TU Kaiserslautern & DFKI
Image Understanding and Pattern Recognition
Prof. Dr. Thomas Breuel

Seminar Pattern Recognition im WS 2005/06

Sequence Learning and Speech Recognition

Martin Krämer

Matrikelnummer 346493

18.05.06

Betreuer: Daniel Keysers

Contents

Abstract

This paper describes application possibilities for statistical methods and particulary hidden Markov models in the domain of sequence learning after treating the required basics. Especially the task of natural language processing is treated by elaborating on speech recognition and speech translation. Furthermore a network intrusion detection system to detect complex and coordinated Internet attacks is presented briefly to illustrate the universality in the concept of sequence learning.

1 Fundamentals

In this section a brief overview of the needed fundamentals to understand the further discussed concepts is given and the domain of sequence learning is outlined.

Further literature references are given at adequate spots to permit the consultation of more detailed background information.

1.1 Notation Overview

Some syntax rules for better comprehension:

- P denotes probabilities whereas p is used for probability densities.
- Calligraphic letters like $\mathcal{V}$ are employed for sets.
- Capital letters like N are utilized for constant values.

1.2 Statistical Basics

Here we define stochastic processes and Markov chains – a time-restricted stochastic process. Both concepts are essential to understand the principle of hidden Markov models.

Some very elementary statistical principles – that means Bayes' rule, random variables, probability distributions and similar things - are assumed to be familiar.

1.2.1 Stochastic Processes

A *stochastic process* constitutes a series of random variables $q_1, q_2, \ldots$ which adopt values Q_t out of a discrete or continous domain according to specific probability distributions. The distribution function of the Q_t may depend on the current state q_t and its predecessors $q_1, q_2, \ldots, q_{t-1}$.

The stochastic process is called *stationary* if its behaviour is independent of the absolute value of time t.

It is called *causal* if additionally the distribution of Q_t is only a function of the previous states $q_1, q_2, \ldots, q_{t-1}$, i.e. it does not rely on states in the future.

$$P(q_t = Q_t | q_1 = Q_1, q_2 = Q_2, \ldots, q_{t-1} = Q_{t-1})$$

or simplified

$$P(Q_t|Q_1, Q_2, \ldots, Q_{t-1})$$

is the probability distribution for a discrete, stationary and causal stochastic process. [7]

1.2.2 Markov Chains

Stationary and causal stochastic processes that only rely on a finite number n of past states are named n^{th}-order *Markov chains* and satisfy the so-called *Markov property*:

$$P(Q_t|Q_1, Q_2, \ldots, Q_{t-1}) = P(Q_t|Q_{t-n}, \ldots, Q_{t-1})$$

In the case of a stochastic process that is further constrained to only the last state

$$P(Q_t|Q_1, Q_2, \ldots, Q_{t-1}) = P(Q_t|Q_{t-1})$$

we speak of first-order Markov chains.

It should be noted that higher order Markov chains are semantically equivalent to first-order Markov chains because state extensions can be coded into a single state. Although the higher order chains may be more convenient to use in some cases [7].

Furthermore they can be expressed via finite state machines.

1.3 Parameter Estimation

A method to supply proper parameters for a statistical model with the aid of training data are needed.

Subsequently we present two common methods achieving this: the maximum-likelihood-estimation that maximizes the probability of a model on specific training data and the maximum-a-posteriori-estimation which additionally considers the initial estimates.

1.3.1 Maximum-Likelihood-Estimation

The *maximum-likelihood-estimation* (MLE) tries to determine estimates $\hat{\theta}$ for statistical parameters based on a sample $\omega = \{x_1, x_2, \ldots, x_T\}$, such that the probability is maximized for the observed data.

$L(\theta|\omega) := p(x_1, x_2, \ldots, x_T|\theta) = \prod_{t=1}^{T} p(x_t|\theta)$ should be maximized using the likelihood functional $L(\theta|\omega)$.

The aim of MLE is to maximize the value of the likelihood functional for a given model and set of samples ω dependent on θ:

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta} L(\theta|\omega)$$

MLE delivers an optimal solution for an infinite number of samples, i.e. $\lim_{T\to\infty} \hat{\theta}_{\mathrm{ML}} = \theta^*$ [7].

1.3.2 Maximum-A-Posteriori-Estimation

In the *maximum-a-posteriori-estimation* (MAPE) the optimization criterion is the a-posteriori-probability of the model parameters based on a sample $\omega = \{x_1, x_2, \ldots, x_T\}$:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|\omega) \overset{\text{Bayes}}{=} \arg\max_{\theta} \frac{p(\omega|\theta)p(\theta)}{p(\omega)}$$

Hence $p(\omega)$ is independent of the sought model parameters, it can be neglected during maximization:

$$\hat{\theta}_{\text{MAP}} \equiv \arg\max_{\theta} p(\omega|\theta)p(\theta)$$

The MAPE is superior (but more complex as well) to the MLE if only a small number of samples is given. This is caused by the inclusion of the initial estimates $p(\theta)$ in the MAPE calculation which are ignored by the MLE. [7]

1.3.3 Expectation-Maximization-Algorithm

An *expectation-maximization-algorithm* (EM-algorithm) is a meta-algorithm to derive specific algorithms, like the Baum-Welch-algorithm for hidden Markov models, that find estimates of parameters in probabilistic models which depend on unobserved variables (latent variables). They work in a two-step scheme: first the expectation step is accomplished where the expected values of the latent variables are computed, then the maximization step occurs where the estimates are evaluated according to the model parameters and the data, assuming the previously calculated expected values for the latent variables.
A very thorough explanation can be found in [7, 9].

Formal: Let y be the observed variables, z be the latent variables, p be the joint model of complete data with parameters θ: $p(y, z|\theta)$. Now an EM-algorithm iteratively improves the initial estimate θ_0 through the construction of new estimates $\theta_1, \ldots, \theta_N$ [19]:

$$\theta_{n+1} = \arg\max_{\theta} \sum_{z} p(z|y, \theta_n) \log p(y, z|\theta) = \arg\max_{\theta} Q(\theta)$$

It is proven that an EM-algorithm converges to a local maximum of the observed data's likelihood function.

1.4 Basic Terms

Here we sketch some concepts or linguistic terms used throughout the following chapters.

- *Q-filter bank:* a filter bank in which the energy sensitivity follows a logarithmic relationship in each channel.
- *constant Q-filter bank:* a Q-filter bank in which the ratio of the center frequencies of adjacent channels is constant and the filter bandwidth is proportional to the center frequencies.

- *phoneme:* it is the smallest distinguishable unit of speech without semantic information, i.e. changing a phoneme inside a word changes the meaning but a phoneme in itself has no meaning [23].

  Phonemes are no sounds in themselves, but they have to be made audible through allophones (→ realization of phonemes) and depending on the language the same phonemes are mapped onto different allophones. Moreover important is the fact that one word may have several representations in phoneme sequences, so for instance 'the' corresponds to 'DH/AH' as well as to 'DH/IY' depending on its context.

- *coarticulation:* this describes the assimilation of speech sounds because of the place of articulation, i.e. adjacent speech sounds distort the original spectral content. As an example the 'n' in 'tenth' is pronounced dental because of the 'th' while it normally has an alveolar place of articulation [18, 13].
  It is particularly prevalent in spontaneous speech where the speaker does not care for enunciation.

2 Hidden Markov Models

A *hidden Markov model* (HMM) is a two-stage stochastic process. The first layer consists of a discrete Markov process with a finite number of states probabilistically describing the state transitions. In the second layer an emission O_t is generated for every point in time t [20].

Only that series of emissions is observable but the sequence of states is unknown – hence the O_t are called observation sequence and the Markov models are called 'hidden'.

HMMs are particularly suitable for the modelling of time-oriented processes and have therefore been extensively used in natural speech processing systems with great success but they are applicable in other areas of sequence learning as well.

2.1 Definition

A first-order HMM λ is completely described by [7]:

- a finite set of states $\{q_i | 1 \leq i \leq N\}$,
- a matrix A of state-transition probabilities

$$A = \{a_{ij} | a_{ij} = P(q_t = j | q_{t-1} = i)\},$$

- a vector π of state-start probablities

$$\pi = \{\pi_i | \pi_i = P(q_1 = i)\},$$

- the state-specific emission distributions

$$\{b_j(o_k) | b_j(o_k) = P(O_t = o_k | q_t = j)\}$$

or

$$\{b_j(x)|b_j(x) = p(x|q_t = j)\}.$$

Discrete emissions, i.e. $O_t \in \{o_1, o_2, \ldots, o_M\}$, lead to discrete HMMs where the $b_j(o_k)$ are discrete probability distributions that can be integrated into a matrix B of emission probabilities with

$$B = \{b_{jk}|b_{jk} = P(O_t = o_k|q_t = j)\}.$$

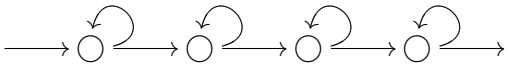
Vector-like emissions $x \in \mathbb{R}^n$ lead to continous HMMs where the $b_j(x)$ are continous density functions with

$$b_j(x) = p(x|q_t = j).$$

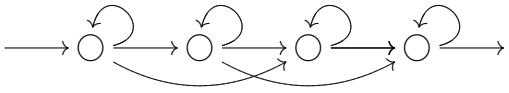
2.2 Types of Models

An HMM can be classified according to its allowed state transitions:

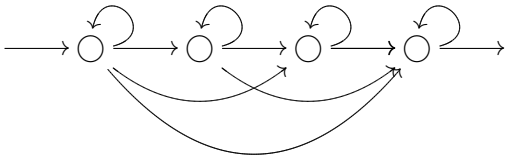
- *linear:* only state transitions to the current or next state are allowed



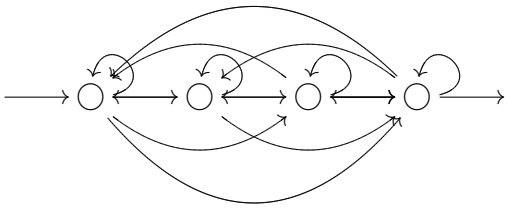
- *Bakis:* like a linear model but the jump over single states is possible



- *left-right:* all state transitions pointing to the current or following states are allowed



- *ergodic:* all possible state transitions are allowed



Jumps to past states are not needed in many cases because of the time-linear input data of the application domain. Generally it can be said that the more state transitions are allowed the more parameters have to be trained and therefore a compromise between flexibility and usability has to be made.

A more elaborate illustration can be found in [15, 7].

2.3 Problems and Solutions

Here we want to give a short summary of the three basic problems arising with the use of HMMs. Since this is not the main theme of this paper only a short outline is given – for details the reader is referred to [15, 3, 7].

1. *Evaluation Problem:* Given the observation sequence $O = O_1 O_2 \cdots O_T$, and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

   The naive approach which consists of considering all possible paths is unfeasible since for N states and T time-slots it needs $2TN^T$ computations. Since this technique ignores the fact that there are only N possible states at any moment, we can improve the complexity by merging every possible state sequence into a corresponding state. This method is called *Forward-Backward-procedure* and its complexity is N^2T.

2. *Decoding Problem:* Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$, which is optimal in some meaningful way, i.e. best "explains" the observations?

   Here a dynamic programming solution called *Viterbi* (MLSE) exists which is very well applicable. In a few words Viterbi finds the most probable hidden state sequence for a given observed state sequence by traversing through a Trellis with traceback information and after completion it traces back to the start determining the overall best path. This process can be sped up with beam search where a heuristic estimation function is used, however not only the one but the m most probable pathes are passed through [9].

3. *Optimization Problem:* How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

   This is the most complex of the three basic problems. Because of complexity no analytic method is feasible, hence a locally maximizing method like the *Baum-Welch-algorithm* has to be used. It is an instance of the EM-algorithm and calculates the maximum likelihood estimates and posterior mode estimates for the given parameters of an HMM iteratively.

2.4 Implementation Issues

Here we want to give a short overview of different aspects to be considered when implementing HMMs [15]:

1. *Scaling:* The reestimation procedure of HMMs includes a term that heads exponentially to zero, practically exceeding the precision range of essentially every real machine. Therefore a scaling coefficient is used.

2. *Multiple Observation Sequences:* To train HMMs in a left-right model we have to use multiple observation sequences for reliable estimates as the transient nature of

states within the model only allows a small number of observations for each state. The key is to remove the scaling factor from each term before summing.

3. *Initial Estimation of HMM Parameters:* The problem is how to choose the initial estimates of the HMM parameters so that the local maximum is close to the global maximum of the likelihood function. To this problem there is no straightforward answer and only empirical experience has been established.

4. *Effects of Insufficient Training Data:* The finite length of observation sequences used for training is a problem in practice. Possible solutions are the enlargement of the training set, model size reduction and interpolation of different parameter estimates although not all methods are always applicable.

5. *Choice of Model:* This fully depends on what the HMM should model and has to be decided from case to case.

3 Sequence Learning

Sequence learning either describes the process of estimating following sequence elements given the current one and its predecessors or the analysis of a sequence. No limitations are made on the types of data that are contained within the sequence.

These tasks can be accomplished via statistical methods, the usage of HMMs and the incorporation of techniques from other scientific areas.

It should be remarked that the common problem description permits utilization in many other application areas not described here. As the definition is rather general, all data that are representable in a sequence are appropriate. Especially systems with time-discrete data are particulary suitable because vectors consisting of all data values of the same point in time constitute a sequence of data.

In the following subsections some possible applications of sequence learning techniques are presented and a few chosen examples are discussed in more detail in the next sections.

3.1 Anomaly Detection

Anomaly detection describes the recognition process of suspicious or somehow deviant information out of a usually rather big block or stream of information.

Normally the problem is to provide rules classifying data as either abnormal or normal, which proves to be rather tedious if possible at all. Using HMMs we are able to tackle it another way – first we gather a large amount of data to train the system and then it learns to classify on its own without specified rules.

As the definition of anomaly detection is very general, there are many different application areas. Here we picked out a network intrusion detection system (NIDS) [12, 4] for closer inspection.

Other possible uses not closer inspected in this paper include for instance visual recognition or visual surveillance. Some background information for the interested reader about the former task can be found in [2, 6, 16] whereas the latter one is treated in [10].

3.2 Bioinformatics

Bioinformatics or computational biology combines techniques from applied mathematics, informatics, statistics and computer science to solve biological problems.

Research areas in this field include sequence alignment, gene finding, genome assembly, proteine structure alignment, proteine structure prediction, prediction of gene expression, protein-protein interactions and the modeling of evolution [17].

The reader is referred to [5, 8] for some recent applications of HMMs in computational biology as they are not closer reviewed in this paper.

3.3 Natural Language Processing

Natural language processing (NLP) is a subfield of artificial intelligence and linguistics. It studies the problems inherent in the processing and manipulation of natural language and is devoted to making computers "understand" statements written or spoken in human languages [21].

We will discuss the various sections of NLP in-depth, that are speech recognition (spoken to written) and speech translation (between two natural languages) as well as present the current state of those techniques.

The task of speech understanding (natural to semantic) needs no exclusive treatment as it is a special form of speech translation and similar methods can be applied. It can be seen as the translation of natural speech into the formal language that is used for semantic representation or database requests [11].

There exist two common approaches to this problem area: while the one by Noam Chomsky formalizes a language and works rule-based (also called the linguistic approach), the statistical one tries to maximize the probability of correct detection through established procedures like MAPE. As the statistical method seems to deliver much better results [11] – without the problematic usage, i.e. hard setup of rules - we will restrict ourselves to the inspection of such a system in this paper.

It is worth a comment that experiments on audio-visual speech recognition have been conducted in [1] with asynchronous HMMs leading to very interesting results: without the presence of noise there are almost no differences in the word-error-rate of audio-only (2.9%) vs. audio-visual (4.8%) recognition in their experiments. But as soon as the noise reaches a significant level the audio-visual system clearly performs better with a word-error-rate of 41.1% compared to the 79.1% of the audio-only approach.

4 Detection of Multi-Staged Internet Attacks

A NIDS monitors network traffic and tries to detect ongoing malicious activity [22], i.e. Denial-of-Service attacks, port scans, traffic monitoring, transported viruses/malware or similar actions. Normally systems work according to a set of rules to classify different types of intrusion. The inherent problem here is that a typical sophisticated attack will stretch across several stages, i.e. probing, consolidation, exploitation and compromisation.

Due to the temporal seperation of these stages, potentially incomplete data, interchangeable actions, random/noisy action sequences and the low number of examples[1] classical systems are at best able to identify such a single stage of attack because of the tedious manual definition of appropriate classification rules (if possible at all[2]). Machine learning techniques have to be applied to find correlations between the different stages and enable the detection of fully coordinated multi-staged attacks [12].

In the following subsections we want to give a short overview of a system using HMMs to recognize such complex attacks as presented in [12] by first elaborating on its design and then presenting some operational results obtained. The authors state that such a system is particulary interesting because it "operates a level higher", i.e. it uses packet filter alarms (from programs like Snort) as input instead of raw packet data, than normal NIDS and shows promise to deliver some interesting results as soon as it reaches suitability for common application.

As this seemed more absorbing than the inspection of classical NIDS using machine learning techniques they have been skipped in this paper and the reader is for instance referred to [4] where more information on a typical NIDS working with HMMs can be found.

4.1 System Architecture

See Figure 1 for the block diagram illustrating the following description.
The flowing network traffic is analysed by the network sensors and they declare alarms on possible intrusion attempts, which are stored in the alarm database for further processing. In the data prefiltering step redundant data is removed from the input but although the abstraction through usage of network sensors instead of raw packet data already extremely simplifies this task[3] two new problems are introduced.

On the one hand network sensors are designed to prefer false positves to false negatives leading to probably wrong alerts and on the other hand repetitive actions (like repeated port probes) that generate a series of alarms may obfuscate the real alarm sequence. The

[1] The Rare Data Problem: As the most skilled types of attacks only happen rarely and are detected even less, machine learning techniques have difficulties to adapt to them due to missing training data.

[2] In fact it seems absolutely impossible to define rules in a way that they include all (or even many) types of specific multi-staged attacks from a specific level of complexity on, i.e. the series of correlated actions is long enough. There are just too many possibilities to permutate action sequences, to insert/remove redundant actions or that distinct actions get interwoven to define generally working rules.

[3] The normal network traffic – for instance connection setups, disconnections, legal browser requests - is completely ignored in this model.

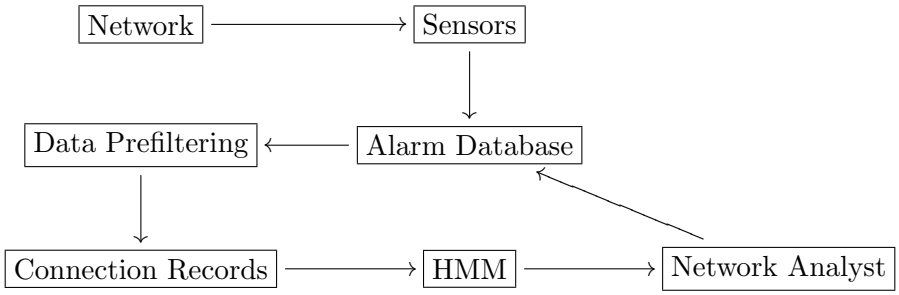


Figure 1: Multi-Staged Attack Detection System – Block Diagram [12]

latter problem is dealt with by just replacing such sequences by their period whereas the first one still remains open.

Now the data consists of connection records including the source/destination IP address and the ordered sequence of alerts during a specified duration of 24 hours. In the HMM classification module interested examples are highlighted for further human evaluation and false classifications are manually corrected by a network analyst to update the system by sending them back to the alert database.

4.2 Usage of HMMs

Observables of an HMM correspond to the different alarms of a given example in this system. Those examples include all alarms between two specified hosts in a 24-hour interval; so an assignment between a set of observables and a specific attack can be made. The task of the HMM is to determine the most probable type of attack to a given sequence of alarms. Additionally the calculation of the likelihood of a specific attack is made possible. A seperate HMM gets generated for each classification category, i.e. type of attack, since HMMs are designed to simulate a single category. Furthermore the designed HMMs have in principle an ergodic structure, so all possible state transistions are generally allowed.

The probabilistic transitions in the HMM match with the variations of an attack sequence and tool choices. Every possible state sequence traversing the HMM from start to end describes a more or less probable way to accomplish the type of attack specified by the specific HMM. The state transition probabilities are generated through calculation of the relative frequencies of alarm sequences in the training data. Such a model can then be used to acquire the most probable attacker's intentions just by evaluating the alarm sequences, which is very valuable.

4.3 Empirical Results

In [12] a performance comparison of this system, implemented with a slightly modified version of HMMLib (a HMM library in Java), C4.5 decision trees and neural networks (both out of the WEKA machine learning library) has been conducted. The definition of performance is done via evaluating the quotient of examples classified as the correct type of attack and the number of overall examples.

The HMM approach always performed at least as good as C4.5[4]. Neural networks only performed slightly better than guessing, although they required more training.

According to [12] it can be said that HMMs provide good detection chances (92.5%). They state that categories whose training data consists of few examples perform worse and practically determined around ten examples as a reasonable lower limit (on required training data for a type of attack) to provide adequate classifications.

5 Speech Recognition

The task of a speech recognition system is to map a piece of sampled speech onto a spoken word sequence. We will elaborate the process in detail by breaking the system up into different functional blocks and discussing them piece-wise [13, 11]. Furtermore the usage of HMMs, methods to evaluate a system's performance and advanced speech recognition-systems' (ASR-systems[5]) evolution in the last time are treated.

ASR-systems or large-vocabulary speech recognition systems are systems capable of understanding a naturally spoken language in contrast to systems which only understand a very limited subset of a language like airline reservation systems. When talking about speech recognition systems in this paper in fact ASR-systems are meant.

5.1 Probabilistic Model

The system tries to maximize the probability of the recognized word sequence $\hat{w}_1^N$ given an acoustic sequence y_1^T using Bayes' rule where w_1^N is a sequence of words out of the lexicon [13]:

$$\hat{w}_1^N = \arg\max_{w_1^N} p(w_1^N|y_1^T) \equiv \arg\max_{w_1^N} p(y_1^T|w_1^N)p(w_1^N)$$

So really meaningless sentences, i.e. they would practically never occur in natural language, are excluded from recognition and common sentences get preferred over similar sounding but less probable sentences, which ultimately improves the hit/miss-ratio.

In contrast a MLSE would ignore these initial estimates and only the audible features would correspond to the classification process. That would render the process to gather the initial estimates unnecassary and thus save quite some effort but the performance would be severly constrained – so this technique seems rather uncommon if applied at all.

5.2 System Architecture

See Figure 2 for the block diagram illustrating the following paragraphs.

[4] 10 examples: 45% vs. 30% – 50 examples: 79% vs. 67% – 400 examples: both $\approx$ 92%

[5] ASR is more commonly used to abbreviate 'automatic speech recognition' but in this paper we mean 'advanced speech recognition' to emphasize the ability to understand a very large dictionary spoken by any speaker in natural intonation as defined in [13].

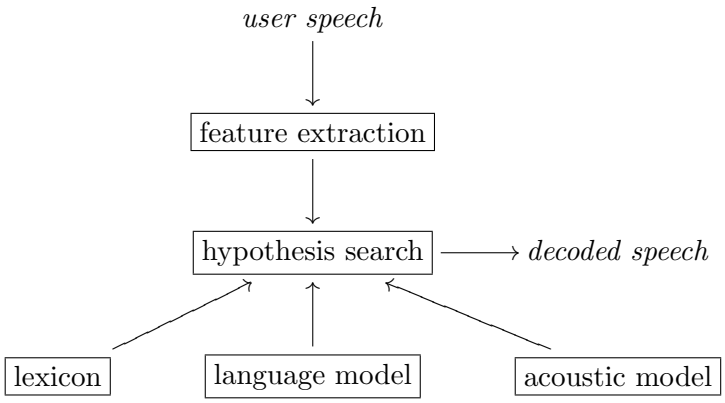


Figure 2: Speech Recognition System – Block Diagram [13]

5.2.1 Feature Extraction

In this component a multi-dimensional vector is extracted from the sampled speech at equidistant points in time, i.e. 39 parameters every 10 ms – these vectors are called feature vectors or acoustic observations. They are mapped onto phonemes later on, which is why the temporal variation is included in the coding.

Multiple feature vectors can be packed together for compression purposes. Further data reduction is achieved through a model of the human auditory and speech system. Redundancy introduced by the human speech articulators and the human auditory system, as it emulates the behaviour of a constant Q-filter bank, is eliminated.

5.2.2 Lexicon

Here each word to be recognized gets mapped to one or more sequences of phonemes. A commercial lexicon typically covers several tens of thousands of words but a trade-off between size and coverage has to be made. So for domain-specific jargon either a task-aligned system has to be used or a high miss rate is inevitable.
It is assumed that all spoken words are included in the lexicon to simplify the modelling of the system. Approaches to recognize that a spoken word is not contained in the lexicon – called out of vocabulary recognition – exist, but we won't treat them here further.

5.2.3 Language Model

The computation of $p(w_1^N)$, i.e. the absolute probability of occurence of a word sequence, is done by the language model for each possible sequence of words. As this simple approach grows exponentially with growing sequence length it is not applicable for real applications. The heuristic solution is the abstraction to the trigram model where the probability of a current word is only dependent on the last two words:

$$p(w_1^N) = p(w_1)p(w_2|w_1) \prod_{i=3}^{i=N} p(w_i|w_{i-1}, w_{i-2})$$

with

$$p(w_i|w_{i-1}, w_{i-2}) = \frac{N(w_i, w_{i-1}, w_{i-2})}{N(w_{i-1}, w_{i-2})}$$

where N refers to the relative frequency of the associated event.

To achieve a good set of estimates $p(w_i|w_{i-1}, w_{i-2})$ hundreds of millions of words are needed for training. Even then many trigrams of a test subset never occur in the training subset[6]. Therefore it is required to smooth the probability estimates through interpolation of the trigrams', bigrams' and unigrams' relative frequencies:

$$p(w_i|w_{i-1}, w_{i-2}) = \lambda_3 \frac{N(w_i, w_{i-1}, w_{i-2})}{N(w_{i-1}, w_{i-2})} + \lambda_2 \frac{N(w_i, w_{i-1})}{N(w_{i-1})} + \lambda_1 N(w_i)$$

The trigram model and the issue of optimal selection of the λ_i is extensively treated in [9].

5.2.4 Acoustic Model

$p(y_1^T|w_1^N)$, i.e. the probability of a sequence of acoustic observations under the assumption that they have been generated by a specific word sequence, is evaluated by the acoustic model. Since variations in the duration of speech and the spectral content due to coarticulation are inevitable and these are of stochastic nature, HMMs are an optimal choice for its modelling.

Each HMM represents a phoneme and a pass through it matches its articulation whereby durations are modelled by loops in the states of the HMM. At each time frame a probabilistic transition is made and a feature vector is emitted.

This permits the computation of the likelihood that a particular phoneme has generated a given series of feature vectors. As the actual state sequence of the HMM is hidden it has to be summed over all possible ones which is efficiently accomplishable with dynamic programming due to the lack of global dependencies [9].

However, the system has to be trained first and for this purpose we need large amounts of training data, i.e. spoken word sequences with their corresponding feature vectors and recognized word sequences. Then the HMM parameters can be trained to maximize the likelihood of the acoustic observations with MLE.

Formal: Let y_1^T be a sequence of T acoustic observations and w_1^N the correct word sequence with the MLE $\hat{\theta}_{\mathrm{ML}}$:

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta} \log[p_\theta(y_1^T|w_1^N)]$$

Now an HMM is constructed for the correct word sequence w_1^N by generating an HMM for each word by concatenation of the phoneme-representing HMMs first and then the connection of those to produce the final one which recognizes the word sequence afterwards. But as the state sequence of an HMM is unknown, the assumption of passing it in T time

[6] An evaluation by IBM in the 70s led to 23% unknown trigrams according to [9].

frames is problematic. Thus we introduce s_t to describe the hidden state at the point in time t allowing MLE:

$$\arg\max_{\hat{\theta}} \sum_{t=1}^{T} \sum_{s_t} p_\theta(s_t|y_1^T) \log[p_{\hat{\theta}}(y_t|s_t)]$$

This is solved iteratively with an EM-algorithm where the expectation-step can be accomplished via the Forward-Backward-procedure.

Another problem that has to be regarded is coarticulation as it distorts the spectral content of speech depending on its context. Therefore a theoretical approach would require the generation of as many probability densities per phoneme as possible combinations of surrounding phonemes exist. But this is practically not viable because just 50 phonemes would require the modelling of 125,000 probability densities. So the phonemes are clustered in a small number of equivalence classes [13].

5.2.5 Hypothesis Search

The hypothesis search spans the lexicon and the results of the language/acoustic model to determine the word sequence $\hat{w_1^N}$ with the highest likelihood depending on the speech input and the model's parameters:

$$\hat{w_1^N} = \arg\max_{w_1^N} p(y_1^T|w_1^N)p(w_1^N)$$

This is equivalent to search a tree whose branches are labelled with the words of a dictionary $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$. Such a tree branches $|\mathcal{V}|$ times at each node – once per word in the dictionary.

Since dictionaries contain a large number of elements a naive approach is unfeasible. A technique called fast match [14], where a heuristic evaluation function generates a list of the best $N \ll |\mathcal{V}|$ candidates that are the only ones to be considered further, is applied to reduce the computational requirements to a bearable level.

We can use the Viterbi-algorithm with beam search or a tree search algorithm like A^* which searches through a tree in depth-first-manner but traversing paths with the best value according to a heuristic estimation function first. It has to be regarded that both techniques are suboptimal in some sense: Viterbi is not guaranteed to return the most probable word sequence and tree search methods may always prune off the best path as they have to use heuristics to be efficient enough [9].

5.3 Performance Measurement

To measure system performance an appropriate quanity has to be introduced. The word error rate which represents the ratio of word recognition errors to the overall number of words to be recognized is quite suitable for this purpose. Depending on the type of system the error ratio can vary by a few orders of magnitude [13].

Some factors affecting the word error ratio have to be considered:

- *Amount of Training Data:* a bigger amount of training data...
- *Background Noise:* a cleaner background...
- *Language Model Perplexity:* a lower language model branching factor...
- *Sampling Rate:* a higher sampling rate...
- *Speech Spontaneity:* a less spontaneous speech...
- *Vocabulary Size:* a smaller vocabulary...

...usually improves the system performance by lowering the word error ratio.

For instance an ASR-system, as presented in our paper, has been tested for spontaneous and speaker-independant telephone speech recognition tasks in [13] leading to a word error rate of 32.7%. The system can be used for different tasks without major changes as the underlying language's dictionary is huge and therefore the results seem quite significant for ASR-systems in general.

Furthermore there have been illustrated the recent advantages in the past few years on the field of ASR by first constructing the system without them – leading to a first word error rate of 40.5% – and then applying each one step by step to finally reach the above mentioned 32.7% for a relative improvement of 19%.

5.4 State-of-the-Art

Although ASR-systems have undergone major progress in the last years, humans still perform better at various tasks by at least an order of magnitude [13]: i.e. connected digits (0.72% vs. 0.009%), letters (5.0% vs. 1.6%), transactional speech (3.6% vs. 0.1%), dictation (7.2% vs. 0.9%), conversational telephone speech (43.0% vs. 5.0%).

It is clearly visible that human performance is yet unmatched, but we can also educe that ASR-systems are quite usable under certain circumstances. So what are the other limiting factors at the moment?

Primarily the natural intonation of the human voice is a major problem for ASR-systems as it increases the word error rate significantly. Another problem is the distinction of different voices: while training to specific speakers is helpful, it may not be applicable everywhere.

Besides these points, ASR-systems are still very vulnerable to background noise. While human recognition degrades by about 20% for a transition from a quiet background to 10dB, machines still deteriorate at least by a factor of 2.

Despite those unresolved problems, according to [13], human performance could be matched

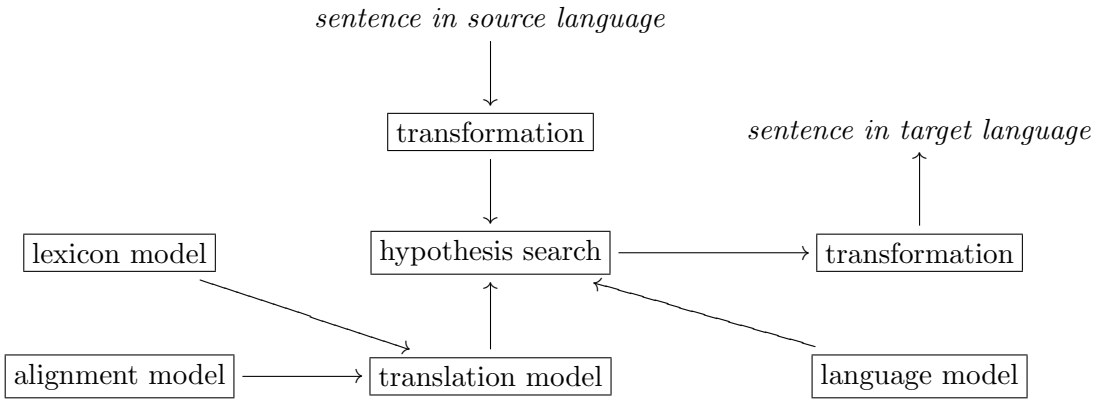


Figure 3: Speech Translation System – Block Diagram [11]

within ten years on condition that the research progress remains steady. This would allow the utilization in new domains where today's systems still prove too error-prone.

6 Speech Translation

The task of speech translation describes the process of mapping sentences out of a source language into a different target language. We will approach this similar as in the previous chapter but only give short outlines as both tasks are closely related.

This is the most general case of NLP as every other NLP task can be reduced to the translation between two languages.

6.1 Probabilistic Model

The goal of a translation system is to generate an approriate sentence $t_1^J = t_1 \dots t_j \dots t_J$ in the target language to a corresponding and given sentence in the source language $s_1^I = s_1 \dots s_i \dots s_I$. Here s_i describes a word in the source language at position i and t_j means a word in the target language at position j accordingly. Now the system tries to maximize the probability of the translated word sequence $\hat{t}_1^J$ given the original sentence s_1^I using Bayes' rule:

$$\hat{t}_1^J = \arg\max_{t_1^J} p(t_1^J|s_1^I) \equiv \arg\max_{t_1^J} p(t_1^J)p(s_1^I|t_1^J)$$

6.2 System Architecture

See Figure 3 for the block diagram illustrating the following elaborations.

Potentially needed preprocessing of input and postprocessing of output data is taken care of by the transformation component.

In the language model $p(t_1^J)$ gets evaluated for all possible sentences in the target language while considering its semantical and syntactical constraints.

Aligments $a_1^I = a_1 \dots a_i \dots a_I$ are introduced as hidden variables to consider the different word positions between the sentences in source or target language. That means the word s_i in source language gets mapped onto the word t_j in target language at position $j = a_i$.

The translation model $p(s_1^I|t_1^J)$ establishes a connection between source and target sentence by spanning the lexicon-model $p(s_1^I|a_1^I, t_1^J)$ and the alignment-model $p(a_1^I|t_1^J)$:

$$p(s_1^I|t_1^J) = \sum_{a_1^I} p(s_1^I, a_1^I|t_1^J) = \sum_{a_1^I} p(a_1^I|t_1^J) p(s_1^I|a_1^I, t_1^J)$$

To simplify further both models get decomposed in conditional distributions with special assumptions, i.e. $p(a_i|a_{i-1}, I)$ for the alignment and $p(s_1^I|t_{a_i})$ for the lexicon. [11]

6.3 Empirical Results

Two problem descriptions with corresponding systems and empirical tests are cited in [11]:

In the first one (VERBMOBIL) the task was to translate spoken natural language between German and English where the vocabulary consisted of approximately 8000 words. The best results have been achieved by a statistical system with an error rate of 29% whereas the rule-based approach yielded the worst result with an error rate of 62%. This does not mean that statistical systems work generally better but it is worth noting that in most empirical experiments they achieve the better results.

The other assignment was to translate between between spoken and/or written English and/or Chinese. Hereby the mapping between a small vocabulary in English and a large vocabulary in Chinese introduces another problem to be regarded. As the statistical approach again led to the best results it has been refuted that it is only applicable for similar languages.

Conclusion

Although the application areas of sequence learning are numerous, all systems function basically similar and rely on the same mathematical principles. HMMs are particulary suitable for the modelling of such problems as they allow to find the most probable series of state transitions given observed values and permit to evaluate the probability of occurence of a specific series of state transitions which covers a high ratio of sequence learning tasks. They are so adaptive because of the introduction of the hidden layer, i.e. they abstract the observations from the real series of actions we are interested in.

The inspected NIDS to detect multi-staged complex Internet attacks seems a very promising idea as it tries to handle the task at a higher level of abstraction than conventional

systems. Only such a global view allows to detect and classify the most sophisticated attacks that naturally are the most interesting ones as well. To allow more significant results a standardization of classification categories, i.e. types of attack, and a more widespread use to gather large amounts of examples would be helpful. Especially the problem of rare data, i.e. very complex attacks have no or too little corresponding training data, and the problem of partial data, i.e. find a way to identify multi-staged attacks as early as possible by determining likely preludes to complex intrusions, are still unresolved.

Progress has been made in the field of NLP in the last decades and such systems are applicable in specialized environment settings, i.e. agreement of appointments via conversational telephone speech, airline reservation systems or call centers to just name a few. Nevertheless they are not yet capable for universal usage as the very large dictionary and the ability to understand all types of speakers needed lead to an injustifiable performance. Further areas of interest include, amongst other things, the handling of natural intonation, the elimination of the need to train to a specific speaker while maintaining good performance on a large dictionary, and the elimination of unwanted background noise where humans still perform a lot better.

References

[1] Bengio, S. An Asynchronous Hidden Markov Model for Audio-Visual Speech Recongition. *Advances in Neural Information Processing Systems 15* (2003), 1237–1244.

[2] Brand, M., Oliver, N., and Pentland, A. Coupled Hidden Markov Models for Complex Action Recognition. *Proceedings of the Conference on Computer Vision and Pattern Recognition* (1997), 994ff.

[3] Cao, N. Hidden Markov Models with Bioinformatics Applications. `http://www.altum.com/bcig/events/tutorials/2004/2004_06.pdf`, June 2004.

[4] Cho, S.-B. Incorporating Soft Computing Techniques Into a Probabilistic Intrusion Detection System. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 32/2* (2002), 154–160.

[5] Choo, K. H., Tong, J. C., and Zhang, L. Recent Applications of Hidden Markov Models in Computational Biology. *Geno. Prot. Bioinfo. 2/2* (2004), 84–96.

[6] Eickeler, S., Kosmala, A., and Rigoll, G. Hidden Markov Model Based Continous Online Gesture Recognition. *14th International Conference on Pattern Recognition 2* (1998), 1206–1209.

[7] Fink, G. A. *Mustererkennung mit Markov-Modellen.* Teubner, 2003.

[8] Henderson, J., Salzberg, S., and Fasman, K. H. Finding Genes in DNA with a Hidden Markov Model. *Journal of Computational Biology 4/2* (1997), 127–141.

[9] Jelinek, F. *Statistical Methods for Speech Recognition.* MIT Press, 1998.

[10] Nair, V., and Clark, J. J. Automated Visual Surveillance Using Hidden Markov Models. *International Conference on Vision Interface* (2002), 88–93.

[11] Ney, H. Maschinelle Sprachverarbeitung. *Informatik Spektrum 2* (2003), 94–102.

[12] Ourston, D., Matzner, S., and Stump, W. Applications of Hidden Markov Models to Detecting Multi-stage Network Attacks. *Proceedings of the 36th Hawaii International Conference on System Sciences 5* (2003), 334.2ff.

[13] Padmanabhan, M., and Picheny, M. Large-Vocabulary Speech Recognition Algorithms. *IEEE Computer Journal 35/4* (2002), 42–50.

[14] Picone, J. Lecture 36: Stack Decoding. `http://www.cavs.msstate.edu/hse/ies/publications/courses/ece_8463/lectures/2004_fall/lecture_36/lecture_36.pdf`, 2004.

[15] Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77/2* (1989), 257–286.

[16] Starner, T. Visual Recognition of American Sign Language Using Hidden Markov Models. *Master's Thesis, MIT, Program in Media Arts* (1995).

[17] Wikipedia. Bioinformatics. http://en.wikipedia.org/wiki/Bioinformatics, January 2006.

[18] Wikipedia. Coarticulation. http://en.wikipedia.org/wiki/Coarticulation, January 2006.

[19] Wikipedia. Expectation-Maximization Algorithm. http://en.wikipedia.org/wiki/Expectation-maximization_algorithm, January 2006.

[20] Wikipedia. Hidden Markov Model. http://en.wikipedia.org/wiki/Hidden_Markov_Model, January 2006.

[21] Wikipedia. Natural Language Processing. http://en.wikipedia.org/wiki/Natural_language_processing, January 2006.

[22] Wikipedia. Network Intrusion Detection System. http://en.wikipedia.org/wiki/Network_intrusion_detection_system, February 2006.

[23] Wikipedia. Phoneme. http://en.wikipedia.org/wiki/Phoneme, January 2006.