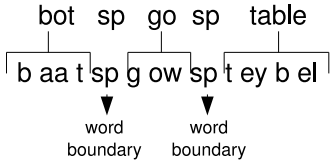

Figure 5.7: An example command sentence and its subword-unit (phoneme-level) transcription where the small pause "sp" denotes the word boundaries.

For continuous speech recognition tasks, it is usually difficult to determine word boundaries, because there is no boundary such as a small pause between words. Therefore, we assume that the word boundary is detected when there is no transition between the current and the previously recognized subword-units. Figure 5.8 gives an example sentence from the Wall Street Journal (WSJ1) for continuous speech recognition.

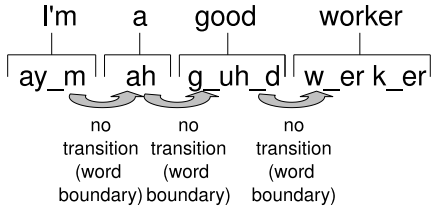

Figure 5.8: An example command sentence and its subword-unit (syllable-level) transcription.

In the single word recognition network, subword-unit and word patterns are stored in each memory as shown in the following section.

5.4.1 Pattern Storage

Memories M1 and M3

The memories M1 and M3 are autoassociative memory matrices of dimension $n \times n$ where n is the number of distinct subword-units in the task vocabulary. They store subword-unit patterns in columns using 1 out of n sparse binary code vectors (see section 5.2). Figure 5.9 shows the storage of two subword-unit patterns in the memory matrix M1. In figure 5.9 the subword-unit "X" is stored in the matrix entry $(1,1)$ because the first neurons in the input and output neuron populations are assigned to the subword-unit "X", and the subword-unit "Y" is stored in the matrix entry $(5,5)$ in the same way.

The memory load p_1 of M1 is computed using equation 4.8. Given pattern size n, the memory load p_1 is obtained as $1 - e^{-1/n}$. Figure 5.10 shows the dependence of the memory load p_1 on the pattern size n.
Totally stored information is computed using equation 4.23 as

$$C(k,n,M) = C(1,n,n) = nlog_2 n \tag{5.2}$$

and the stored information per synapse is obtained as

$$C(1,n,n)/n^2 = \frac{log_2 n}{n}. \tag{5.3}$$

Figure 5.10 shows also the stored information per synapse as the pattern size n increases.

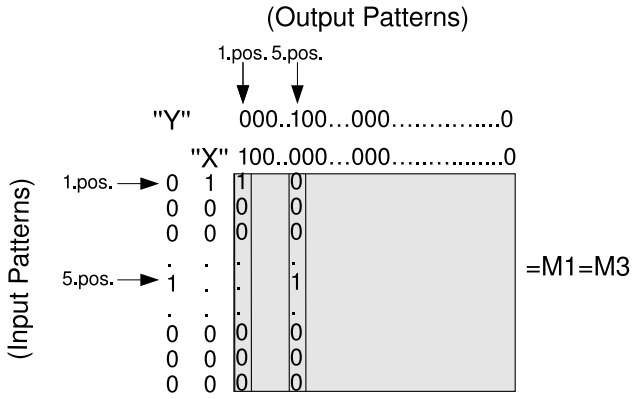

Figure 5.9: Storage in the memory matrices M1 and M3.

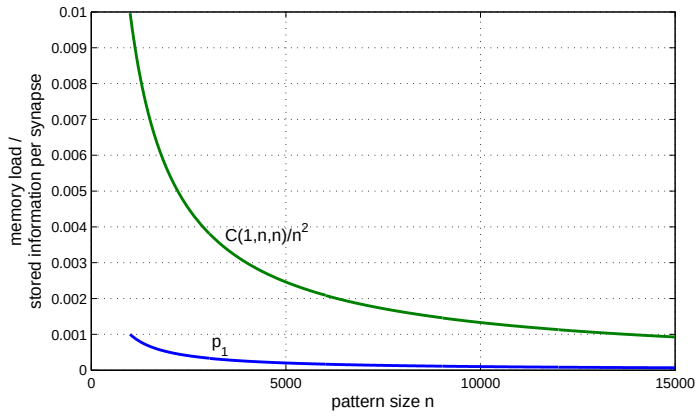

Figure 5.10: The dependence of the memory load p_1 and the stored information per synapse on the pattern size n in the memories M1 and M3. As the pattern size n, which is also the number of patterns to be stored, increases, the memory load p_1 and the stored information per synapse decreases monotonically. But the stored information per synapse decreases faster than the memory load p_1.

Memory M2

The memory M2 is a heteroassociative memory matrix of dimension $n \times n$. M2 stores subword-unit transitions within the words in the vocabulary using 1 out of n sparse binary code vectors. To store subword-unit transitions in the memory, subword-unit level transcriptions of the words are necessary. For example, given the subword-unit (e.g. phoneme) level transcription "b l uw" of the word "blue", the subword-unit transitions within the word "blue", "b→l" and "l→uw", are stored as shown in figure 5.11. For the transition "b→l" the input pattern is the 1 out of n binary code vector of the subword-

unit "b" and the output pattern is the 1 out of n binary code vector of the subword-unit "l". In the same way, for the transition "l→uw" the input pattern is the 1 out of n binary code vector of the subword-unit "l" and the output pattern is the 1 out of n binary code vector of the subword-unit "uw".

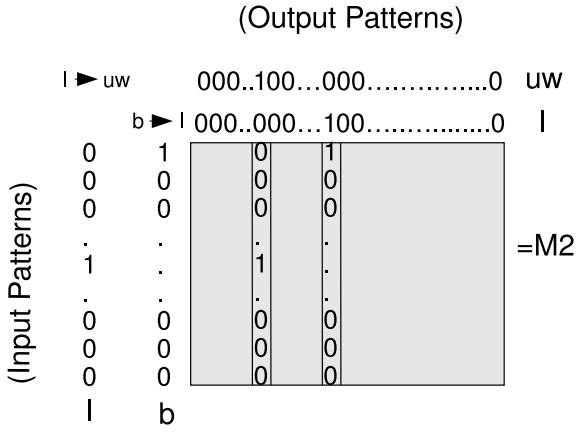

Figure 5.11: Storage of the subword-unit transitions "b→l" and "l→uw" in the memory matrix M2.

Given pattern size n and the number of subword-unit transitions to be stored M, the memory load p_1 is computed using equation 4.8 as

$$p_1 = 1 - e^{-M/n^2}. \tag{5.4}$$

From equation 4.23 totally stored information is computed as

$$C(k, n, M) = C(1, n, M) = Mlog_2 n, \tag{5.5}$$

and the stored information per synapse is obtained as

$$C(1, n, M)/n^2 = \frac{Mlog_2 n}{n^2}. \tag{5.6}$$

Figure 5.12 shows the dependence of the memory load p_1 and the stored information per synapse on the pattern size n for a given $M = 10000$. Figure 5.13 shows the memory load p_1 and the stored information per synapse for a given pattern size n as the number of patterns M increases.

Memory WRD

The memory WRD is a heteroassociative memory matrix of dimension $n \times r$, where r is the size of randomly generated word code vectors. WRD stores the

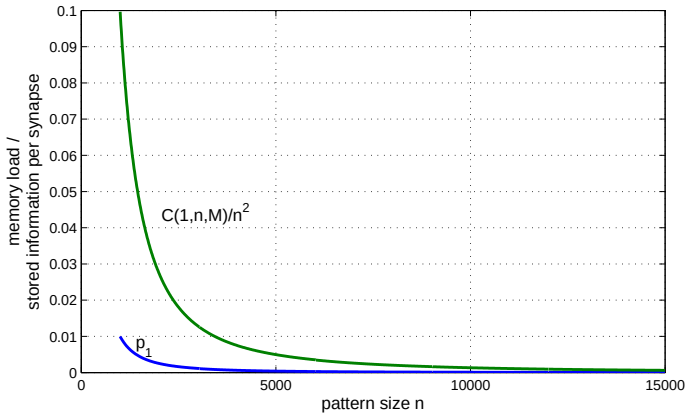

Figure 5.12: The dependence of the memory load p_1 and the stored information per synapse on the pattern size n for a given $M = 10000$ in the memory M2. As the pattern size n increases, the memory load p_1 and the stored information per synapse decreases monotonically. But the stored information per synapse decreases faster than the memory load p_1.

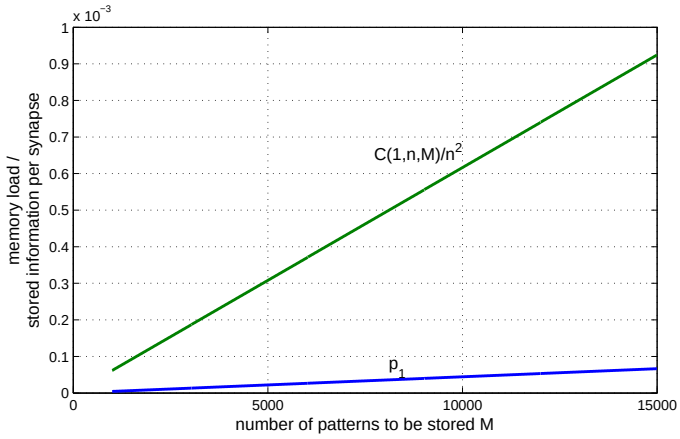

Figure 5.13: The dependence of the memory load p_1 and the stored information per synapse on the number of patterns M for a given pattern size $n = 10000$ in the memory M2. As the number of patterns M increases, the memory load p_1 and the stored information per synapse increases. But the memory load p_1 increases more slowly than the stored information per synapse.

word patterns in the vocabulary in a distributed way using two representations, namely the binary sparse neural representations reflecting the phonological similarities of words as input patterns and the randomly generated representations as output patterns given in section 5.2. Figure 5.14 shows the storage of the word pattern "blue" in the memory matrix WRD.

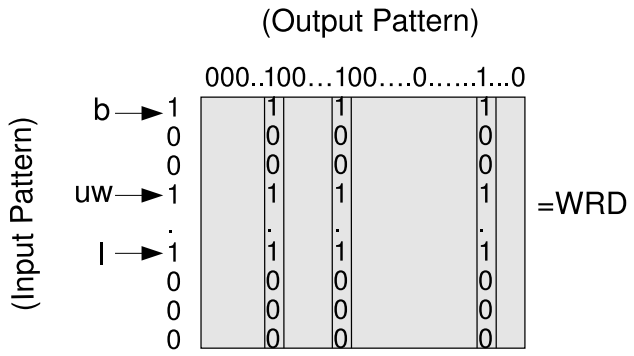

Figure 5.14: The memory WRD after the word "blue" has been stored. For the word "blue" the input pattern is the binary sparse word representation based on the phonetic transcription and the output pattern is the randomly generated representation where the pattern activity is equal to 3.

The output pattern activity k (number of one-entries in the pattern) is the same for each pattern, while the input pattern activity l is different depending on the length of the words. The memory load p_1 is computed using equation 4.8 as

$$p_1 = 1 - e^{-\sum_{i=1}^{M} \frac{l_i k}{nr}} \tag{5.7}$$

$$\approx 1 - e^{-\frac{M\bar{l}k}{nr}}, \tag{5.8}$$

where $\bar{l}$ is the average input pattern activity.

The input pattern size n is determined in terms of the number of distinct subword-units which are seen in the vocabulary. Totally stored information is computed using equation 4.23 as

$$C(k, r, M) = Mklog_2(r/k), \tag{5.9}$$

and the stored information per synapse is obtained as

$$C(k, r, M)/(nr) = \frac{Mklog_2(r/k)}{nr}. \tag{5.10}$$

Figure 5.15 shows the dependence of the memory load p_1 and the stored information per synapse on the output pattern size r for a given $M = 10000$, an average $\bar{l} = 4$, $k = 2$, and $n = 8000$. Figure 5.16 depicts the dependence of the memory load p_1 on the number of patterns M for a given $n = 10000$, $r = 2000$, $k = 2$, and an average $\bar{l} = 4$. Figure 5.17 shows the memory load p_1 as the average $\bar{l}$ increases for a given $M = 5000$, $n = 10000$, $k = 2$, and $r = 2000$.

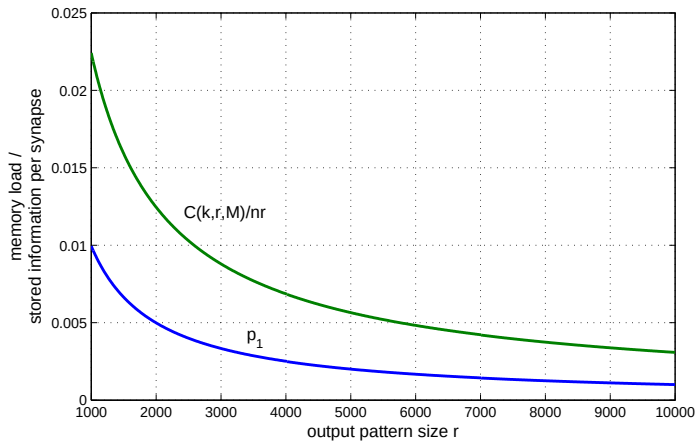

Figure 5.15: The dependence of the memory load p_1 and the stored information per synapse on the output pattern size r in the memory WRD for a given $M = 10000$, an average $\bar{l} = 4$, $k = 2$, and $n = 8000$. As the output pattern size r increases, the memory load p_1 and the stored information per synapse decreases monotonically. But the stored information per synapse decreases faster than the memory load p_1.

Memory M4

The memory M4 is a heteroassociative memory matrix of dimension $r \times n$, which is the transposed form of the memory WRD. M4 also stores the word patterns in the vocabulary in a distributed way using two representations. The randomly generated representation is used as the input pattern and the binary sparse word representation reflecting the phonological similarities of words is used as the output pattern. Figure 5.18 shows the storage of the word pattern "blue" in the memory matrix M4.

The output pattern size is determined in terms of the number of distinct subword-units which are seen in the vocabulary. The memory load p_1 is computed in the same way as in WRD (equation 5.8). Totally stored information is computed using equation 4.23 as

$$C(l_i, n, M) = \sum_{i=1}^{M} l_i log_2(n/l_i) \tag{5.11}$$

$$C(\bar{l}, n, M) = M\bar{l} log_2(n/\bar{l}), \tag{5.12}$$

and the stored information per synapse is obtained as

$$C(\bar{l}, n, M)/(nr), \tag{5.13}$$

where $\bar{l}$ is the average input pattern activity.

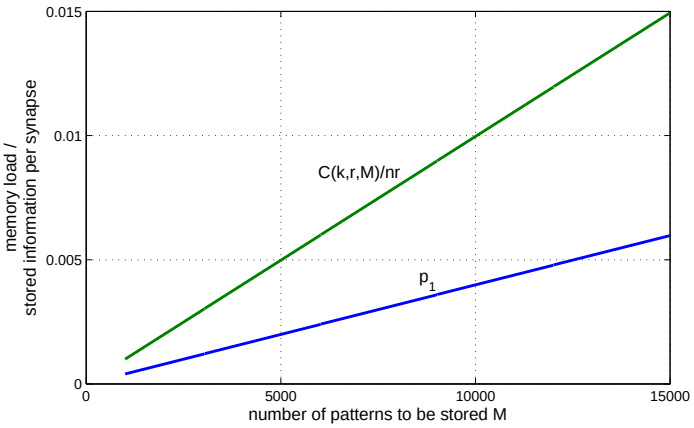

Figure 5.16: The dependence of the memory load p_1 and the stored information per synapse on the number of patterns M in the memory WRD for a given $n = 10000$, $r = 2000$, $k = 2$, and an average $\bar{l} = 4$. As the number of patterns M increases, the memory load p_1 and the stored information per synapse increases. But the memory load p_1 increases more slowly than the stored information per synapse.

Figure 5.19 shows the memory load p_1 and the stored information per synapse as the output pattern size n increases for a given number of patterns to be stored $M = 10000$, pattern size $r = 2000$, and input pattern activity $k = 2$ and output pattern activity average $\bar{l} = 4$.

Figure 5.20 shows the memory load p_1 and the stored information per synapse as the number of patterns to be stored M increases for given pattern sizes $n = 10000$, $r = 2000$, and input pattern activities $k = 2$ and output pattern activity average $\bar{l} = 4$.

5.4.2 Retrieval

During the retrieval of words from the HMM output sequence, each subword-unit in the sequence is transformed into its corresponding binary sparse neural representation (see section 5.2) before it is applied to the word recognition network. During retrieval each memory has a special task.

Memory M1

The memory M1 serves as an input area and presents the HMM output subword-unit to the network. After applying the binary sparse code vector that corresponds to the HMM output subword-unit, the memory M1 retrieves the same subword-unit pattern as the output pattern (the HMM output subword-

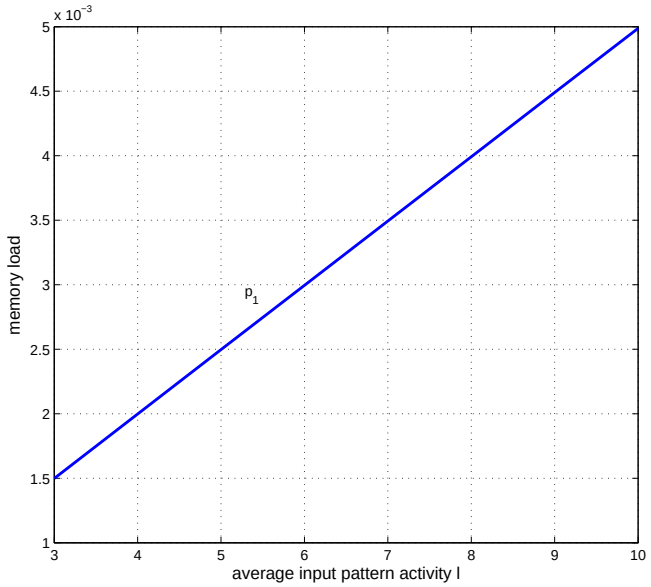

Figure 5.17: The dependence of the memory load p_1 on the average $\bar{l}$ in the memory WRD for a given $M = 5000$, $n = 10000$, $k = 2$, and $r = 2000$. The memory load p_1 increases as the average $\bar{l}$ increases.

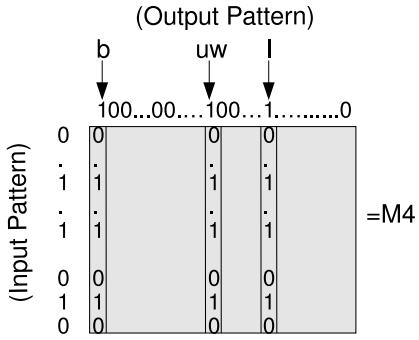

Figure 5.18: The memory M4 after the word "blue" has been stored. For the word "blue" the input pattern is the randomly generated representation where the pattern activity is equal to 3 and the output pattern is the binary sparse word representation based on the phonetic transcription.

unit) at the threshold level $\Theta = 1$. Figure 5.21 shows the representation of the HMM output subword-unit to the network via the memory M1.

Memory M2

As mentioned in the previous section, the memory M2 stores the subword-unit transitions within the words. During retrieval it uses this information

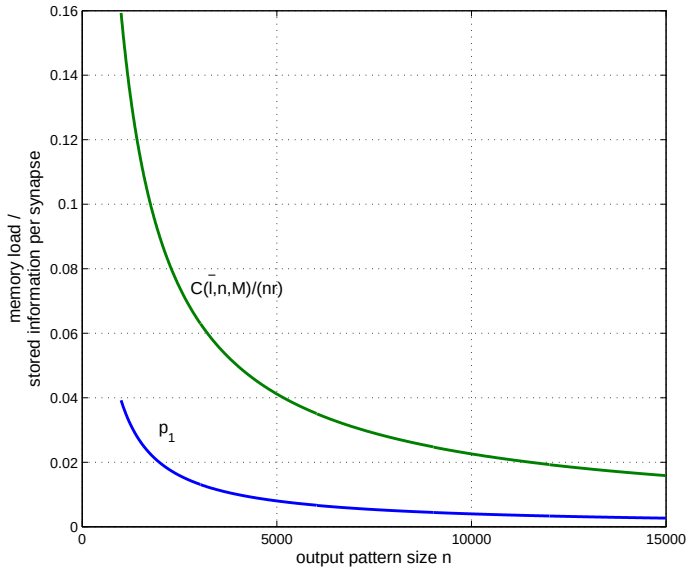

Figure 5.19: The dependence of the memory load p_1 and the stored information per synapse on the output pattern size n in the memory M4 for a given $M = 10000$, an average $\bar{l} = 4$, $k = 2$, and $r = 2000$. As the pattern size n increases, the memory load p_1 and the stored information per synapse decreases monotonically. But the stored information per synapse decreases faster than the memory load p_1.

to predict the possible subword-units which follow the input subword-unit. Figure 5.22 shows an example of the retrieval of the subword-unit (phoneme) "l" in M2 given the subword-unit (phoneme) "b" as input pattern.

Given the input subword-unit (phoneme) "b", the memory M2 activates the output subword-unit (phoneme) "l" which follows the input subword-unit "b". This means that only the subword-unit "l" follows the subword-unit "b" in the vocabulary.

Memory M4

In the memory M4 each output unit (neuron) is assigned to a subword-unit. The memory M4 takes two inputs: the word pattern as a randomly generated neural representation and subword-units that activate the corresponding output subword-units as a superposition of 1 out of n binary neural representations. During retrieval, the memory M4 predicts the possible subword-units among the activated output subword-units with respect to the input word pattern. Figure 5.23 shows an example of the prediction of the output subword-unit given the word pattern "blue" and the two input subword-unit

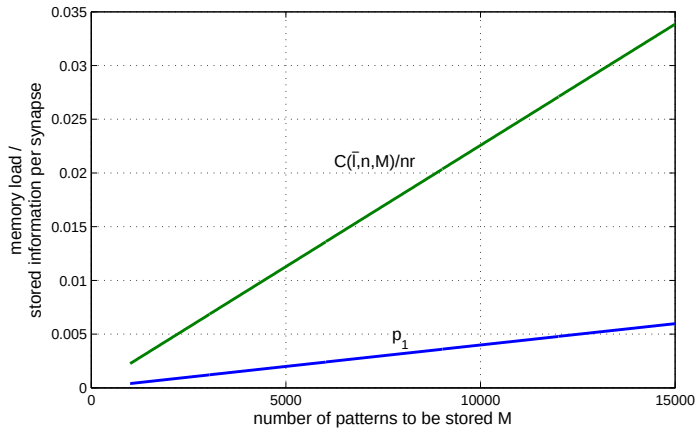

Figure 5.20: The dependence memory load p_1 and the stored information per synapse on the number of patterns to be stored M in the memory M4 for given $n = 10000$, $r = 2000$, $k = 2$ and, average $\bar{l} = 4$. As the number of patterns M increases, the memory load p_1 and the stored information per synapse increases. But the memory load p_1 increases more slowly than the stored information per synapse.

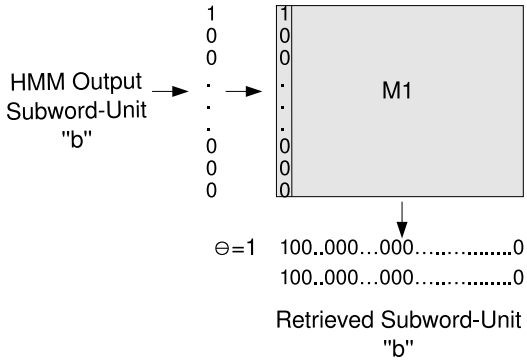

Figure 5.21: Representation of the HMM output subword-unit to the network via the memory M1.

patterns “k” and “l”.

Given the input word pattern “blue” and subword-unit (phoneme) patterns “k” and “l”, the output subword-unit “l” is predicted. The threshold value Θ is set to a value of 3 because the number of one-entities in the binary randomly generated neural representations of the words is equal to 3. At the threshold level $\Theta = 3$ only the subword-unit “l” is activated.

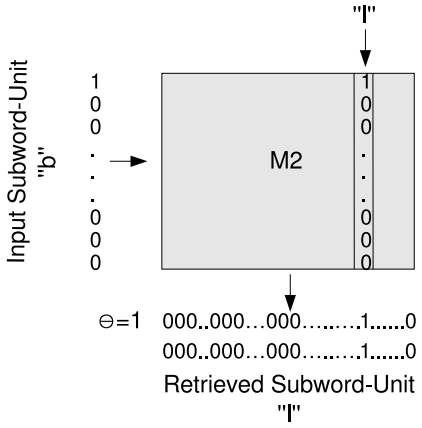

Figure 5.22: Retrieval in the memory M2.

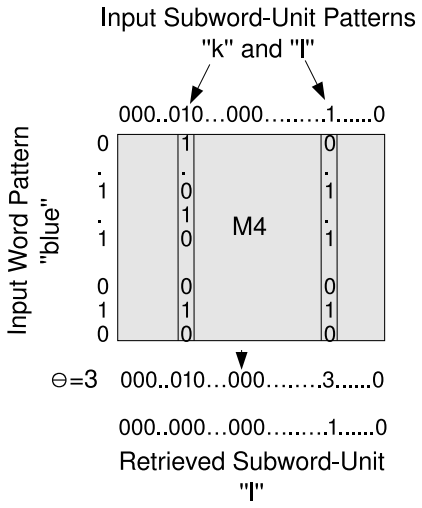

Figure 5.23: Retrieval in the memory M4.

Representation Area SWU

The weighted outputs of the memories M1, M2 and M4 are summed as follows:

$$d_i = c \cdot {d_i}^{M1} + \frac{c}{2} \cdot {d_i}^{M2} + \frac{c}{2} \cdot {d_i}^{M4}, \quad 1 \leq i \leq n, \tag{5.14}$$

where ${d_i}^{Mj}$ is the output unit i of the memory Mj, $j = 1, 2, 4$ and c is the weight coefficient. Then, a common threshold is applied:

$$SWU(i) = \begin{cases} d_i \geq \Theta_{SWU}, & 1 \\ d_i < \Theta_{SWU}, & 0 \end{cases}, \tag{5.15}$$

where $SWU(i)$ is the ith output unit in the memory SWU and $\Theta_{SWU} = c$ is the threshold value in SWU.

The weight of the memory M1 is equal to the summed weight of the memories M2 and M4. In this way, the HMM output subword-unit has the same weight as that of the predicted subword-units. The resulting subword-unit is represented in the area SWU.

Memory M3

The memory M3 holds the processed subword-units in SWU. Figure 5.24 (a) shows the activation of the subword-unit pattern "l". In figure 5.24 (b), another subword-unit pattern "uw" is applied to the memory M3 which holds the subword-unit pattern "l". The subword-unit pattern held in M3 (e.g. subword-unit pattern "l") is back-propagated to the memory M3 as the new subword-unit pattern is applied to M3. In this way, the subword-unit pattern already held in M3 is not removed. Therefore, both subword-unit patterns are held in M3.

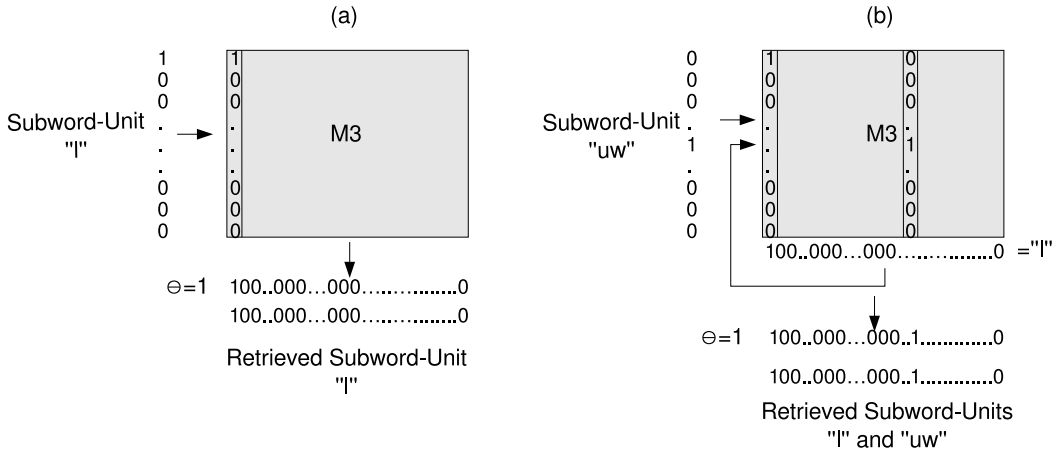

Figure 5.24: Retrieval in the memory M3.

Memory WRD

The memory WRD generates word hypotheses with respect to the superposition of 1 out of n binary neural representations of subword-units. Figure 5.25 shows the retrieval process in WRD. A superposition of two subword-unit patterns "b" and "l" are given as the input pattern, and the retrieval result is the corresponding randomly generated neural representation of the word "blue" as the global threshold $\Theta = 2$.

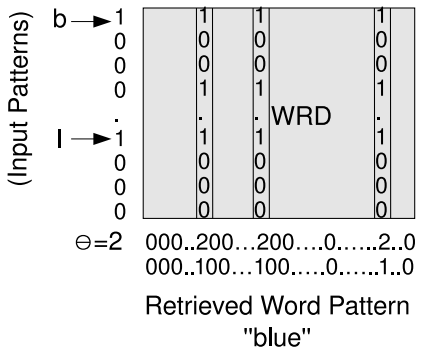

Figure 5.25: Retrieval in the memory WRD.

5.4.3 Functionality

In order to simplify the explanation of the retrieval, a "global time step" is introduced. In one global time step, each memory performs a pattern retrieval and the results are forwarded to subsequent memories. All memories work in parallel. A retrieval process in one global time step is composed of 5 steps:

- The HMM output subword-unit is presented to the memory M1 (figure 5.26).

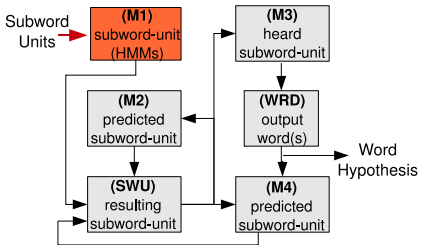

Figure 5.26: Representation of the HMM subword-unit to the memory M1.

- The memory M2 takes the input subword-unit from SWU via an autoassociative connection and represents the possible subword-units which follow the resulting subword-unit in SWU in the previous global retrieval step (figure 5.27).

- The memory M4 represents the expected subword-unit in the current global time step with respect to the word pattern generated in WRD and the resulting subword-unit from SWU in the previous global time step. The memory M4 takes two inputs, namely the word pattern from WRD via an autoassociative connection and the subword-unit patterns from

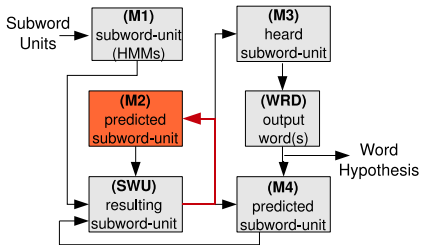

Figure 5.27: Prediction of the subword-units which follow the subword-unit in SWU in the previous global retrieval step.

SWU via a heteroassociative connection, which activates the subword-unit patterns that follow the resulting subword-unit pattern in SWU (figure 5.28).

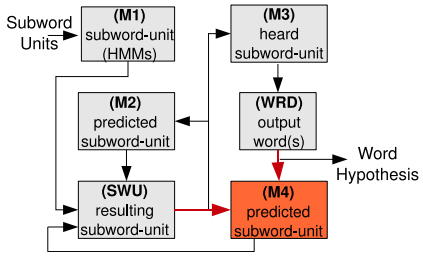

Figure 5.28: Representation of the expected subword-unit in the current global time step with respect to the word pattern generated in WRD and the resulting subword-unit from SWU in the previous global time step.

However, at the beginning of each word, the memories M2 and M4 do not represent any subword-units, due to the fact that no expectation can be generated in the beginning of the word recognition process.

- The weighted outputs of the memories M1, M2 and M4 are summed and a global threshold is then applied. Therefore, the spurious subword-units (incorrectly recognized by HMMs), which can cause ambiguities on the word level, may be corrected by the network. The resulting subword-unit is represented in the area SWU (figure 5.29).

- The memory M3 holds the resulting subword-unit in SWU at the current global time step without removing the subword-units processed up until the current global time step (figure 5.30).

- The subword-units in the memory M3 are then forwarded to the memory WRD. The memory WRD generates a word hypothesis or superposition of word hypotheses with respect to the subword-units activated in M3 (figure 5.31).

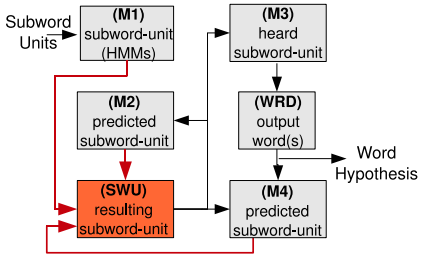

Figure 5.29: Representation of the resulting subword-unit in SWU.

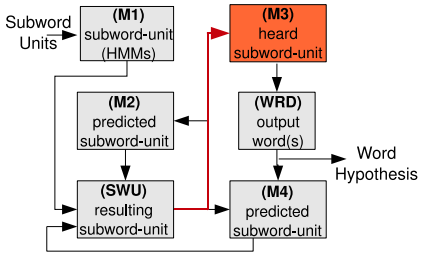

Figure 5.30: Storage of the resulting subword-unit (in SWU) in M3.

When a word boundary is detected, the iterations for the current word end, and the memories in the network are set to their initial states in order to recognize the next word in the sentence.

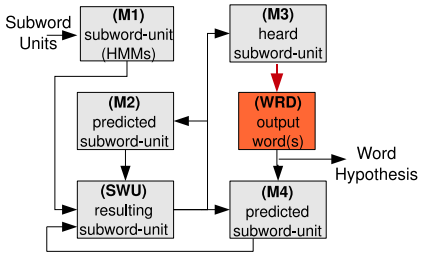

Figure 5.31: Generation of a word hypothesis or superposition of word hypotheses with respect to the subword-units activated in M3.

5.5 An Example of the Network Functionality

Given a part of a subword unit sequence generated by HMMs, e.g. “jh_ah p_ae_n p_l_ey_z ...”, which means “japan plays ...”, the single word recognition network processes it as follows:

Figure 5.32 shows the state of the single word recognition network after the first syllable “jh_ah” in the HMM output sequence has been processed. M1 represents the first syllable received from the HMM output at the first global

time step, while M2 and M4 do not represent any syllables because they do not receive any input at the beginning of the word recognition process. Therefore, SWU represents the same syllable and it is forwarded to M3. The syllable in M3 does not allow for a unique word interpretation because there are many words in the vocabulary which contain the syllable "jh_ah" and thus a list (superposition) of all matching word patterns (with the highest activation) is finally displayed in WRD. Note that the additional calculation of the list of all word pattern names is only held for display and only in the WRD memory. The number of pattern names displayed in the memories is limited to 5 pattern names.

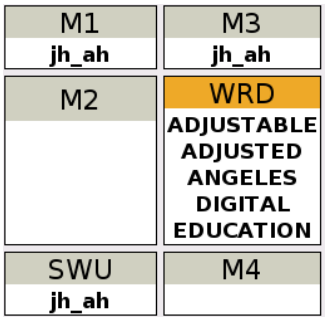

Figure 5.32: Processing of the first syllable "jh_ah" in the network.

As shown in figure 5.33, in the next global time step, M1 represents the HMM output "p_ae_n", M2 represents the expected syllable with respect to the syllable represented in SWU in figure 5.32 (in the previous global time step) and M4 represents the predicted syllables at the current global time step with respect to the word hypotheses represented in WRD and the syllable represented in SWU in figure 5.32 (in the previous global time step). After the summing of the outputs of the memories M1, M2 and M4 and applying a global threshold strategy, the resulting syllable is shown in SWU (figure 5.33).

The resulting syllable in SWU is then forwarded to M3 and stored in this memory. The final process at the current global time step is that M5 generates a word hypothesis that matches the set of the syllables in M3 with the highest probability.

In figure 5.35 the network processes the next syllable in the HMM output. As shown in figure 5.35, M4 can not make an estimation about the current syllable in the HMM output with respect to the outputs of WRD and SWU. Therefore, there is not an activated pattern in M4. But M2 generates a syllable, which is expected at the current global time step, in terms of the syllable in SWU. Because the sum of the outputs of M1, M2 and M4 can not exceed the threshold Θ_{SWU} (see section 5.4.2), the network decides on a word boundary at the current global time step and the syllable in M1 is handled again as a first syllable of a new word.

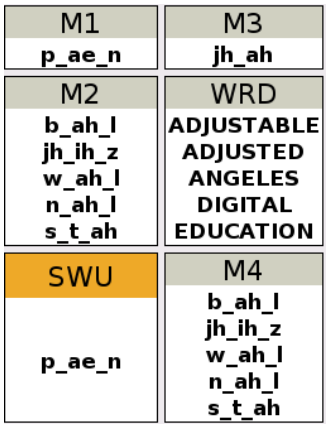

Figure 5.33: Processing of the second syllable "p_ae_n" in the network.

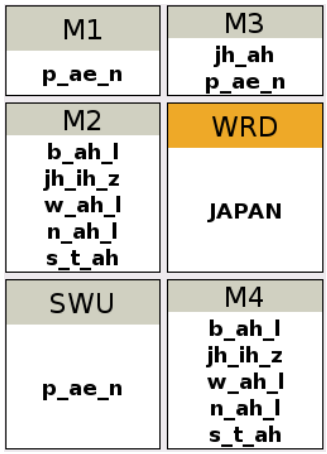

Figure 5.34: Generating of a word hypothesis with respect to the syllables held in M3.

5.6 Ambiguities

ASR has different kinds of ambiguities, pronunciation variability, word boundary ambiguity and homophones. The *pronunciation variability* occurs when the speaker's pronunciation of a word is different from its canonical pronunciation. The *word boundary ambiguity* occurs when there are multiple ways of grouping phones into words, e.g., "It's not easy to wreck a nice beach" and "It's not easy to recognize speech". The concept *homophones* refers to unrelated words that sound the same, but have different orthography, e.g., sail and sale. In addition to these ambiguities, background noise and insufficient training data can also decrease the recognition performance of HMMs.

These ambiguities cause HMMs to recognize (additional) wrong subword-units or to not recognize all correct subword-units. But the neural associa-

Figure 5.35: Detecting word boundaries in the network.

tive networks are able to handle ambiguities of subword-unit level transcriptions since they are fault tolerant. A superposition of words is activated if the address pattern (the subword-units activated in the memory M3) does not strongly address one unique word pattern, but addresses several weaker word patterns. Ambiguities on the word level are represented by a superposition of many items (words) in question, e.g., if the HMM-based subword-unit recognizer generates a subword-unit sequence like "ao l" for the word "ball", whose correct transcription is "b ao l", the word recognition network can not make a unique decision on the correct word. It generates a superposition of the words "ball" and "wall" because "wall" has the phonetic transcription "w ao l" and the two words share the same subword-units "ao" and "l". Another example is given from the Wall Street Journal (WSJ1): the words "writes" and "rights" are homophones, i.e., they have the same canonical transcription, "r ay t s". Therefore, during retrieval for a given subword-unit sequence like "r ay t s", the word recognition network does not decide on a single word and produces a superposition of the words "writes" and "rights".

The ambiguity on the word level might be resolved in further processes (e.g. on the language or sentence levels) if additional context information (semantics) or a priori information on the word sequences to be recognized supports one of the probabilities (see section 6), and weakens the others. In the neural network, this means that additional input must precisely support one pattern of the superposition in order to resolve the ambiguity. Contextual input can help to resolve ambiguities, e.g., using a bidirectional connection between the predicate of a sentence and the object. Syntactic input can also help to resolve ambiguities using the information about the way words are put together to form sentences.

5.7 Discussion

The presented hybrid HMM/NAM approach takes advantage of both HMMs and NAMs in order to develop a flexible and robust recognition system. In real-world environments, where the performance of HMMs is negatively affected, NAMs, combined with HMMs, can increase the recognition performance due to the "fault-tolerance" property of NAMs. A subword-unit is represented in NAMs using a binary sparse code vector. A word is represented as a superposition of binary sparse code vectors which belong to the subword-units involved in the word. This representation enables the system to preserve the similarity between words, e.g., homophones have the same binary sparse code vector. The subword-unit and word patterns are stored in NAMs using the Hebbian learning rule. For retrieval, two different retrieval strategies are employed: One step retrieval strategy, where the threshold is set to a global value, and Willshaw's retrieval strategy, where the threshold is set to the number of ones in the binary input vector.

In terms of the memory capacity analysis it is also seen that the stored information per synapse and the memory load in NAMs in the hybrid system decreases monotonically as the output pattern size increases. The reason the stored information per synapse is low is that the input and output patterns are very sparse.

The memory usage of the memories in the single word recognition network depends on the size of the subword-units and the words in the task vocabulary. In other words, in order to add new subword-unit or word patterns to the memories, their memory size should be increased because subword-unit patterns are represented as 1 out of n code vectors and words are superpositions of subword units. The size of the subword-units is determined by the type of the subword-unit, and the type of the subword-units is determined with respect to the size of the task vocabulary. For small vocabularies, it is usually suitable to use phonemes as output subword-units. But, if the size of the vocabulary is large, then subword-units longer than phonemes, such as demi-syllables or syllables, enable HMMs to give better recognition performance.

For the words that are mispronounced by a speaker, it is usually possible for HMMs to generate wrong subword-units. The network of NAMs is able to handle ambiguities that occur because of the recognized additional wrong subword-units and the subword-units that can not be recognized by HMMs. If the ambiguity on the subword-unit level can not be solved, the network of NAMs then represents the ambiguity on the word level as a superposition of all possible words (see section 5.6). In the case of homophones, which refers to words that sound the same, but have different orthography, NAMs (the memory WRD) can not decide on a single word. Therefore, a superposition

of homophones is retrieved. In order to solve the ambiguity on the word level, it is necessary to make use of additional information, such as a priori information on the word sequences to be recognized, contextual information, or syntactic information.

Because of the sparse representation of subword-units and words in NAMs, the computational cost in NAMs is only limited to active input units. Due to the high storage capacities of the sparse binary associative memories [73], the presented approach scales well with large vocabularies. For large vocabulary speech recognition tasks, the presented hybrid system utilizes a task vocabulary of subword-units which is usually smaller than that of the words. In spite of this, it takes a bit more time on the HMM level to search for the most appropriate subword-unit sequence for a given speech utterance because of the complexity of the subword-unit search space in the HMM recognition network.

The hybrid HMM/NAM approach also facilitates a link between symbolic information and sub-symbolic information processing. The HMM-based subword-unit recognition is performed using speech vectors to represent sub-symbolic information, whereas the NAM-based word recognition generates symbolic information from sub-symbolic information. This property is very useful for neuro-symbolic integration.

Chapter 6

Extension of the Hybrid System

A language model plays an important role in speech recognition, especially in large vocabulary continuous speech recognition, to improve the recognition performance. Thus the single word recognition network in the presented hybrid system (see section 5.4) is extended with a neural associative memory (NAM) based language model network, where the word hypotheses generated in the memory WRD are processed by a network of NAMs which holds the language modeling information to recognize a sequence of the output words within the spoken sentence. This section deals with the language model network which is a network of NAMs. First the architecture of the language model network is described. Then the conversion of words, word bigrams, and trigrams into sparse binary code vectors is explained. Afterwards the "language model network" is introduced and an example of the functionality of the extended hybrid system is given.

6.1 Architecture of the Language Model Network

An overview of the NAM-based language model network within the extended model is shown on the right of figure 6.1. In the hybrid hidden Markov model (HMM)/NAM system, HMMs are sometimes unable to discern every subword-unit uttered by a speaker. Because of the wrongly recognized or missing subword-units, the NAM-based word recognition network can generate wrong words or superpositions of several words. But, the language model network takes advantage of a priori information on the word sequences to be recognized to correct wrongly recognized words.

The language model network is based on the Willshaw model of binary neural associative memories as well as the word recognition network and consists of one autoassociative memory M5 and two heteroassociative memories BGW and SEN. The word recognition and language model networks are interconnected via a heteroassociative connection from the memory WRD in

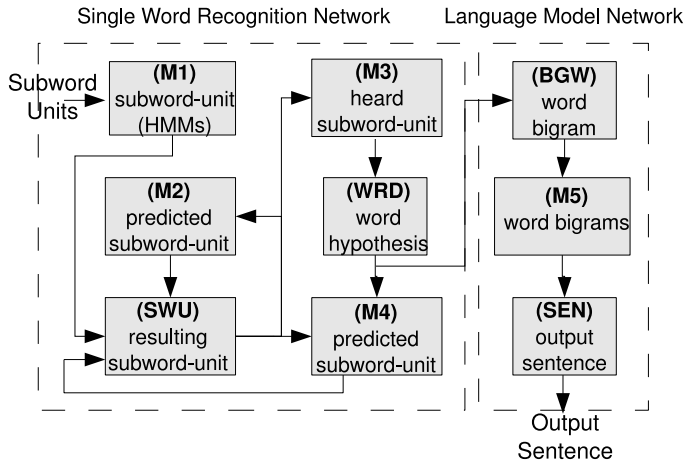

Figure 6.1: Overview of the language model network (on the right).

the word recognition network to the memory BGW in the language model network.

6.2 Neural Representations

6.2.1 Word Representation

In the language model network the words are represented using binary sparse neural representations as 1 out of V code vectors, where V is the total number of words in the task vocabulary. The size of the neuron population for this word representation is determined in terms of the number of words in the task vocabulary, and each neuron in the population is assigned to one word. Figure 6.2 shows examples of binary sparse neural representations of words.

6.2.2 Word Bigram and Trigram Representations

The word bigrams and trigrams are represented using binary sparse neural representations as 1 out of B and 1 out of T code vectors, respectively, where B is the number of word bigrams required for the recognition task and T is the number of word trigrams required for the recognition task. The sizes of the neuron populations for the word bigrams and trigrams are dependent on the number of the word bigrams and trigrams that are required for the recognition task. Figure 6.3 shows examples of binary sparse neural representations of the word bigrams and trigrams.

These binary sparse neural representations are used to represent the word sequences within the sentences in associative memories. The binary sparse

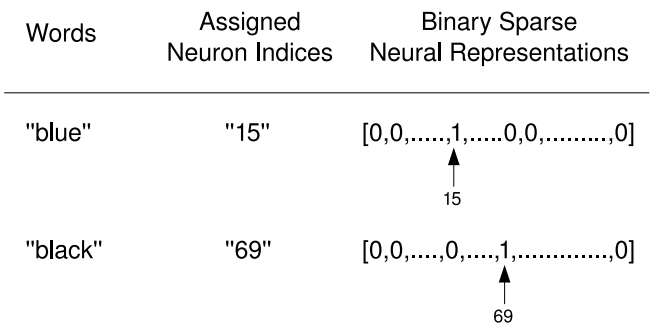

Figure 6.2: Binary sparse neural representations of words as 1 out of V code vectors. For each word only the entry in the code vector corresponding to the assigned neuron index is set to 1.

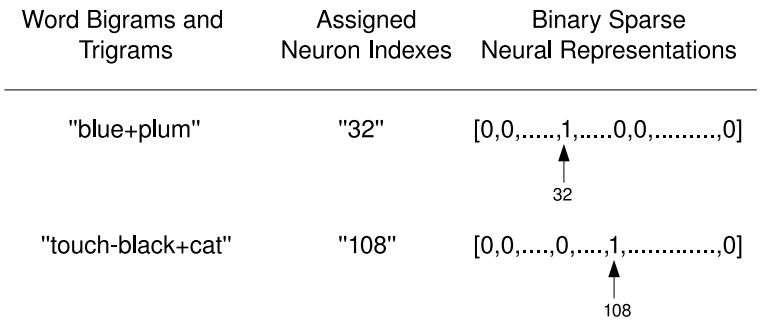

Figure 6.3: Binary sparse neural representations of word bigrams and trigrams. For each word bigram or trigram only the entry in the code vector corresponding to the assigned neuron index is set to 1.

neural representations of sentences are defined as superpositions of the binary sparse neural representations of the word bigrams (or trigrams). Figure 6.4 shows an example of binary sparse neural representations of a sentence.

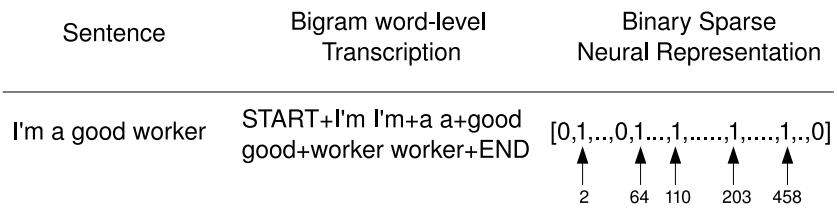

Figure 6.4: Binary sparse neural representation of a sentence as superpositions of the binary sparse neural representations of the word bigrams.

6.2.3 Pattern Storage

Memory BGW

The memory BGW is a heteroassociative memory matrix of dimension $V \times B$, where V is the total number of words in the task vocabulary and B is the number of word bigrams required for the recognition task. It stores the pattern pairs (x^μ, y^μ), $\mu = 1, 2, ..., B$, where x^μ is a superposition of the binary sparse neural representations of the word patterns that construct the word bigram and y^μ is the binary sparse neural representation of the word bigram pattern (1 out of B sparse binary code vectors). Figure 6.5 shows the storage of a pattern pair in the memory matrix BGW.

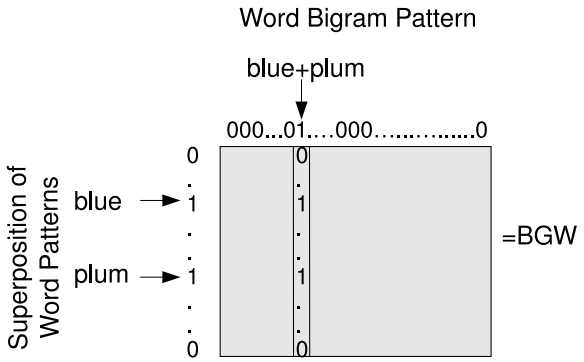

Figure 6.5: Storage in the memory matrix BGW. The address (input) pattern is a superposition of word patterns "blue" and "plum". The output (content) pattern is the word bigram pattern "blue + plum".

Memory M5

The memory M5 is an autoassociative memory matrix of dimension $B \times B$, where B is the number of word bigrams required for the recognition task. It stores word bigram patterns columnwise using 1 out of B sparse binary code vectors. Figure 6.6 shows the storage of a word bigram pattern in the memory matrix M5.

Memory SEN

The memory SEN is a heteroassociative memory matrix of dimension $B \times T$, where B is the number of word bigrams required for the recognition task and T is the number of word trigrams required for the recognition task. It stores the sentence patterns in a distributed way using two representations, the binary sparse neural representations reflecting the word bigram-level similarities of sentences as input patterns and the binary sparse neural represen-

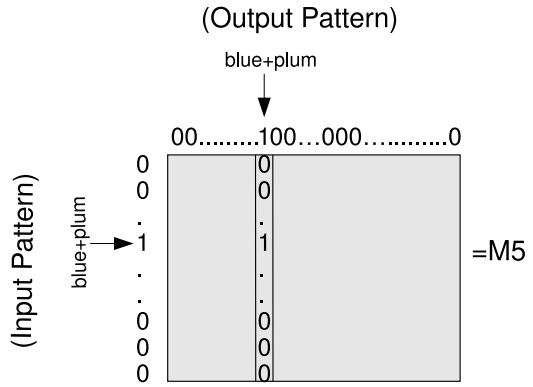

Figure 6.6: Storage in the memory matrix M5. The address (input) word bigram pattern is equal to the content (output) word bigram pattern.

tations reflecting the word trigram-level similarities of sentences as output patterns given in section 5.2. Figure 6.7 shows the storage of a sentence pattern in the memory matrix SEN.

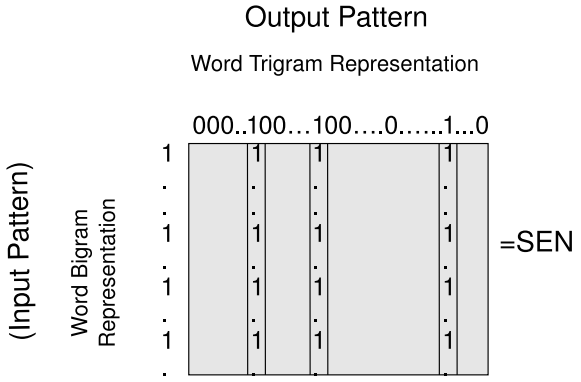

Figure 6.7: Storage in the memory matrix SEN.

6.2.4 Retrieval

During the retrieval of sentences from the word hypotheses recognized in the word recognition network, the word hypotheses are first transformed into word bigrams (memory BGW). Then, the generated word bigrams are stored (memory M5). After the storage of all word bigrams, they are then used to retrieve the word trigrams (memory SEN) from which output words within the spoken sentence are extracted.

Memory BGW

Given two word patterns, the memory BGW generates a word bigram that corresponds to the two input words. A superposition of the binary sparse neural representations of the two word patterns is applied to the memory matrix BGW, and a word bigram pattern is retrieved as a 1 out of B binary sparse code vector (figure 6.8).

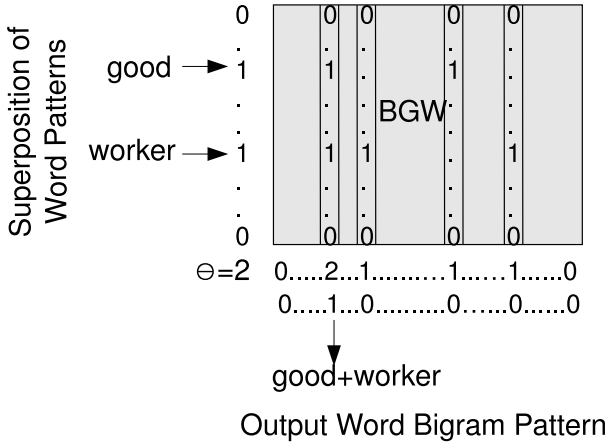

Figure 6.8: Retrieval in the memory matrix BGW.

In figure 6.8, the nonzero-entities in the retrieved pattern belong to the word bigram patterns which have a connection either with the word pattern "good" or with the word pattern "worker". But, at the threshold level $\Theta = 2$, only the word bigram pattern "good+worker" is activated.

Memory M5

The memory M5 holds the word bigram patterns retrieved in BGW. Figure 6.9 (a) shows the activation of the word bigram pattern "a+good". In figure 6.9 (b) another word bigram pattern "good+worker" is applied to the memory M5 which holds the word bigram pattern "a+good". The word bigram pattern held in M5 (e.g. "a+good") is back-propagated to the memory M5 as the new word bigram pattern is applied to M5. In this way, the word bigram pattern already held in M5 is not removed. Therefore, both word bigram patterns are held in M5.

Memory SEN

When a superposition of word bigram patterns is applied to the memory SEN, it retrieves a superposition of word trigram patterns (figure 6.10). For the retrieval, the threshold is set to a global value which is determined by first

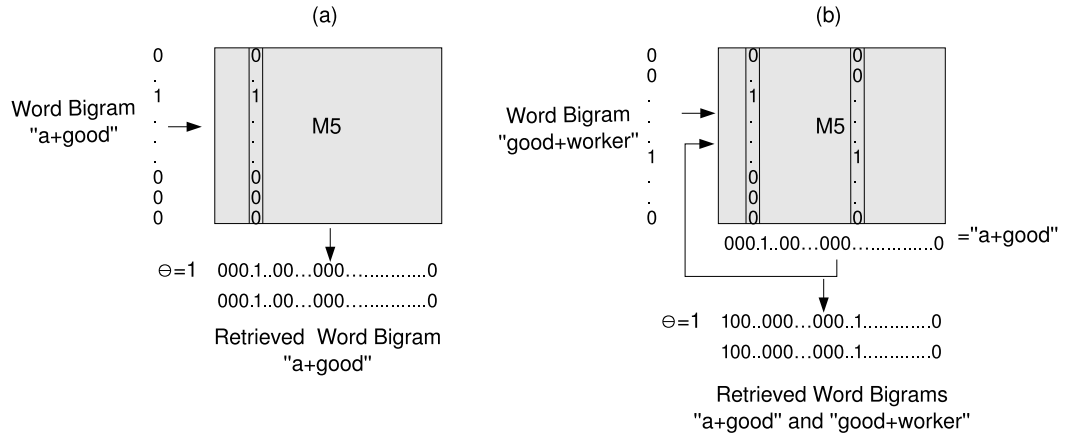

Figure 6.9: Retrieval in the memory matrix M5.

computing the respective activation of each sentence in terms of the word bigrams and then by choosing the maximum activation value.

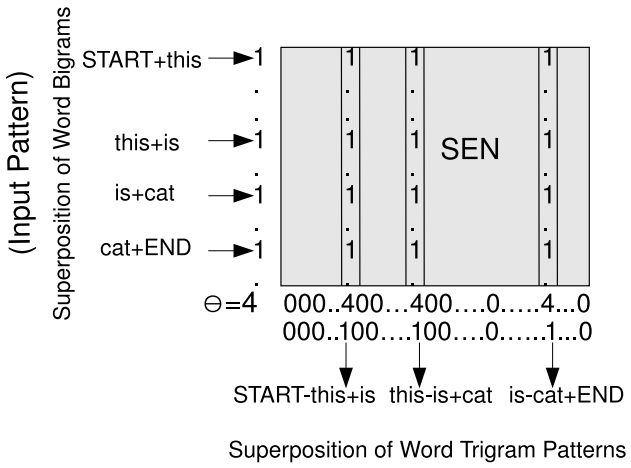

Figure 6.10: Retrieval in the memory matrix SEN.

6.2.5 Functionality

Retrieval in the sentence recognition network is composed of 3 steps.

- A superposition of two input word patterns is provided to the memory BGW via a heteroassociative connection from WRD in the word recognition network. Therefore, the word recognition network waits one global time step to forward a pair of word patterns to the language model network. Then, a word bigram pattern is retrieved in terms of the superposition of two input word patterns (figure 6.11).

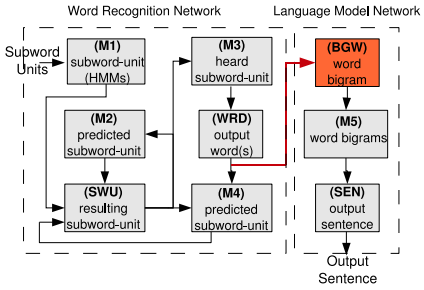

Figure 6.11: Retrieval of a word bigram pattern by applying a superposition of two input word patterns to the memory BGW via a heteroassociative connection from WRD.

- The resulting word bigram pattern is then forwarded to the memory M5 and stored in M5, which holds the word bigram patterns generated up to the current global time step (figure 6.12).

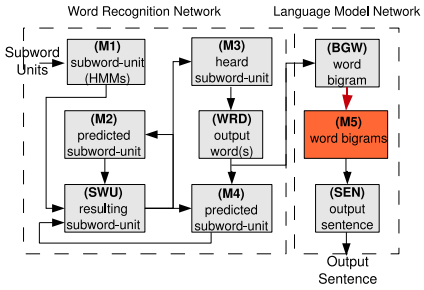

Figure 6.12: Storage of the resulting word bigrams (in BGW) in M5.

- The word bigram patterns in the memory M5 are forwarded to the memory SEN. A global threshold value for the memory SEN is calculated (see section 6.2.4). The memory SEN generates a superposition of word trigrams with respect to the word bigram patterns activated in M5 (figure 6.13).

6.3 An Example of the Extended Hybrid System's Functionality

A speech utterance, e.g. "japan plays by different rules ones rigged for the producer" from the Wall Street Journal (WSJ1) is first processed by HMMs and a subword-unit (e.g. syllable) sequence is then generated, e.g. "START jh_ah p_ae_n p_l_ey_z b_ay d_ih f_er *** r_uw_l_d w_ah_n_z r_ih_g_d f_ao_r dh_ah p_r_ah d_uw s_er END", where the last syllable "***" of the word "different"

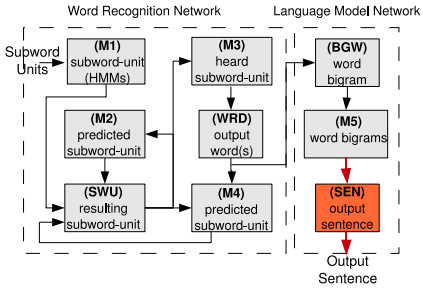

Figure 6.13: Retrieval of word trigrams with respect to the word bigrams held in M5.

can not be recognized (it should have been "ah_n_t") and the single syllable word "rules" is also incorrectly recognized as "r_uw_l_d", which should have been "r_uw_l_z". "START" and "END" denote the beginning and end of the sentence, respectively.

Figure 6.14 shows the word recognition network after the syllables belonging to the word "japan" have been recognized, as explained in section 5.5. The word recognition network generates a unique decision for "JAPAN" in WRD, after processing both syllables belonging to the word.

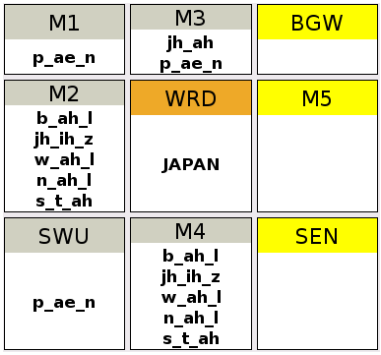

Figure 6.14: The word recognition network after the syllables belonging to the word "JAPAN" have been processed.

In figure 6.15 the sentence recognition network has processed the first word hypothesis "JAPAN" which was generated by the single word recognition network. After recognition, the generated word hypothesis was forwarded to the memory BGW to generate the word bigram. Since the word "JAPAN" is the first word in the sentence, the first word bigram is given as "START+JAPAN" and stored in M5.

In figure 6.16, a few steps later, the syllables "d_ih" and "f_er" belonging to the word "DIFFERENT" have been processed and stored in M3. The single word recognition network produces a superposition of word hypotheses in

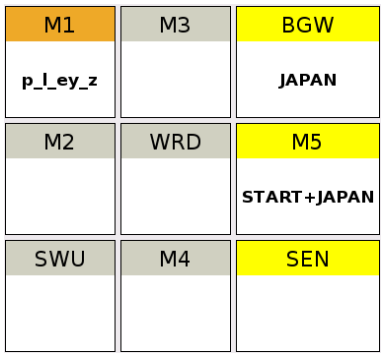

Figure 6.15: The sentence recognition network after the first word "JAPAN" has been recognized.

WRD containing the syllables in M3. The superposition of word hypotheses is then sent to BGW with the previous word hypothesis to generate the corresponding word bigrams.

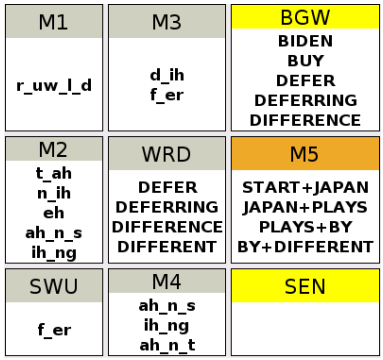

Figure 6.16: The word recognition and sentence recognition network after the incomplete set of syllables for the word "DIFFERENT" have been processed.

Figure 6.17 shows the sentence recognition network after all word hypotheses have been recognized. M5 stores all word bigrams generated by BGW. These word bigrams will be used as input in SEN in order to recognize the output words spoken within the sentence. The output of SEN is a sequence of word trigrams of the spoken sentence and these word trigrams are used to detect the correct sequence of words within the sentence. The word trigrams, e.g. "start-japan+plays japan-plays+by plays-by+different by-different+rules different-rules+ones rules-ones+rigged ones-rigged+for rigged-for+the for-the+producer the-producer+end", are then transformed into "japan plays by different rules ones rigged for the producer".

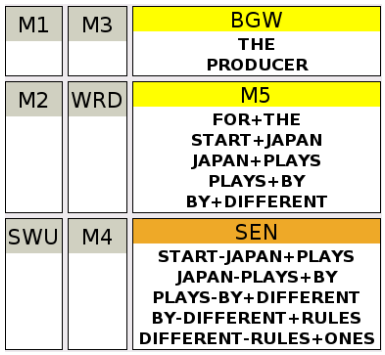

Figure 6.17: The sentence recognition network after all words have been recognized.

6.4 Discussion

In the case of a large vocabulary speech recognition where the search space in the HMM recognition network is very large, the recognition system is not able to recognize every subword-unit spoken by a speaker and can not recognize the correct subword-units but can recognize additional wrong subword-units. If the single word recognition network in the presented hybrid system is not able to recognize the correct words, this can lead to an uncertainty on the word level, i.e., a superposition of possible words. At this point, language models play an important role in recognition because recognition accuracy can be greatly improved by taking advantage of a priori information on the sequences to be recognized. The role of a language model consists of reducing the set of admissible words or sequences of words and therefore the so-called search space, and a priori knowledge is available on the admissible words and word sequences to be recognized. Thus, an NAM-based language model network which works as a language model based on the sentences in the recognition task, is integrated with the single word recognition network to use a priori information on the word sequences to be recognized.

The language model network takes advantage of the correctly recognized words in order to correct the wrongly recognized words. First, it generates word bigrams from the sequence of recognized words and takes only the admissible word bigrams. If two adjacent words in the sequence do not construct a valid word bigram, the NAM does not generate a word bigram.

Willshaw's retrieval strategy is not appropriate for the retrieval process in the language model network (in the memory SEN) because the input pattern can also be composed of wrong word bigrams. Therefore, one step retrieval strategy with a global threshold is applied in the memory SEN.

The memory usage of the memories in the language model network depends on the size of the word bigrams and trigrams within the sentences in the

recognition task. But, for medium size vocabularies, it is also possible to implement a language model network only with word bigrams instead of word trigrams.

The evaluation of the presented hybrid HMM/NAM yields improved recognition results compared to other research as described in the literature, which are HMM-based recognition systems (see chapter 9).

Chapter 7

Learning of Novel Words

In 1993 Young and Ward [104] detected new words in speech by using hidden Markov models (HMMs) for context-dependent phonemes. The idea was, since unknown-words are composed of novel sequences of known, modelled sounds, an unknown-word model could be developed. This model competed with known words from the class N-gram language model and a new word was detected if the "unknown-word" model scored higher than competing known words in a standard recognition search. When a new word was detected, the system asked the user to type the word. It then looked the word up in a large phonetic dictionary to generate a word model from the phonetic spelling. The class N-gram language model was rebuilt in terms of the new word.

In 1996 Sloboda and Waibel [92] proposed a data-driven approach to improve existing dictionaries and to automatically add new words and word variants whenever needed. The approach was based on both the phoneme and the speech recognizer. They also needed transcriptions on a word level for training, resulting in the word boundaries for all word occurrences. Their dictionary learning algorithm briefly contained the following steps:

- collect all occurrences of each word in the database and run the phoneme recognizer on them
- compute statistics of the resulting phonetic transcriptions of all words
- sort the resulting pronunciation candidates using a confidence measure and define a threshold for rejecting statistically irrelevant variants
- add the resulting variants to the dictionary and test with the modified dictionary on the cross-validation set
- retrain the speech recognizer and optionally perform corrective phoneme training

- create a new smoothed language model for the phoneme recognizer, incorporating all new variants.

They showed that using a simple algorithm to extract candidates for phonetic variants will still yield a significant increase in recognition performance.

The hybrid HMM/neural associative memory (NAM) system, presented within the context of this thesis, is capable of incrementally learning novel words during runtime. This chapter first outlines the basic principles of the incremental learning of novel words. Afterwards, the learning of new subword-unit sequences for novel words by HMMs is presented. Another aspect of the learning of novel words in HMMs is the retraining of HMMs for newly generated subword-unit sequences. Then the learning of novel words in the network of NAMs is explained. The chapter concludes with a discussion of the learning of novel words in the presented hybrid system, compared to the learning in HMM-based recognition systems.

7.1 Incremental Learning in the Hybrid System

An important aspect of speech recognition systems that are employed in non-trivial real-world environments is the ability of the system to expand the task vocabulary with novel words during runtime.

In our implementation for the MirrorBot project (see section 9.2), the special command sentence "This is X" triggers the learning process, where X stands for a novel object. The learning of novel words is performed in two steps. In the first step, HMMs preprocess the auditory input "This is X" to generate a plausible subword-unit sequence for the novel word. Therefore, the requirements for the learning is a well-trained set of acoustic models and a large subword-unit-level language model. The learning process in HMMs can be done in two ways. The first way, a few new subword-unit sequences for a novel word are generated without training or adapting HMMs. Or in the second way, HMMs are retrained with newly generated subword-unit sequences for a novel word. The first way requires less time to learn a novel word than the second way because in the second, additional time is needed to retrain HMMs with the new adaptation data for a novel word.

In the second step, the NAM-based word recognition network uses the command "This is" to start the learning process. The learning of a novel word in NAMs is performed by storing the new subword-unit sequence(s) for a novel word in the corresponding memories in the network. The learning of a novel word does not negatively affect the previously learned word patterns.

Before the learning of novel words, it is necessary to identify whether the novel word is already known. This is accomplished by presenting the new

subword-unit representation(s) to the memory WRD in the word recognition network and taking the strength of the memory WRD's output into account. Thereby a strong output indicates that the new subword-unit transcription belongs to an already stored word and a weak output is considered as an unknown-word. The threshold for this is derived from the memory output when testing the new data.

7.2 Incremental Learning without Training HMMs

The auditory input for a novel word is first processed by HMMs to produce a plausible subword-unit sequence for the novel word. As mentioned in the previous section, this requires a well-trained set of acoustic models and a large subword-unit-level language model. It is commonly known that even the same speaker can not pronounce the same word in the same way (consistently). This results in different subword-unit transcriptions for the same word. These different subword-unit transcriptions are used to train NAMs in the word recognition network for the novel word. Figure 7.1 shows the generation of subword-unit transcriptions during the learning of a novel word. The more the subword-unit transcriptions are used to train, the better the recognition performance will be achieved for novel words.

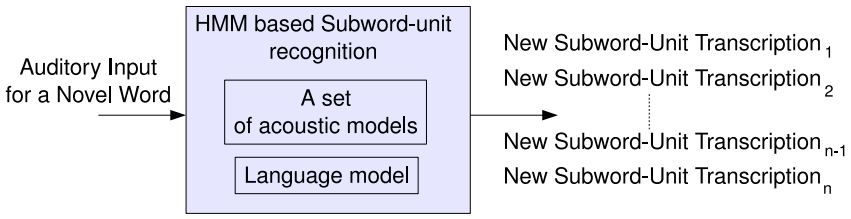

Figure 7.1: Generation of different subword-unit transcriptions for a novel word.

7.3 Incremental Learning by Retraining HMMs

An alternative approach for adding novel words to the task vocabulary is not to simply generate a few new transcriptions for a novel word, but to retrain HMM acoustic models with newly generated transcriptions of a novel word and to expand the language model with these newly generated transcriptions. At first, newly generated subword-unit transcriptions are decomposed into acoustic units using the pronunciation dictionary. Then, these HMM acoustic models are retrained with newly generated transcriptions. The language model is expanded with the probabilities of the new subword-unit transitions within the newly generated transcription (figure 7.2). This alter-

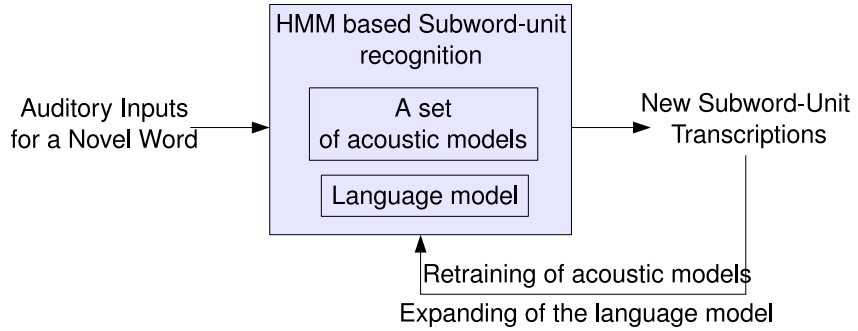

Figure 7.2: Incremental learning with new subword-unit sequences for novel words. Newly generated subword-unit transcriptions are used to retrain the corresponding HMM acoustic models and to expand the language model with new subword-unit transitions.

native approach is more time consuming than the previous approach, but achieves better recognition performance.

7.4 Learning in Neural Associative Memories

After the generation of new subword-unit transcriptions for a novel word, these new transcriptions are forwarded to the NAM-based word recognition network. The word recognition network activates a learn signal at several associative memories involved in the learning of novel words whenever ”This is“ has been recognized. Memories with an active learn signal do not activate superpositions of several patterns if the address pattern does not match a stored pattern well, but rather generate novel representations and connections between memories by performing Hebbian learning if either the source or the target memories have learned a novel pattern.

In the single word recognition network, the learning takes place in the heteroassociative memories M2, M4 and WRD. In the memory M2, new subword-unit transitions within the new word transcriptions are stored. Given a new subword-unit (diphone) transcription ”p+ih ih+z ae-r“ for the novel word “pear”, figure 7.3 illustrates an example of the learning of new diphone transitions “p+ih→ih+z” and “ih+z→ae-r” in the memory M2.

In order to store the novel word in the memory WRD, two sparse binary neural representations are necessary, a sub-symbolic (subword-unit) neural representation and a randomly generated neural representation. The sub-symbolic neural representation is obtained from the subword-unit transcription of the novel word (see section 5.2) and used as the input pattern. The second representation is obtained by randomly choosing a number of neurons from the output population as the output pattern. An example of the

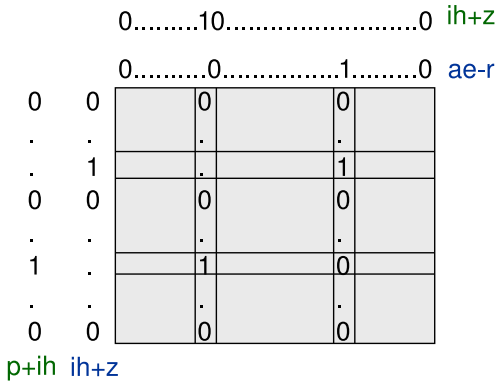

Figure 7.3: The learning process of new subword-unit transitions "p+ih→ih+z" and "ih+z→ae-r" for the novel word "pear" in the memory M2.

learning of the novel word "pear" in the memory WRD is depicted in figure 7.4.

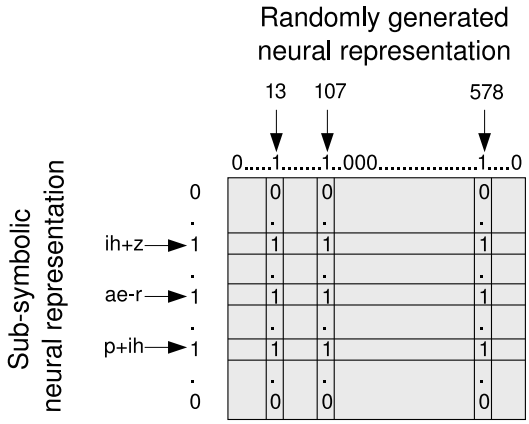

Figure 7.4: An example of the learning of the novel word "pear" in the memory WRD using its sub-symbolic representation "p+ih ih+z ae-r" and its randomly generated neural representation as a 3 out of *r* binary code vector.

As said in section 5.4.1, the memory M4 is the transposed form of the memory WRD. Therefore, the same neural representations used in WRD are also used to store the novel word in the memory M4, but its randomly generated neural representation is used as the input pattern and its sub-symbolic neural representation is used as the output pattern. Figure 7.5 displays the same example for the learning of the novel word "pear" in the memory M4.
Finally, the corresponding auto- and heteroassociative connections in the network are also updated. After learning, the novel word can be used and processed as well as the previously stored words were. Thus, the system can

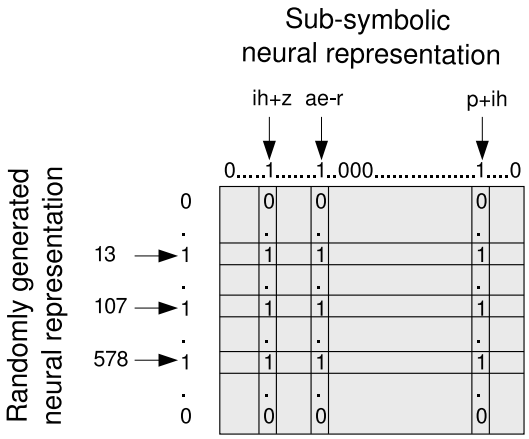

Figure 7.5: An example of the learning of the novel word "pear" in the memory M4 using its randomly generated neural representation, a 3 out of r binary code vector, as the input pattern and its sub-symbolic representation "p+ih ih+z ae-r" as the output pattern.

correctly recognize the word "pear" in a sentence like "bot show pear" after "pear" has been learned.

The language model network is updated with respect to the sentences containing the novel word to store possible novel word bigrams and trigrams in the novel sentences. But, in the case of the learning of novel words, the structures of the memories in the language model network should be changed, i.e., neural representations randomly generated from a specific neuron population can be used to store words, word bigrams and trigrams instead of the binary 1 out of V, 1 out of B, and 1 out of T representations, respectively. These kinds of neural representations enable us to keep the size of the associative memories in the language recognition network stable during the learning of novel word bigrams and trigrams.

7.5 Discussion

In many speech recognition applications such as HMMs, it is usually difficult to increase the vocabulary size during runtime. To achieve this, many parameter files required for the application have to be changed and the modifications to the lexicon, the language model and training of new subword-unit models are necessary. However, the presented HMM/NAM approach enables us to easily enlarge the vocabulary with novel words during runtime because it needs only a sequence of subword-units from HMMs for the novel word [49].

In the context of this thesis two different strategies to generate new tran-

scriptions for novel words in HMMs are introduced: the incremental learning without further training of HMMs and the learning with retraining of HMMs. The first learning strategy needs less time to learn novel words than the latter, but achieves less recognition performance. On the other hand, the incremental learning method with retraining of HMMs requires more time to learn novel words, but gives more satisfactory recognition results. In order to get a reasonable sequence of subword-units from HMMs, both strategies require a large, well-trained set of HMMs.

The learning process in NAMs works very fast using the HMM output subword-unit sequence. As a consequence of the learning of novel words, only novel patterns or pattern pairs need to be amended while the previously stored patterns or pattern pairs remain unchanged. During learning, depending on the recognition task, the language model network needs also a priori information on the sequence of words to be recognized, which includes the novel word.

Comparing the presented hybrid system to standard HMMs, the adding process of novel words in the hybrid system is faster and easier than HMMs and makes it possible to learn during runtime, which meets the requirements of real-world applications.

Chapter 8

Speech Corpora

Within the scope of this thesis, several speech corpora are used to test the recognition performance of the presented hybrid system. In addition to well-known speech corpora for continuous speech recognition, such as TIMIT and Wall Street Journal (WSJ1), we also used speech corpora for special speech recognition tasks, such as language understanding on a mobile robot [26, 47] and distributed speech recognition. All the speech corpora used for evaluation are given in the following sections.

8.1 MirrorBot Speech Data

The MirrorBot speech data is developed for a language understanding task embedded into a robot. The speech data is based on a small vocabulary of 43 words and consists of 105 sentences (see appendix A) from 4 speakers who are not native English speakers. The speech data consists of simple English command sentences without prepositions. The language is relatively easy for humans to understand and speak. 5-fold cross-validation is used to test the performance of the presented hybrid system. Each fold contains 21 speech utterances from each speaker. One of the 5 subsets is used each time, as the test set and the remaining 4 subsets are used as the training set. Each speaker is used both in the training and test sets. A set of 45 phonemes is used, which are based on the TIMIT phonetic symbols. For each utterance the MirrorBot speech data corpus includes a 16-bit, 44,100Hz PCM encoded speech waveform file.

8.2 German Bus-Stop Names Speech Corpus

The German bus-stop names speech corpus is a set of 279 German bus stop names, which originated from the Institute of Information Technology, University of Ulm. The training set consists of 14 speakers, whereas the test set

consists of 5 speakers. Note that the speakers are native German speakers. The speakers, both in the training and the test set utter these 279 bus stop names. The number of word tokens in the test set is 1,395 while it is 3,906 in the training set. A set of 39 phonemes is used. For each utterance, the corpus includes a 16-bit, 16,000Hz PCM encoded speech waveform file.

8.3 TIMIT Continuous Speech Corpus

The TIMIT corpus [79] of read speech is designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems. Text corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), Standford Research Institute (SRI), and Texas Instruments (TI). The TIMIT corpus contains a total of 6,300 sentences, i.e., 10 sentences are spoken by each of 630 speakers from eight major dialect regions of the United States. The text material in the TIMIT corpus consists of 2 dialect sentences (the SA sentences), 450 phonetically-compact sentences (the SX sentences), and 1,890 phonetically-diverse sentences (the SI sentences). The dialect sentences are not included in the experiment. Each speaker uttered 5 of the SX sentences and 3 of the SI sentences. Each SX sentence was uttered by 7 different speakers while each SI sentence was uttered only by a single speaker. Table 8.1 summarizes the TIMIT speech material.

Table 8.1: *TIMIT speech material.*

Sentence Type	#Sentences	#Speakers	Total	#Sentences/Speaker
Dialect (SA)	2	630	1,260	2
Compact (SX)	450	7	3,150	5
Diverse (SI)	1,890	1	1,890	3
Total	2,342		6,300	10

The speech data is composed of three sets: a set for training the acoustic models, a development set for optimizing language model scaling factor and word insertion penalty, and a test set for evaluating the acoustic models. Table 8.2 shows details of the speech data.

Table 8.2: *TIMIT data sets.*

	Train	Test	Devel.	Total
Word tokens	30,132	9,455	1,570	41,157
Speakers	462	144	24	630

The TIMIT corpus transcriptions have been manually labelled and include time-aligned, manually verified phone and word segmentations. The original set of 61 phonemes was reduced to a set of 45 phonemes. For each utterance the TIMIT corpus includes a 16-bit, 16kHz PCM encoded speech waveform file.

8.4 Wall Street Journal (WSJ1)

The Wall Street Journal corpus (WSJ) is a collection of speech about business news topics, as typically found in the printed Wall Street Journal. This corpus consists of two parts, WSJ0 (15.1 hours, 7,184 training utterances) and WSJ1 (approximately 73 hours, 78,000 training utterances). The corpus covers 284 different speakers. The training portion contains 24,180 distinct words. In total, there are 576,859 words. The pronunciations and the phoneme set are from the Carnegie Mellon University (CMU) Pronunciation Dictionary. The phoneme set contains 39 phonemes.

The presented hybrid approach is tested with three test sets from WSJ1 using the non-verbalized 5k (4,986) word closed vocabulary and the non-verbalized 20k (19,979) word open vocabulary. A set of 40 HMMs (39 monophones plus 1 silence model) is used. These test sets are as follows:

- *si_dt_05.odd*: The WSJ1 5k development test has 2,076 distinct words and a total of 13,866 words. In particular, the si_dt_05.odd set is used for the 5k word closed vocabulary task and it is a subset of the WSJ1 5k development test data. It is formed by deleting sentences with out-of-vocabulary (OOV) words and choosing every other remaining sentence and is comprised of 248 sentences from 10 speakers.

- *si_dt_20*: The WSJ1 20k development test has 503 sentences and 2,464 unique words with the total count of 8,227 words. It also contains 187 out-of-vocabulary words. 2.27% of the word occurrences in the development set are not included in the standard 20k-word vocabulary. The WSJ1 20k development test data consists of 503 sentences from 10 speakers.

- *si_dt_s6*: The WSJ1 spoke 6 development test data (the 5k-word read WSJ data) consists of 395 sentences from 10 speakers. The Sennheiser test data contains 4,101 distinct words with the total count of 20,324 words. It also has 1,570 out-of-vocabulary words. 5.35% of the word occurrences in the test set are not included in the 5k-word vocabulary.

8.5 Discussion

Within the scope of this work, different speech corpora are used to evaluate the presented approach. These corpora range from small vocabularies like *MirrorBot and German Bus-Stop Names Speech Data* to large vocabularies like *TIMIT and WSJ1*. Whereas the *MirrorBot Speech Data* is a speaker-dependent corpus, the others are speaker-independent corpora (see Table 8.3).

Table 8.3: *Speech corpora.*

Speech Corpus	Type	Number of Speakers in	
		the Training Set	the Test Set
MirrorBot	Speaker-Dep.	4	4
German Bus-Stop Names	Speaker-Indep.	14	5
TIMIT	Speaker-Indep.	462	144
WSJ1	Speaker-Indep.	284 (WSJ0+WSJ1)	10

The MirrorBot Speech Data is composed of simple English command sentences and is considered for isolated speech recognition. The German Bus-Stop Names Speech Corpus consists of German bus-stop names. The TIMIT and WSJ1 speech corpora are well-known speech data for continuous speech recognition. Among the corpora used within the scope of this thesis, only TIMIT is labelled on the phoneme level. The other corpora contain only word-level transcriptions.

Chapter 9

Application and Evaluation

In this chapter the application of the hybrid approach presented within the scope of this thesis will be shown in several different experiments with small and large data sets. The recognition performance of the hybrid system will be compared with recognition results of other research as described in the literature for TIMIT and the Wall Street Journal (WSJ1) continuous speech corpora. For other speech corpora (MirrorBot and German Bus-Stop Names project) the developed hybrid approach will be compared to pure hidden Markov model (HMM)-based triphone recognizers. In order to compare different approaches to automatic speech recognition (ASR), we will first give some performance-measures in ASR, recognition accuracy, word error rate (WER) and recognition speed. It will be seen that the WERs obtained with the presented hybrid recognition system are competitive to the results reported in other research. Furthermore, the recognition speed of the presented hybrid recognition system is also compared to pure HMMs.

For the MirrorBot and the German Bus-Stop Names projects, the HMM-based subword-unit recognition in the presented hybrid approach was developed using HTK speech recognition toolkit [103]. For the TIMIT and WSJ1 speech corpora, the HMM-based subword-unit recognition in the presented hybrid approach was developed using Sphinx-4 speech recognition system [1]. The syllable based transcriptions of words are obtained using a syllabification software [2].

9.1 Measuring Performance

In order to compare different approaches to ASR, it is necessary to estimate their performance. The experiments used to validate different approaches are mostly based on recognition accuracy, word error rate, and recognition speed measured on actual speech data as evaluation criteria. The recognition accuracy and the word error rate measure whether the suggested approach

improves the recognition of speech data or not. The recognition speed gives information on whether the recognizer is slow or not.

Moreover, it is insufficient to evaluate the approach only on the training data. Thus, the usage of a test data set different from the training set is necessary to get a more realistic assessment of an ASR approach. A method for this is the cross-validation approach.

9.1.1 Recognition Accuracy

The *Word Correct Rate* is given as:

$$WordCorrectRate = 100 \times \frac{Number\ of\ Correctly\ Recognized\ Words}{Number\ of\ Spoken\ Words}. \tag{9.1}$$

Words that are inserted by mistake are not included in computing the *Word Correct Rate*. However, the usability of the recognition output is low because of insertion errors which are made in one sentence. Therefore, the performance of a speech recognition system is usually measured in terms of misrecognized words. The hypothesized text, produced by the speech recognizer, is compared to the reference text by using a dynamic programming string alignment, which globally minimizes the Levenshtein distance [61]. The result of the comparison between the reference and the hypothesized text will be the minimum number of

- *word substitutions*
- *word insertions*
- *word deletions*,

which is defined as *Word Error Rate* (WER):

$$WER = 100 \times \frac{Insertions + Substitutions + Deletions}{Number\ of\ Spoken\ Words}. \tag{9.2}$$

The *recognition accuracy* is defined as $100 - WER$.

9.1.2 Recognition Speed

To evaluate speech recognition systems, the word error rate or recognition accuracy is used. However, it is often interesting to know how fast a recognition system performs. The recognition speed is defined as the required recognition time, in seconds, of speech input. Quite obviously, the recognition time depends on the computer and compiler used for the experiments.

9.1.3 Cross-Validation

Cross-validation is one of the several techniques used to estimate how well a learning approach is going to perform on unseen data when only a limited amount of data is available for evaluation. The idea is not to use the complete data set for training, but to only use a part of the data set for training and the remaining part of the data set for testing.

The holdout method is the simplest kind of cross validation. The data set is seperated into two sets, called the training set and the test set. The advantage of this method is that it takes no longer to compute. However, its evaluation can have a high variance. The evaluation may be significantly different depending on how the division is made.

k-fold cross-validation is one way to improve the holdout method. The data set is divided into k subsets, which are called folds. The number of folds is naturally limited to $2 \leq k \leq M$ where M is the total number of samples in the data set. Each time, one of the k subsets is used as the test set and the remaining $k-1$ subsets are used as the training set. Then the average error across all k trials is computed. The advantage of this method is that how the data actually gets divided matters less. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

Leave-one-out cross-validation is k-fold cross-validation taken to its logical extreme, with k equal to M, the number of data points in the set. This means that at M separate times, the training set contains all the data except for one point and a prediction is made for that point. As before, the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross-validation error is good, but at first pass it seems very expensive to compute.

9.2 MirrorBot Project

The presented hybrid HMM/neural associative memory (NAM) system is integrated with a biologically inspired language understanding system [64], which is embedded into a robot to demonstrate its correct understanding of spoken command sentences by performing the corresponding actions. A white table with objects lying on it stands in front of the robot and then the robot receives short command sentences like “bot show plum” (see section A) via a microphone. The robot has to respond to spoken commands concerning these objects. In order to fulfill the requested tasks, the robot needs to perform word recognition, sentence recognition as well as action planning, simple vision tasks (object recognition) and motor commands (finding objects, lifting them, dropping them and moving the robot to certain places).

An overview of the architecture of the whole robot software is given in figure 9.1. The system consists of several functional modules that are separated in the figure between low-level (Sensory preprocessing, left column) and high-level (Cortical processing, right column). Figure 9.1 gives a broad overview of the functioning of the model.

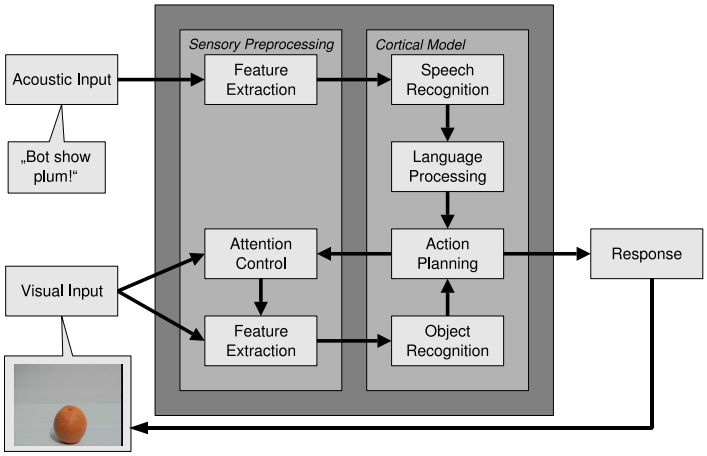

Figure 9.1: Overview of the software modules in the robot.

Information between the functional models flows unidirectionally. The main flow of information in speech recognition modules is as follows: a spoken command sentence enters the system via the microphone into the HMM-based subword-unit recognition module and is transformed into some elementary speech components (e.g., into phonemes or syllables). The output is then forwarded to the word recognition module, which generates matching words and forwards them to the sentence recognition module. The sentence recognition module is responsible for capturing the semantics, i.e., the meaning of the command. It does so by comparing the input words with grammatical rules that are built into the system and tries to determine the sentence type of the command. Once the meaning of the sentence becomes clear, it is then forwarded to the action planning module which is also a network of associative memories and can decompose requested actions (e.g., to put something somewhere) into a sequence of predefined elementary motor commands (e.g., lifting the object, moving it to the right place and dropping it). Communication between NAM based modules takes place using the neural representations.

Feature extraction was carried out at a frame rate of 10 ms, with a pre-emphasis of 0.97. 12 Mel Frequency Cepstral Coefficients (MFCC) and log-energy with first and second order time derivatives were calculated, for a total of 39 features.

The HMMs are constructed of the three-state continuous 8-Gaussian word-

internal triphone models (see section 3.2.2) with a left-to-right topology with self-loops. The training procedure is given in section 5.3. HMMs are trained with the training data produced using 5-fold cross-validation (see section 8.1) and the TIMIT training set (see section 8.3). Therefore, well-trained HMMs can give better recognition performance on the subword-unit level for the learning of novel words. Word-internal triphone-level bigram models are built using all word-internal triphones in the training and test sets of the MirrorBot and the TIMIT speech data.

In the neural associative memories, a total of 7,412 word-internal triphones are used as subword-units and words are represented using their subword-unit level transcriptions and randomly generated 5 out of 200 code vectors.

The presented hybrid HMM/NAM system has been tested on the 5-fold cross validation test set and compared to an HMM-based triphone recognizer (see table 9.1).

Table 9.1: *The average WERs over 5-fold cross validation on the MirrorBot Speech Data.*

Recognizer Type	WER (%)
HMM-based triphone recognizer	2.144
The presented hybrid approach	0.21

The results show that the presented hybrid approach achieved an average WER less than the HMM-based triphone recognizer. This is due to the fact that the wrong word hypotheses generated by the word recognition network can be corrected by the language model network using the correctly recognized words (the priori information on the word sequences to be recognized). The output of the HMM-based triphone recognizer contains more *word insertions* than that of the presented hybrid approach.

The speed of the hybrid system is also compared to the HMM-based triphone recognizer on a standard laptop machine (Intel Core 2 Duo 2.00 GHz). For a sentence like "bot show ball", the speed of the HMM-based triphone recognizer is measured at 2 seconds, whereas that of the presented hybrid system is measured at 4 seconds (3 seconds for HMM-based subword unit recognition and 1 second for NAMs). This is due to the fact that the size of the vocabulary of subword-units (word-internal triphones) is larger than the size of the vocabulary of words. The vocabulary of subword-units contains a very large set of word-internal triphones from MirrorBot and TIMIT Speech Corpora, which is required to learn novel words.

9.3 German Bus-Stop Names Project

The presented hybrid HMM/NAM approach can be applied to various speech controlled systems. An example of these systems is the *German Bus-Stop Names Project* [51]. The user utters a single command word or word phrase, a bus stop name, to a mobile phone and the speech recognizer on the remote server, correctly recognizes the command, implements the corresponding task to send information about the spoken bus stop name from the remote server to the user. Due to the single word commands, the language model network is not used here.

In the system, word-internal triphones are determined as subword units. Therefore, a word-internal triphone recognizer is designed by using HMMs, in which an acoustic model for each word-internal triphone is generated. The HMMs are constructed of the three-state continuous 8-Gaussian word-internal triphone models with a left-to-right topology with self-loops. The training procedure is given in section 5.3. The acoustic models are then trained on the German Bus-Stop Names training set (see section 8.2) and word-internal triphone based simple bigram models are used. The word recognition network is then designed based on a total of $1,284$ word-internal triphones. The words are stored using their word-internal triphone-level transcriptions and randomly generated 5 out of $1,000$ code vectors.

The presented hybrid HMM/NAM system has been tested on the test set (see section 8.2) and compared to an HMM-based triphone recognizer (see table 9.2).

Table 9.2: *WERs on the test set of German Bus-Stop Names Speech Corpus.*

Recognizer Type	WER (%)
HMM-based triphone recognizer	1
The presented hybrid approach	2

There is a slight difference between the presented hybrid system and the HMM-based triphone recognizer. This difference can decrease using larger word parts such as syllables instead of triphones and a more efficient language model.

The speed of the hybrid system is compared to the HMM-based triphone recognizer on a standard laptop machine (Intel Core 2 Duo 2.00 GHz). For a bus stop name like "Abzweigung_roter_berg", the speed of the HMM-based triphone recognizer is measured at 2 seconds, whereas that of the presented hybrid system is measured at 4 seconds (3 seconds for HMM-based subword unit recognition and 1 second for NAMs). The reason for the low speed of the hybrid system is that the search space in the recognition network of HMMs

in the hybrid system is larger and more complex than that of HMM-based triphone recognizer.

9.4 Large Vocabulary Continuous Speech Recognition

The presented hybrid HMM/NAM approach can also be applied to large vocabulary continuous speech recognition systems. Therefore, the recognition performance of the hybrid system is evaluated with the TIMIT and the WSJ1 speech corpora. The results are compared to other research results in the literature.

9.4.1 TIMIT Continuous Speech Corpus

The acoustic waveforms from the TIMIT are first parameterized into a 39-dimensional feature vector consisting of 12 cepstra plus the log energy, deltas and delta deltas, normalized using cepstral mean subtraction. The HMMs are constructed of the three-state continuous 8-Gaussian cross-word triphone models. Each HMM has a left-to-right topology with self-loops. The training procedure is given in section 5.3. HMMs are trained with the TIMIT training data (see section 8.3). The subword unit recognition part is designed to generate "syllables" as output speech components. In total, there are 3,583 syllables in both the TIMIT training and test sets. Syllable-level trigram models are built using all syllables in the TIMIT training and test sets. In the NAM based word recognition network, the words are stored using their syllable-level transcriptions and randomly generated 5 out of 2,000 code vectors. A total of 17,983 word bigrams and 20,075 word trigrams within the sentences from the TIMIT training and test sets are used in the language model network.

In table 9.3 the recognition performance of the presented hybrid HMM/NAM system is compared to three examples of research in the literature on the TIMIT test data [90, 33, 34]. Sethy and Narayanan [90] built three separate recognizers corresponding to the different acoustic units of interest, i.e. phoneme, syllable and word. The design of the phoneme based recognizer follows the standard flat start Baum Welch reestimation strategy with decision tree based triphone creation and clustering (see section 3.2.5). Hämäläinen et al. [33, 34] suggested that longer length acoustic units are better suited for modelling pronunciation variation and long-term temporal dependencies in speech than traditional phoneme-length units and they yielded substantial improvements in recognition accuracy. They used a hierarchical method that employs a mixture of word-, syllable- and phoneme-length units and a stan-

dard procedure with decision tree-based state tying to train the triphone recognizers. A language model for the TIMIT was built using all orthographic words in the training and test sets [90, 33, 34]. The system based on the proposed approach achieves a lower WER than the triphone results, for models with 8 and 16 Gaussian mixtures, reported in other research in the literature [90, 33, 34]. This improvement is due to the "fault-tolerance" property of NAMs. The NAM-based language model network that uses a priori information on the words to be recognized has a significant effect on the recognition performance.

Table 9.3: *WERs on TIMIT.*

Recognizer Type	WER (%)
Triphone [90] with 8 Gaussian mixtures	26
Triphone [33, 34] with 16 Gaussian mixtures (best performing triphones)	8.1 ± 0.6
The Presented Hybrid Approach with 8 Gaussian mixtures	7.03

9.4.2 Wall Street Journal Corpus

The acoustic waveforms from the WSJ are parameterized using 12 MFCC coefficients and normalised energy plus first and second order derivatives. The HMM-based subword-unit recognition is designed using cross-word triphone models to generate syllables as output speech components. Each HMM has a left-to-right topology with self-loops and is constructed of the three-state continuous 8-Gaussian cross-word triphone model. The training procedure is given in section 5.3. HMMs are trained with the WSJ training data (see section 8.4). Syllable-level trigram models are built using all syllables in the test sets.

For the non-verbalized 5k (4,986) word closed vocabulary, 2,682 syllables are used for the subword-unit recognition. In the NAM based word recognition network, the words are stored using their syllable-level transcriptions and randomly generated 2 out of 2,000 code vectors.

For the WSJ1 5k development test set (si_dt_05) a total of 6,241 word bigrams and 7,514 word trigrams, based on 5k-vocabulary words, are used in the language model network.

In table 9.4 the recognition performance of the presented hybrid HMM/NAM system is compared to the research in the literature on a subset (248 sentences) of si_dt_05 test data [101], where several recognizer types, such as

cross-word and word-internal triphones, and training data sets were compared in terms of recognition performance. The WER result reported by Woodland et al. [101] is obtained by building a cross-word triphones with 10 mixture components using the standard training procedure (see section 3.2.5) and the WSJ speaker-independent-284 training set. They run experiments using the standard bigram and trigram grammars and took the pronunciations and the phoneme set from the Dragon Wall Street Journal Pronunciation. The system based on the presented hybrid approach achieved a lower WER than the WER, based on the trigram language model, reported by Woodland et al. [101].

Table 9.4: WER on the WSJ1 5k (si_dt_05.odd).

Recognizer Type	WER (%)
Cross-word Triphone [101]	6.09
The Presented Hybrid Approach	4.91

For the WSJ1 spoke 6 development test data, a total of 13,029 word bigrams and 16,764 word trigrams, based on 5k-vocabulary words, are used in the language model network.

The recognition performance of the presented hybrid HMM/NAM system is tested on the Sennheiser data (395 sentences) of si_dt_s6 test data. Table 9.5 shows the recognition performance of the presented hybrid HMM/NAM system tested on the test data.

Table 9.5: WER on the WSJ1 5k Sennheiser data of si_dt_s6.

Recognizer Type	WER (%)
The Presented Hybrid Approach	2.12

For the non-verbalized 20k (19,979) word open vocabulary, 5,965 syllables are used for the subword-unit recognition. In the NAM based word recognition network, the words are stored using their syllable-level transcriptions and randomly generated 2 out of 5,000 code vectors. A total of 6,543 word bigrams and 7,324 word trigrams, based on 20k-vocabulary words, are used in the language model network.
In table 9.6 the recognition performance of the presented hybrid HMM/NAM system is compared to the research in the literature on the si_dt_20 test data [88], where the standard recognition vocabulary was defined as the most likely 20,000 words in the corpus and the standard language model was defined as a trigram language model estimated specifically for these 20k words. In this research [88], Schwartz et al. modified the language model training

Table 9.6: WER on the WSJ1 20k (si_dt_20).

Recognizer Type	WER (%)
Language Training [88] (standard 20k word vocabulary)	16.4
The Present Hybrid Approach	13.21

text by applying rules to simulate the differences between the training text and what people actually said. Their recognition results are based on the enchanced language model. The system based on the presented hybrid approach accomplished a lower WER than the result reported by Schwartz et al. [88].

9.5 Discussion

In order to evaluate the performance of the developed approach, it is applied to different speech recognition systems and good improvements on the WERs are observed. These systems range from simple command control systems with small vocabularies to the large vocabulary continuous speech recognition systems.

In order to compare the presented approach with other speech recognition research in the literature, it is necessary to estimate their performance using the same speech corpora, i.e. the training and the test sets are to be the same. Otherwise, it is impossible to make a confident comparison among different approaches. Although there are many hybrid HMM/ANN speech recognition systems in the literature, the presented hybrid system can not be compared to these systems because they were evaluated with different speech data. Therefore, the presented hybrid system is compared to the research [33, 34, 101, 88] in the literature because the same corpora was used, including TIMIT and WSJ1, to evaluate their studies and they also reported the lowest WERs on the TIMIT and WSJ1 (5k and 20k word trigrams) test sets. These speech corpora are also well-known corpora in the ASR field.

For speech corpora with **small vocabularies** (MirrorBot and German Bus-Stop Names speech corpora), the presented system is compared to our own pure standard HMM-based speech recognition systems, developed by the Institutes of Neural Information Processing and Information Technology at the University of Ulm [51], in terms of the recognition speed and performance. The results show that the HMM-based subword-unit recognition part of the proposed system consumes more recognition time than pure HMM-based speech recognition systems. This is due to the fact that the search space in the HMM recognition network, used to recognize subword-units, is more com-

plex than the search space in the HMM recognition network in recognizing words. But, the NAM-based word recognition part of the proposed hybrid system works quickly.

The feasibility of the proposed hybrid system was demonstrated by the WER evaluation, which measures the performance of a speech recognition system in terms of misrecognized words. In the MirrorBot project, the HMM-based triphone recognizer has a higher WER of 2.144% than the presented hybrid approach (0.21% WER) because the output of the HMM-based triphone recognizer contains many word insertions. For the German Bus-Stop Names project, where each speech utterance consists of a single word or a phrase, the WER achieved by the HMM-based triphone recognizer is 1% higher than the WER (98%) of the presented approach because there is not a priori information on the word sequences to be recognized. Therefore, the language model network does not help increase the performance of the hybrid system.

From the recognition results of the **large vocabulary** sentence based speech recognition tasks (TIMIT and WSJ1 Speech Corpora), it is seen that the developed approach achieved better performance than the HMM-based recognition systems. For the sentence based speech recognition tasks, the language model network of the proposed approach especially increases the recognition performance using a priori information on the word sequences to be recognized. For the large vocabulary continuous speech recognition systems (TIMIT and WSJ1), the performance of the presented approach is very promising. When its recognition results are compared to other reported results in the literature (see tables 9.3, 9.4 and 9.6), the presented hybrid system yields less WERs than other research in the literature [33, 34, 101, 88]. The improvements achieved by the presented hybrid approach depends on the NAM-based language recognition network using the language information.

In the proposed hybrid system the performance of NAMs is highly dependent on the performance of HMMs. Therefore, it is important for HMMs to choose a correct type for output subword-units.

Chapter 10

Contributions

The presented new hybrid hidden Markov model/neural associative memory (HMM/NAM) approach makes the following contributions:

- **Application of neural associative memories to automatic speech recognition [47, 48]**

  This thesis introduces a new approach to automatic speech recognition to augment the performance of speech recognizers. This approach is based on HMMs on the elementary subword-unit level and networks of NAMs on a higher level, such as word and language levels. First, HMMs generate a sequence of subword-units and provide it to a network of NAMs on a higher level. At the second stage of recognition, possible word hypotheses are then recognized from the HMM output stream and the output words are retrieved according to the priori information on word sequences to be recognized.

- **Evaluation of different subword-units for HMM output speech components**

  During the evaluation of the developed hybrid approach, different subword-units are employed. The type of subword-units used as HMM output speech components plays an important role for the structure of the NAMs in the single word recognition network in the developed hybrid approach. Depending on the size of the vocabulary task, the developed hybrid can utilise different types of subword-units, such as context-dependent phonemes, demi-syllables or syllables. Subword-units in smaller size (e.g. context dependent phonemes) could be suitable for small vocabulary, whereas the usage of subword-units in longer size, such as demi-syllables or syllables, is appropriate for large vocabulary.

- **Representing and handling ambiguities [64]**

  The ambiguity mechanism allows for dealing with ambiguities on the subword-unit and word levels. Ambiguities on the subword-unit level can arise in the HMM output subword-unit sequence when a speaker mispronounces words in the speech utterance, words are run into each other, or there are homophones in the task vocabulary; in other words the subword-units that are wrongly recognized or can not be recognized by HMMs. When NAMs can not generate a unique word on the single word level due to the ambiguities on the subword-unit level, a superposition of all possible words is activated. The ambiguity on the word level might be resolved in further processes using additional context information (semantics) or a priori information on the word sequences to be recognized. This means that additional input must precisely support one pattern of the superposition in order to resolve the ambiguity.

- **Extendability of the task vocabulary by the learning of novel words [49, 50]**

  The hybrid HMM/NAM approach, presented within the context of this thesis, is capable of incrementally learning novel words during runtime. The thesis proposes two strategies for incrementally extending the task vocabulary: incremental learning without training HMMs and incremental learning by training HMMs. Both strategies require a well-trained set of HMMs and a large word model. For the latter approach it is necessary to retrain HMMs with the newly generated transcriptions. During the learning of novel words in NAMs, only new patterns or pattern pairs need to be added while the previously stored patterns or pattern pairs remain unchanged. Compared to HMMs, where many parameter files required for the application have to be changed and modifications to the lexicon, the language model and training of new subword-unit models are necessary, the hybrid approach makes it possible to easily enlarge the vocabulary by using only a sequence of subword-units from HMMs for the novel word. The ability of adding novel words is an important capability for robots employed in real-world environments.

- **Usage of a priori information on word sequences to be recognized in NAMs**

  A language model plays an important role in speech recognition, especially in large vocabulary continuous speech recognition, to augment the recognition performance. In the hybrid HMM/NAM approach, HMMs are sometimes unable to discern every subword-unit uttered by a speaker. Because of the wrongly recognized subword-units, the

NAM-based word recognition network can generate wrong words or superpositions of several words. But, the presented hybrid approach takes advantage of a priori information on the word sequences to be recognized to recognize wrong words. This is done by taking into consideration the correctly recognized word bigrams.

- **Implementation and integration of the developed approach with a biologically inspired language understanding system on a mobile robot [65]**

  In order to show practicality, the proposed approach has been integrated into a language processing system that is embedded in a mobile robot. The presented hybrid system provides words to a sentence processing system using randomly generated binary neural representations. The sentence recognition system then extracts the semantics of the sentence from the stream of words. It can also resolve the ambiguities on the word level, which can not be solved using the priori information on the word sequences to be recognized, using contextual or syntactical information.

Chapter 11

Conclusions

The presented hybrid speech recognition approach was examined in detail and largely evaluated under various aspects. The approach proved practical and useful and showed promising results. It also reveal various advantages, such as learning novel words during run-time and handling ambiguities.

The application area of the developed hybrid speech recognition approach is very large from simple command control systems with small vocabularies to large vocabulary continuous speech recognition systems. During the application of the hybrid approach to various speech recognition tasks, different subword-unit types can be utilised. The type of subword unit best used is determined according to the size of the vocabulary and its recognition performance. Subword-units smaller in size (e.g. phonemes) could be appropriate for small vocabulary, whereas large vocabulary may require the usage of subword-unit longer in size (e.g. demi-syllables or syllables). The memory usage of neural associative memories (NAMs) in the word recognition and the language model networks depends on the subword-unit type that is used in hidden Markov models (HMMs). The structures of neural associative memories (NAMs) can easily be adjusted to a given speech recognition system by varying the size of the associative memories in terms of the chosen subword-unit type.

For continuous speech recognition systems where words run into each other, it is possible for HMMs to generate noisy (incorrect additional subword units) or incomplete subword unit transcriptions, which are called ambiguities on the subword-unit level. In such cases, NAMs might be suitable for solving the ambiguities that occur in the HMM output. If the address pattern containing the subword-units does not strongly address one unique word pattern in the single word recognition network, a superposition of words is activated using their binary neural representations and the ambiguity is kept on the word level. Another example for ambiguties is homophones. If the hybrid system encounters homophones, it is impossible for the single word recognition network to retrieve one unique word pattern. Therefore, a superposi-

tion of homophones is activated by NAMs. The ambiguity on the word level might be resolved in further processes using additional context information (semantics) or a priori information on the word sequences to be recognized.

The developed hybrid system compares favourably well to HMM-based automatic speech recognition research in the literature. With regards to recognition results, it achieves almost the same or an even more improved performance, especially for large vocabulary continuous speech recognition systems. This achievement is due to the "fault-tolerance" property of NAMs. For these systems, the developed approach yielded the best results by utilising syllables as HMM output subword-units.

From the results, it is seen that the developed hybrid system is more suitable for the speech recognition tasks which contain sentence utterances because the ambiguities on the word level can be solved in the language model network using the additional priori information about the word sequences to be recognized, but less suitable for single word speech recognition tasks because the language model network does not contain any priori information. Thus, the ambiguities on the word level can not be solved using any additional information.

The easy extendability of the task vocabulary during runtime makes the hybrid approach particularly suitable for the speech recognition application where unknown words are likely to be uttered, as it is possible to incrementally learn novel words during runtime. The performance of the hybrid system for novel words highly depends on the performance of well-trained HMMs.

When the presented hybrid system is compared to a pure HMM-based speech recognition system in terms of the computation time, the hybrid system requires more time for recognition than an HMM-based system. The HMM-based subword-unit recognition part of the hybrid system especially takes most of the recognition time although NAMs work quickly. This is due to the complexity of the subword-unit-level search space in HMMs.

For continuous speech recognition tasks, it is sometimes difficult for the hybrid system to determine the word boundaries because there is no word boundary between words, like small pauses. In this case, for a given subword-unit sequence, NAMs usually look for the longest corresponding word. If two small words are adjacent in a sentence and a longer word in the vocabulary is composed of these adjacent words, NAMs retrieve the long word instead of these two short words. This weakness of the hybrid system can be corrected by taking the other recognized words in the sentence into consideration.

These advantages and capabilities make the hybrid HMM/NAM speech recognition approach a promising approach which can be applied for various speech

recognition tasks and offers diverse possibilities for further improvements. The performance of the single word recognition network against ambiguities can be increased with backward connections from the NAM-based language model network and from the language understanding module to the memory WRD in the single word recognition network. A further research on the hybrid HMM/NAM approach could be the application of the presented hybrid system to multi-lingual speech recognition tasks.

Bibliography

[1] *SPHINX Tutorial*. http://www.speech.cs.cmu.edu/sphinx/tutorial.html.

[2] *Syllabification Software*. ftp://jaguar.ncsl.nist.gov/pub/tsylb2-1.1.tar.Z.

[3] J. K. Baker. Stochastic modeling for automatic speech understanding. In *Speech Recognition, Editor R. Reddy, Academic Press, New York, USA*, pages 512–542, 1975.

[4] C. Becchetti and L. P. Ricotti. *Speech Recognition: Theory and C++ Implementation*. Wiley, 2002.

[5] Y. Bengio. A connectionist approach to speech recognition. *International J. Pattern Recognition Artificial Intelligence*, 7(4):647–667, 1993.

[6] Y. Bengio. *Neural Networks for Speech and Sequence Recognition*. International Thomson Computer Press, 1996.

[7] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

[8] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–259, 1992.

[9] H. J. Bentz, M. Hagstroem, and G. Palm. Information storage and effective data retrieval in sparse matrices. *Neural Networks*, 2:289–293, 1989.

[10] H. Bosch and F. J. Kurfess. Information storage capacity of incompletely connected associative memories. *Neural Networks*, 11:869–876, 1998.

[11] H. Bourlard and N. Morgan. Continuous speech recognition by connectionist statistical methods. *IEEE Trans. Neural Networks*, 4(6):893–909, 1993.

[12] H. Bourlard and N. Morgan. *Connectionist Speech Recognition. A Hybrid Approach*. Kluwer Academic Publishers, 1994.

[13] H. Bourlard and C. Wellekens. Links between hidden Markov models and multilayer perceptrons. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:1167–1178, 1990.

[14] J. S. Bridle. Alphanets: a recurrent neural network architecture with a hidden markov model interpretation. *Speech Communication*, 9(1):1167–1178, 1990.

[15] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[16] J. Buckingham and D. Willshaw. On setting unit thresholds in an incompletely connected associative net. *Network*, 4:441–459, 1993.

[17] J. Buckingham and D. Willshaw. Performance characteristics of associative nets. *Network: Computation in Neural Systems*, 3:407–414, 1993.

[18] P. Cosi, Y. Bengio, and R. De Mori. Phonetically-based multi-layered networks for acoustic property extraction and automatic speech recognition. *Speech Communication*, 9(1):15–30, 1990.

[19] R. Cosi, P. Frasconi, M. Gori, and N. Griggio. Phonetic recognition experiments with recurrent neural networks. In *Proceedings of the International Conference on Spoken Language, Canada*, pages 1335–1338, 1992.

[20] K. H. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *Journal Acoustic Society. Am.*, 24(6):637–642, 1952.

[21] R. De Mori. *Spoken Dialogs with Computers*. Academic Press, London, UK, 1998.

[22] Y. Dongsuk. *Robust Speech Recognition using Neural Networks and Hidden Markov Models. PhD Thesis*. PhD thesis, State University of New Jersey, 1999.

[23] R. J. Elliott, L. Aggoun, and J. B. Moore. *Hidden Markov Models: Estimation and Control*. Springer, NewYork, 1995.

[24] Y. Ephraim and N. Merhav. Hidden markov processes. *IEEE Trans. Information Theory*, 48(6):1518–1569, 2002.

[25] T. C. Ervin and J. H. Kim. Speaker-independent speech recognition using an associative memory model. In *Southeastcon '93, Processings IEEE*, 1993.

[26] R. Fay, U. Kaufmann, A. Knoblauch, H. Markert, and G. Palm. Combining visual attention, object recognition and associative information processing in a neurobotic system. *Biomimetic Neural Learning for Intelligent Robots. Intelligent Systems, Cognitive Robotics, and Neuroscience, Springer*, 3575:118–143, 2005.

[27] M. A. Franzini, K. F. Lee, and A. Waibel. Connectionist viterbi training: a new hybrid method for continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, 1990.

[28] P. Frasconi, M. Gori, and G. Soda. Recurrent networks for continuous speech recognition. In *Computational Intelligence 90*, 1990.

[29] B. Graham and D. Willshaw. Capacity and information efficiency of a brain-like associative net. *NISP*, 7:513–520, 1995.

[30] B. Graham and D. Willshaw. Improving recall from an associative memory. *Biological Cybernetics*, 72:337–346, 1995.

[31] B. Graham and D. Willshaw. Capacity and information efficiency of the associative net. *Network:Comput. Neural Systems*, 8:35–54, 1997.

[32] P. Haffner, M. A. Franzini, and A. Waibel. Integrating time alignment and neural networks for high perormance continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 105–108, 1991.

[33] A. Hämäläinen, L. Boves, and J. De Veth. Syllable-length acoustic units in large-vocabulary continuous speech recognition. In *SPECOM-2005*, pages 499–502, 2005.

[34] A. Hämäläinen, J. De Veth, and L. Boves. Longer-length acoustic units for continuous speech recognition. In *Proceedings EUSIPCO*, 2005.

[35] J. B. Hampshire and A. H. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Trans. Neural Networks*, 1(2):216–228, 1990.

[36] D. O. Hebb. *The Organization of Behaviour*. New York:Wiley, 1949.

[37] M. M. Hochberg, S. J. Renals, A. J. Robinson, and D. J. Kershaw. Large vocabulary continuous speech recognition using a hybrid connectionist-HMM system. In *Processing of CSLP*, pages 1499–1502, 1994.

[38] H. W. Hon and K. F. Lee. Recent progress in robust vocabulary independent speech recognition. In *Proc. DARPA Speech and Natural Language Processing Workshop*, pages 258–263, 1991.

[39] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Science*, pages 79:2554–2558, 1982.

[40] J. J. Hopfield. Neurons with graded pesponse have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Science*, pages 81(10):3088–3092, 1984.

[41] J. J. Hopfield and D. W. Tank. Computing with neural circuits. *Science*, 233:625–633, 1986.

[42] H. Iwamida, S. Katagiri, and E. McDermott. Speaker-independent large vocabulary word recognition using an LVQ/HMM hybrid algorithm. In *International Conference on Acoustics, Speech anf Signal Processing*, pages 553–556, 1991.

[43] C. S. Jang and C. K. Un. A new parameter smoothing method in the hybrid TDNN/HMM architecture for speech recognition. *Speech Communication*, 19(4):317–324, 1996.

[44] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of IEEE*, 64(4):532–556, 1976.

[45] F. Jelinek, F. Bahl, and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Trans. Information Theory*, 21(3):250–256, 1975.

[46] J. C. Junqua and J. P. Haton. *Robustness in Automatic Speech Recognition: Fundamentals and Applications*. Kluwer Academic Publishers, Boston, USA, 1996.

[47] Z. Kara Kayikci, H. Markert, and G. Palm. Neural associative memories and hidden markov models for speech recognition. In *IJCNN 2007 Proceedings*, 2007.

[48] Z. Kara Kayikci, H. Markert, and G. Palm. Speech recognition using neural associative memories and hidden markov models. In *Technical Report, Promotionskolleg, University of Ulm*, 2007.

[49] Z. Kara Kayikci and G. Palm. Word recognition and incremental learning based on neural associative memories and hidden markov models. In *Proceedings of 16th ESANN*, pages 119–124, 2008.

[50] Z. Kara Kayikci and G. Palm. Word recognition and learning based on associative memories and hidden markov models. *International Journal of Intelligent Technology*, 3(1):19–23, 2008.

[51] Z. Kara Kayikci, D. Zaykovskiy, H. Markert, W. Minker, and G. Palm. Distributed architecture for speech controlled systems based on associative memories. In *Mathematical Analysis of Evolution, Information, and Complexity*. Wiley.

[52] D. Kimber, M. A. Bush, and G. N. Tajchman. Speaker-independent vowel classification using hidden markov models and LVQ2. In *International Conference on Acoustics, Speech anf Signal Processing*, pages 497–500, 1990.

[53] A. Knoblauch. *Synchronization and Pattern Separation in Spiking Associative Memories and Visual Cortical Areas. PhD Thesis*. PhD thesis, University of Ulm, 2003.

[54] T. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, pages 353–359, 1972.

[55] T. Kohonen. *Associative Memory: a System Theoretic Approach*. Berlin:Springer-Verlag, 1977.

[56] T. Kohonen. *Self-organization and Associative Memory*. Berlin:Springer-Verlag, 1983.

[57] T. Kohonen. *Learning Vector Quantization for Pattern Recognition*. University of Technology, Espoo, Finland, 1986.

[58] B. Kosko. Birectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:49–60, 1988.

[59] C. Lee and L. Rabiner. A frame-synchronous network search algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1649–1658, 1989.

[60] H. C. Leung and V. W. Zue. Phonetic classification using multi-layer perceptrons. *Acoustics, Speech, and Signal Processing*, 2(1):525–528, 1990.

[61] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[62] E. Levin. Word recognition using hidden control neural architecture. In *International Conference on Acoustics, Speech and Signal Processing*, pages 433–436, 1990.

[63] H. Markert. *Neural Associative Memories for Language Understanding and Robot Control. PhD Thesis*. PhD thesis, University of Ulm, 2008.

[64] H. Markert, Z. Kara Kayikci, and G. Palm. Sentence understanding and learning of new words with large-scale neural networks. *Artificial Neural Networks in Pattern Recognition. Lecture Notes in Computer Science*, 5064:217–227, 2008.

[65] H. Markert, U. Kaufmann, Z. Kara Kayikci, and G. Palm. Neural associative memories for language understanding and action planning in a robotics scenario. In *Language and Robots. Proceedings of the Symposium*, pages 119–120, 2007.

[66] H. Markert, A. Knoblauch, and G. Palm. Modelling of syntactical processing in the cortex. *BioSystems*, 89:300–315, 2007.

[67] N. Morgan and H. Bourland. Continuous speech recognition using multilayer perceptrons with hidden markov models. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 413–416, 1990.

[68] N. Morgan, Y. Konig, S. L. Wu, and H. Bourlard. Transition-based statistical training for ASR. In *Proceedings of IEEE Automatic Speech Recognition Workshop (Snowbird)*, pages 133–134, 1995.

[69] T. Moudene, R. Sokol, and G. Mercier. Segmental phonetic features recognition by means of neural fuzzy networks and integration in an n-best solutions post-processing. In *Processings of ICSLP*, pages 338–341, 1996.

[70] J. P. Nadal and G. Toulouse. Information storage in sparsely coded memory nets. *Network*, 1:61–74, 1990.

[71] L. T. Niles and H. F. Silverman. Combining hidden markov models and neural network classifiers. In *International Conference on Acoustics, Speech and Signal Processing*, pages 417–420, 1990.

[72] J. J. Odell. *The Use of Context in Large Vocabulary Speech Recognition*. University of Cambridge, UK, 1995.

[73] G. Palm. On associative memories. *Biological Cybernetics*, 36:19–31, 1980.

[74] G. Palm. On the storage capacity of an associative memory with randomly distributed storage elements. *Biological Cybernetics*, 39:125–127, 1981.

[75] G. Palm. *Neural Assemblies*. Berlin:Springer-Verlag, 1982.

[76] G. Palm. Memory capacities of local rules for synaptic modification. a comparitive review. *Concepts in Neuroscience*, 2:97–128, 1991.

[77] G. Palm. On the information storage capacity of local learning rules. *Neural Computation*, 4:703–711, 1992.

[78] G. Palm and F. T. Sommer. Associative data storage and retrieval in neural networks. *Models of Neural Networks III, Springer*, pages 79–118, 1995.

[79] NTIS Order No. PB91-505065. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. National Institute of Standards and Technology Speech Disc 1-1.1, 1990.

[80] L. Rabiner and B. H. Juang. *Fundementals of Speech Recognition*. Prentice Hall, 1993.

[81] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[82] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi. On the application of vector quantization and hidden markov models to speaker-independent, isolated word recognition. *Bell Systems Tech. J.*, 62(4):1075–1105, 1983.

[83] T. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Trans. Neural Networks*, 5(2):298–305, 1994.

[84] T. Robinson and F. Fallside. A recurrent error propagation network speech recognition system. *Comput. Speech Language*, 5(3):259–274, 1991.

[85] A. E. Rosenberg, L. R. Rabiner, J. G. Wilpon, and D. Kahn. Demisyllable based isolated word recognition. *IEEE Trans. on Acoustic, Speech, and Signal Proc.*, 31:713–726, 1983.

[86] J. Rottland, Ch. Neukirchen, D. Willett, and G. Rigoll. Large vocabulary speech recognition with context dependent MMI-connectionist/HMM systems using the WSJ database. In *Processing of 5th European Conference on Speech Communication and Technology*, 1997.

[87] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modelling for acoustic-phonetic modeling of continuous speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 1205–1208, 1985.

[88] R. Schwartz, L. Nguyen, F. Kubala, G. Chou, G. Zavaliagkos, and J. Makhoul. On using written training data for spoken language modeling. In *Proceedings of the workshop on Human Language Technology*, pages 94–98, 1994.

[89] F. Schwenker, F. T. Sommer, and G. Palm. Iterative retrieval of sparsely coded associative memory patterns. *Neural Networks*, 9:445–455, 1996.

[90] A. Sethy and S. Narayanan. Split-lexicon based hierarchical recognition of speech using syllable and word level acoustic units. In *Proceedings ICASSP*, pages 772–776, 2003.

[91] E. Singer and R. P. Lippman. A speech recognizer using radial basis function neural networks in an HMM framework. In *International Conference on Acoustics, Speech anf Signal Processing*, pages 629–632, 1992.

[92] T. Sloboda and A. Waibel. Dictionary learning for spontaneous speech recogition. In *In Proceedings of the International Conference on Spoken Language Processing*, pages 2328–2331, 1996.

[93] F. T. Sommer and G. Palm. Improved bidirectional retrieval of sparse patterns stored by hebbian learning. *Neural Networks*, 12:281–297, 1999.

[94] K. Steinbuch. Die lernmatrix. *Kybernetik*, 1:36–45, 1961.

[95] J. Tebelskis, A. Waibel, B. Petek, and O. Schmidbauer. Continuous speech recognition using linked predictive networks. *Advances in Neural Information Processing Systems*, 3:199–205, 1991.

[96] E. Trentin and M. Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37:91–126, 2001.

[97] M. V. Tsodyks and M. V. Feigelman. The enchanced storage capacity in neural networks with low activity level. *Europhysics Letters*, 6:101–105, 1988.

[98] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1:39–46, 1989.

[99] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 37:328–339, 1989.

[100] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[101] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young. Large vocabulary continuous speech recognition using HTK. In *Proceedings Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 125–128, 1994.

[102] Y. Yan, M Fanty, and R. Cole. Speech recognition using neural networks with forward-backward probability generated targets. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 3241–3244, 1997.

[103] S. Young and et al. *The HTK Book (for HTK Version 3.4)*. University of Cambridge, UK, 2006.

[104] S. R. Young and W. Ward. Learning new words from spontaneous speech. In *IEEE International Conference on Acoustic, Speech, and Signal Processing*, pages 2:590–591, 1993.

[105] G. Zavaliagkos, Y. Zhao, R. Schwartz, and J. Makhoul. A hybrid segmental neural net/hidden markov model system for continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):151–160, 1994.

Appendix A

Sentences in the MirrorBot Speech Data

The sentences are spoken without prepositions. The missing prepositions are most noticeable for the put command, where a command in the robot's language, e.g. "Bot put apple plum", means "Bot put the apple to the plum". "Bot" is the name of the robot used in the MirrorBot project. The simple command sentences used in the MirrorBot project are as follows:

Bot show pepper
Bot lift orange
Bot touch brown dog
Bot stop
Bot drop brown nut
Bot go wall
Bot turn body left
Bot go yellow tangerine
Bot turn body left
Bot lift black nut
Bot turn head right
Bot pick brown banana
Bot turn body right
Bot lift black ball
Bot touch table
Bot touch red lemon
Bot put black dog
Bot put orange tangerine
Bot touch cup
Bot stop
Bot put blue plum
Bot drop orange
Bot pick green apple

Bot move body backward
Bot pick blue plum
Bot move body forward
Bot touch red lemon
Bot move body backward
Bot show orange pepper
Bot turn head down
Bot show white cat
Bot turn head up
Bot put blue wall
Bot turn head up
Bot drop white cup
Bot move body forward
Bot pick yellow banana
Bot lift white cat
Bot show ball
Bot go table
Bot turn head down
Bot move body forward
Bot drop green apple
Bot show orange pepper
Bot turn head down
Bot put red plum yellow lemon
Bot put red plum lemon
Bot lift wall
Bot show green apple
Bot put ball orange orange
Bot put orange orange orange plum
Bot turn body left
Bot drop orange
Bot pick brown banana
Bot pick green apple
This is plum
Bot pick yellow banana
Bot drop white cup
Bot drop brown nut
Bot put orange tangerine
Bot go table
Bot lift orange
This is pepper
Wall is red
Bot lift red plum
Bot put wall orange orange

Bot drop green apple
Bot lift black nut
Bot turn head up
Bot lift black ball
Bot lift white cat
Bot drop orange
Bot pick blue plum
Bot move body backward
Bot move body backward
Bot show white cat
Bot touch red lemon
Bot turn head up
Bot move body forward
Bot touch cup
Bot show green wall
Bot pick orange
Bot show pepper
where is plum
Bot put blue plum
Bot show red plum
Bot go yellow tangerine
Bot touch brown dog
Bot put orange blue wall
Bot stop
This is green
Bot put black dog
Bot touch table
Bot go wall
Bot show ball
Bot turn head down
This is cup
Bot show plum
Bot put apple plum
Bot lift ball
Bot lift orange
Bot turn body left
Bot turn head right
Bot turn body right
Bot touch red lemon

Appendix B

Information Theory

B.1 Basic Information Theory

B.1.1 Information of Random Variables

Let X be a random variable on $\Omega = \{w_1, w_2, ...\}$, and $p_i := p[X = w_i]$ the probability that X has value $w_i \in \Omega$. Then the information of X is defined as

$$I(X) = \sum_{i \in \Omega} - p_i \log_2 p_i. \tag{B.1}$$

$-\log_2 p_i$ indicates the amount of information content (in bit) associated with the event $\{X = w_i\}$. $I(X)$ is the average amount of information content of X over the entire value space. In other words, $I(X)$ also measures the uncertainty of the possible value of X.

Let Y be another random variable on Ω. We define the conditional information of X given that $Y = w_j$ as

$$I(X|Y = w_j) = -\sum_{i \in \Omega} p[X = w_i|Y = w_j] \log_2 p[X = w_i|Y = w_j], \tag{B.2}$$

where

$$p[X = w_i|Y = w_j] = p[X = w_i \wedge Y = w_j] / p[Y = w_j]. \tag{B.3}$$

Then, in the same way, the *conditional information* $I(X|Y)$ of X given Y can be defined as

$$I(X|Y) = \sum_{j \in \Omega} p[Y = w_j] \cdot I(X|Y = w_j), \tag{B.4}$$

which determines the average information (or uncertainty) that remains in X, given that one knows Y. From this interpretation, it is reasonable to assume

that $I(X|Y)$ is generally less than $I(X)$, because knowing Y can not increase the uncertainty of X. Correspondingly, it can be shown that

$$I(X,Y) = I(Y) + I(X|Y) \leq I(X) + I(Y), \tag{B.5}$$

where equality holds if X and Y are independent.

B.2 Transinformation

The transinformation or mutual information between two random variables X and Y is defined as

$$T(X,Y) = I(X) + I(Y) - I(X,Y) = I(X) - I(X|Y) \geq 0. \tag{B.6}$$

This quantity measures the information about X that is contained in Y, or the amount by which knowing Y reduces the uncertainty of X. For example, if X and Y are independent, then $T(X,Y) = 0$.

The *transinformation rate* between two processes $(X^\mu)_{\mu=1,2,...}$ and $(Y^\mu)_{\mu=1,2,...}$ is givens as

$$T((X^\mu),(Y^\mu)) = \lim_{M\to\infty} \frac{1}{M} \cdot T(X^1,.....,X^M;Y^1,....,Y^M). \tag{B.7}$$

B.3 Channel Capacity

The information transmission from a stationary source process $(X^\mu)_{\mu=1,2,...}$ to a target $(Y^\mu)_{\mu=1,2,...}$ can be described by a *channel*. It is identified with a *transition probability* $P : \Omega^N \hookrightarrow \Omega^N$. If Ω is finite and P is memory free, then P can be defined with a matrix of transition probabilities

$$p_{ij} = prob[Y = j|X = i]. \tag{B.8}$$

The *channel capacity* C_p is defined for a channel $P : \Omega^N \hookrightarrow \Omega^N$ as the maximal achievable transinformation rate

$$C_p = \sup_P T(X;Y). \tag{B.9}$$

B.4 Binary Channels

In the binary case $\Omega = \{0, 1\}$, a random variable X on Ω with $p_1 := prob[X = 1]$ the information $I(X)$ equals

$$I(p_1) := -p_1 \cdot log_2 p_1 - (1 - p_1) \cdot log_2(1 - p_1) \tag{B.10}$$

$$\approx \begin{cases} -p_1 \cdot log_2 p_1 & , p_1 \ll 0.5 \\ -(1 - p_1) \cdot log_2(1 - p_1) & , 1 - p_1 \ll 0.5 \end{cases} . \tag{B.11}$$

Note the symmetry $I(p_1) = I(1 - p_1)$, and that $I(p_1) \to 0$ for $p_1 \to 0$ (and $p_1 \to 1$).

A binary channel without memory is already determined by the two error probabilities p_{01} (false one) and p_{10} (false zero). For two binary random variables X and Y, where X is transmitted over a binary channel with the result Y, we can write

$$I(Y) = I(p_1(1 - p_{10}) + (1 - p_1)p_{01}) \tag{B.12}$$

$$I(Y|X) = p_1 \cdot I(p_{10}) + (1 - p_1) \cdot I(p_{01}) \tag{B.13}$$

$$T(X; Y) = I(Y) - I(Y|X). \tag{B.14}$$