IDIAP RESEARCH REPORT

Keyword Spotting on Word Lattices

De Greve Zacharie [a,b] and Joel Praveen Pinto [b]

IDIAP–RR 07-22

September 2007

[a] Faculté Polytechnique de Mons, Belgique
[b] IDIAP Research Institute, Martigny, Suisse

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 07-22

KEYWORD SPOTTING ON WORD LATTICES

De Greve Zacharie and Joel Praveen Pinto

SEPTEMBER 2007

Abstract

In spite of its numerous potential applications, Automatic Speech Recognition (ASR) remains a difficult (and mainly unsolved) problem. In addition to the intrinsic difficulty of the task, users tend to go beyond the pre-defined lexicon words, and the important keywords necessary to understand voice requests are often lost in extra words. In this context, it is often interesting to develop Keyword Spotting (KWS) approaches that will focus on the detection and recognition of pre-defined keywords lost in unconstrained, conversational, speech.

The goal of this project is to perform confidence based keyword spotting on word lattices, which are a compact way for storing the most probable hypotheses previously generated by a Large Vocabulary Continuous Speech Recognizer (LVCSR). Every spoken utterance is turned into a word lattice, and fast search and rescoring techniques are then applied to detect keywords in the lattice, using their confidence scores to take an accept/reject decision. By doing so, more knowledge (lexical and syntactic) can be taken into account in the first pass (LVCSR pass), while limiting the search space in the second pass, hence also allowing us to use more complex algorithms.

More specific to the work presented here, keyword hypotheses posteriors are estimated from the reduced search space (the lattices), and used as confidence scores to take the final decision. As posteriors minimize, by definition, the probability of error, we thus aim at minimizing the false alarm rate while maximizing true detection rates. Moreover, in order to take into account the word posterior probability mass dispersion among parallel lattice edges, various posterior rescoring techniques are investigated, all based on posterior accumulation over keyword hypotheses.

Generation and rescoring of the above word lattices were all based on the popular Hidden Markov Model (HMM) approaches. In this context, two particular instances of HMMs have been used and compared; the standard approach using Gaussian Mixture Models (GMMs) to estimate local (likelihood) scores, and a second one using a Multilayer Perceptron (MLP) directly estimating local posteriors.

All the experiments were performed on a 3 hours segment of Conversationnal Telephone Speech database (CTS) provided for international evaluations by the National Institute of Standard and Technology (NIST) within the 2006 Spoken Term Detection (STD) evaluation framework. We show here that our posterior based LVCSR keyword spotter yields more accurate results than classical single-pass, one-best approaches (such as the "Online Garbage Approach") by comparing Receiver Operating Characteristics (ROC) curves and Figure Of Merits (FOM).

Keywords : Automatic Speech Recognition, Keyword Spotting, word lattices, lattice rescoring, hybrid models

Acknowledgements

This work was funded by the AMI project, for Augmented Multiparty Interaction, a European project which aims to substantially advance state-of-the-art in the context of meeting multi-modal interaction, and more precisely by the AMIDA project, an extension of the foundation prepared in AMI with an emphasis on Distance Access, or meetings involving remote participants.

The authors also want to thank Brno University of Technology (Czesh Republic) and the Berkeley International Computer Science Institute (California, United States) for their technical support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic Speech Recognition, also named ASR, consists in extracting, by means of algorithms implemented in computers, the lexical information contained in a speech signal, converting it into a sequence of words, and to possibly interprate it. ASR offers a powerful way of communication between human and machines, and the financial community has understood it if we consider the huge amount of money invested in speech recognition research since the past 40 years.

Indeed, ASR covers a large area of applications. Telecommunication providers massively focus their efforts on the development of interfaces that allow access to data and various services over the telephone (like the AT&T's Maxwell personnal phone attendant). Others try to develop office related applications, such as PC/Workstations voice controlers and dictation systems. ASR can also yield useful help in manufacturing, control and business tasks : stock management, quality control, home automation,... Even the medical world has manifested some interest in the creation of systems which would automatically write medical reports or diagnostics from the surgeon's voice. Without forgetting the use of speech recognition techniques as a mean to improve the disabled persons quality of life, in video games, in all kind of information systems,...

However, even with this ambient enthusiasm, ASR remains a non-trivial problem far from being solved (understand ASR in real conditions, and not in laboratories). Many factors (such as speaker variability, environment effects altering speech signal...) drastically limit recognizer performance (as we will see in section 2.1 later). J.R. Pierce even spoke about "innocent enthusiasm" in his famous letter published in 1969 in the *Journal of Acoustical Society of America* ([35]) :

> *It would be too simple to say that work in speech recognition is carried out simply because one can get money for it. That is a necessary but not a sufficient condition. We are safe in asserting that speech recognition is attracting to money. ... People who work in that field are full of (in their own view) innocent enthusiasm...*

Indeed, as Pierce said, money is not sufficient to carry out the speech recognition task, especially in real conditions, *e.g.* where various non-wanted signals (car noise, radio music,. . .) could superpose to the desired speech waveform. A particular field of ASR adresses one of these "real-condition" problems, namely **Keyword Spotting** (often refered as KWS during this report).

Keyword spotting systems are designed to search words or group of words embedded in unconstrained speech. Typically, we understand the use of such kind of applications by considering the fact that, when a subject is prompted to speak by a prestored voice in real conditions, he will respond with useful information (**keywords**, to show that the rest of the speech should be ignored) embedded in extraneous speech (see figure 1.1 below). By the way, Wilpon *et.al.* have shown in 1990 ([51]) that when people were requested to speak one of five isolated words in a definite way over the telephone, only 65% followed the protocol.

Figure 1.1: Keyword Spotting problem ([28])

KWS systems proved themselves to be really useful in various contexts, such as command and control systems, information retrieval (*e.g.* indexing audio documents, meeting data,. . .) and automatic telephone services (*e.g.* catalog order entry, travel resevation, bank services,...). Another classical application could be the wiretap of telephonic lines.

Many KWS techniques are available nowadays, each of them adapted to various contexts. The On-Line Garbage Modeling technique (Bourlard *et.al*, [4] ; we will speak about it in section 3.1.2 on page

3.1.2) is for example well adapted in case of design of small vocal interfaces, where the system has only to deal with a few extraneous words around keywords. But such an approach is not sufficient when large databases, such as meeting or telephone conversation databases, need to be spotted for an information retrieval task.

Aim of the work

In this work, we intend to study the word spotting task on large databases, by performing posterior based keyword spotting on word lattices, which are a compact way for storing the most probable hypotheses generated by a Large Vocabulary Continuous Speech Recognizer (we will define word lattices on 2.4.4.2 on page 30). By doing so, more knowledge (lexical and syntactic) is taken into account than single-pass, one-best approaches. Moreover, as posteriors minimize, by definition, the probability of error, keyword hypotheses posteriors appear to be the best value to be used in order to take the final acceptance/rejection decision (see Bayes theory, section 2.4.2.1 on page 17).

We will also investigate various rescoring techniques. On one hand, keyword hypotheses posteriors will be accumulated following different criteria to take into account the fact that posterior probability mass is splitted among parallel lattice edges. On the other hand, we will analyse the impact of lattice acoustic likelihoods functional approximation by running the keyword spotter on lattices generated using a GMM-LVCSR recognizer, and on the same lattices rescored using a forced Viterbi alignment on hybrid HMM/ANN models.

Finally, we will show that word lattice based techniques are better adapted when searching for words in large databases, within an information retrieval framework for instance, than classical garbage based techniques by comparing our system with the well known On-Line Garbage Modeling approach.

Organisation of the Master Thesis

This work will be divided into three parts. The two firsts intend to offer to the reader unfamiliar with keyword spotting, and even with speech recognition, the tools indispensable to read the third chapter. These two chapters will each end with a complete conclusion/summary, to provide a fast and general picture of the problem. The third one will deal with the word lattice based word spotter we developed.

During the first part, we will expose basic but indispensable ASR concepts (chapter 2 page 5). We will show that the problem can be divided into two main tasks : the acoustic front-end (or features extraction module), which is a signal processing phase (section 2.3 on page 9), and the decoding in itself, which uses the features got from the first phase to really decode the utterance. For the second task (section 2.4 on page 12), we will see that several techniques exist nowadays : deterministic decoding by template matching, statistical decoding (thanks to Hidden Markov Models later reffered as HMMs), or the combination of HMMs and Artificial Neural networks, ANNs, that lead to the so-called hybrid theory. Statistical model training will also be adressed, and we will finish the chapter with more advanced concepts, some just for information, but others, such as word lattices definition, to understand the following work. All the steps will be tackled but not into deep details, keeping in mind that the goal of this chapter is not to demonstrate difficult mathematical concepts or algorithms (however indispensable to ASR), but is to present the bases of speech theory to help the reader to understand what will follow.

The second part will adress the KWS problem, as defined above, by dressing a brief state-of-the-art (chapter 3 on page 34). This chapter will present all the issues that need to be considered to develop a KWS strategy, and following them, will try to classify the currently available techniques and algorithms.

Finally, the third part will present the posterior-based LVCSR keyword spotter we developed. We will first begin with a global system description (4.2.1 on page 50), before exposing each step with more details (from 4.2.2 on page 51 to 4.2.4 on page 57). Posterior rescoring techniques will be adressed in section 4.2.4.2 on page 60. The impact of lattice link acoustic scores functional approximation will be analysed in section 4.3 on page 63. To that end, we will replace all the lattice acoustic scores by their estimation via HMM/ANN forced alignment. Generally, as likelihoods are not normalized values, scaling factors need to be introduced to optimize the recognizer performance. Such factors appear during word lattices generation, but also during keyword posterior estimations as we will see. This issue will de discussed in section 4.5 on page 67. The chapter will end with the system evaluation thanks to experiments on CTS database given within the 2006 NIST Spoken Term Detection evaluation framework (via ROC curves plots and FOM computations), and with its comparison with On-Line Garbage Modeling approach (4.6 on page 70).

Chapter 2

Automatic Speech Recognition (ASR) : an Overview

The automatic speech recognition problem consists in extracting, thanks to a computer, the lexical information contained in a speech signal, and to possibly interprate it. The first question we have to ask ourselves is : why speech recognition ?

ASR covers indeed a large area of applications :

- **telecommunications** : access to databases and different services over the telephone…
- **office/Desktop related applications** : PC voice controlers, dictation systems…
- **process control** : quality control, home automation…
- **business** : stock management…
- **medical or Legal world** : automatic creation of medical or legal reports, diagnostics…
- **miscellaneous** : aid to disabled persons, video games…
- …

Moreover, developing a system able to at least recognize prnounced word sequences (understand is another step, maybe impossible for someone, but at least more difficult…) appears to be indispensable as speech is the most prefered human way of communication, no one could argue with it. This last argument, added with ethical considerations (aid to handicapped), defend single-handedly the need to develop a speech science.

This chapter intends to dress a picture of speech theory by exposing the knowledge needed to understand the continuation of this report (and maybe a bit more…).

2.1 Challenges

To really dread the ASR task, it is important to understand its different complexity levels.

There is at first the *intra* (in the way words are pronounced) or *inter* (effects of dialects) *speaker variability* problem, which obviously increases the task difficulty. Moreover, is the system speaker dependent (optimized for a specific user) or speaker independent (which can recognize any user utterance) ? The first type of system is obviously easier to develop ; but it is possible to acquire this speaker independance thanks to specific training of acoustic models (see below for what an acoustic model is). Some applications, such as biomedical authentication, focus their effort not on the word sequences pronounced by the speaker, but on which speaker has uttered it. We have just pointed the **Speaker Identification** problem, for information only (the interested reader should refer to [6], section 5.17).

Is the system able to recognize isolated words or continuous speech ? We get also in this case better performances with the first aspect of the problem, where words are separated by silence periods. In continuous speech recognition, word boundaries are not known any more. The highest degree of complexity is reached when you have to recognize natural language (in contrast with read text for example), with its typical hesitations. We have moreover to deal with the word-form of the *coarticulation* phenomenon : words are not uttered the same way depending on the previous and next words in the sentence (french liaisons). More generally, we speak about coarticulation when phonemes (typical word sub-units) are pronounced differently following their context : place in the word, previous or next phoneme,... (see figure 2.1 below to understand its complexity). This problem increases drastically the task difficulty : we can even say that the most important ASR challenge consists in finding a way to overcome coarticulation.

Another important factor is the vocabulary size and its confusion degree. A smaller lexic is obviously simpler to recognize. Note that some small vocabularies are difficult to treat : for instance, for alphabet letters, the words are very short and similar from an acoustic point of view.

The ability of the system to work correctly in noisy environments is also really important. A lot of variables could affect the system performances, we distinguish for example :

- environment noise (due to recording environment, *e.g.* in a car), we speak about **additive** noise,
- additive noise correlated with speech signal (reverberation problem),
- using of different devices (microphons or filters with their own transfer function) resulting in **convolutive** noise,

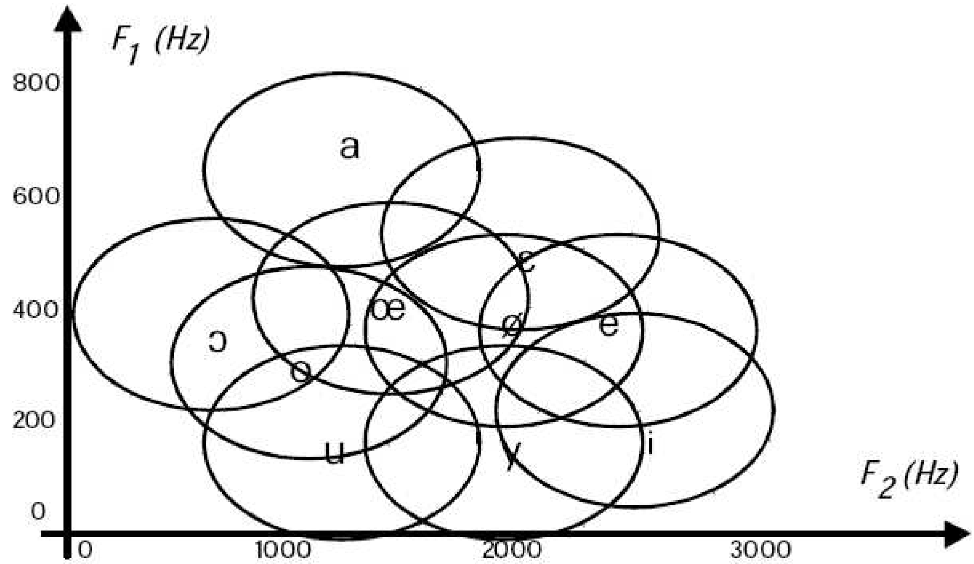

Figure 2.1: Coarticulation phenomenon ([13]) : the two axes represent the two first formant frequencies. We observe that phoneme areas overlap in the formant space, making ASR a non-trivial problem !

- effects of limited frequency bandpass (*e.g.* telephonic lines),
- effects due to the elocution context (nervosity, unusual speed during elocution, lips and breath noise, Lombard effect[1]. . .)

For information, figure 2.2 combines most of the constraints exposed above to show the different levels of complexity reached (from 1 to 10). Nowadays, we are more or less able to handle real tasks until a level of 5 or 6.

[1]Term which regroup all the modifications, often imperceptible, of the acoustic signal when pronounced in a noisy environment

		Isolated	*Connected*	*Continuous*
Speaker dependent	small	1	4	5
	large	4	5	6
Multi speaker	small	2	4	6
	large	4	5	7
Speaker independent	small	3	4	5
	large	5	8	10

LVCSR

Figure 2.2: ASR different levels of complexity ([13]), from 1 (easiest) to 10 (most difficult). Small and large refer to the vocabulary size. The most complex task appears to be the speaker independent Large Vocabulary Continuous Speech Recognition (LVCSR).

2.2 ASR flow-chart

Typically, an automatic speech recognition system is based on a generic structure as it can be seen below (see figure 2.3). We can divide the entire problem into two big parts we will discuss during next sections :

1. the **Acoustic front end** (see section 2.3 on page 9), or features extraction step, which consists in a signal processing phase,

2. the **Decoding** part, which performs the speech recognition in itself from the features and which is divided into several substeps we will discuss in section 2.4 on page 12.

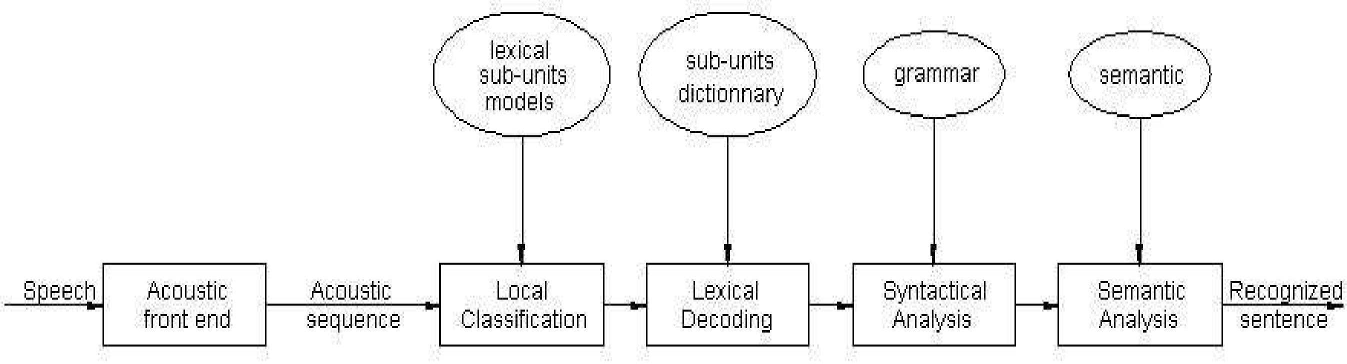

Figure 2.3: Basic structure in a ASR system

2.3 Features extraction for ASR

As said above, this step consists in a signal processing phase. We start from the speech signal to transform it in order to extract its source independent information (except the lexical content which has obviously to be present at the output of this module).

However, humans produce continuous signals, so the first step will be to **sample** the speech waveform thanks to an A/D converter. Typically, telephonic lines are the most employed in ASR, with a bandwith from 200 Hz to 3400 Hz, and speech is sampled at 8 kHz according to the Nyquist criterion (the minimum sampling frequency should be at least twice the maximum frequency carried by the signal to avoid aliasing). When the output comes from a microphone, sampling frequency F_e varies from 10 kHz to 16 kHz. But most of time, F_e will not exceed 10 kHz because it has been shown that only the four first formants[2] were really useful in speech recognition. If we consider having more or less one formant per kHz of bandpass, we easily understand the choice of this value.

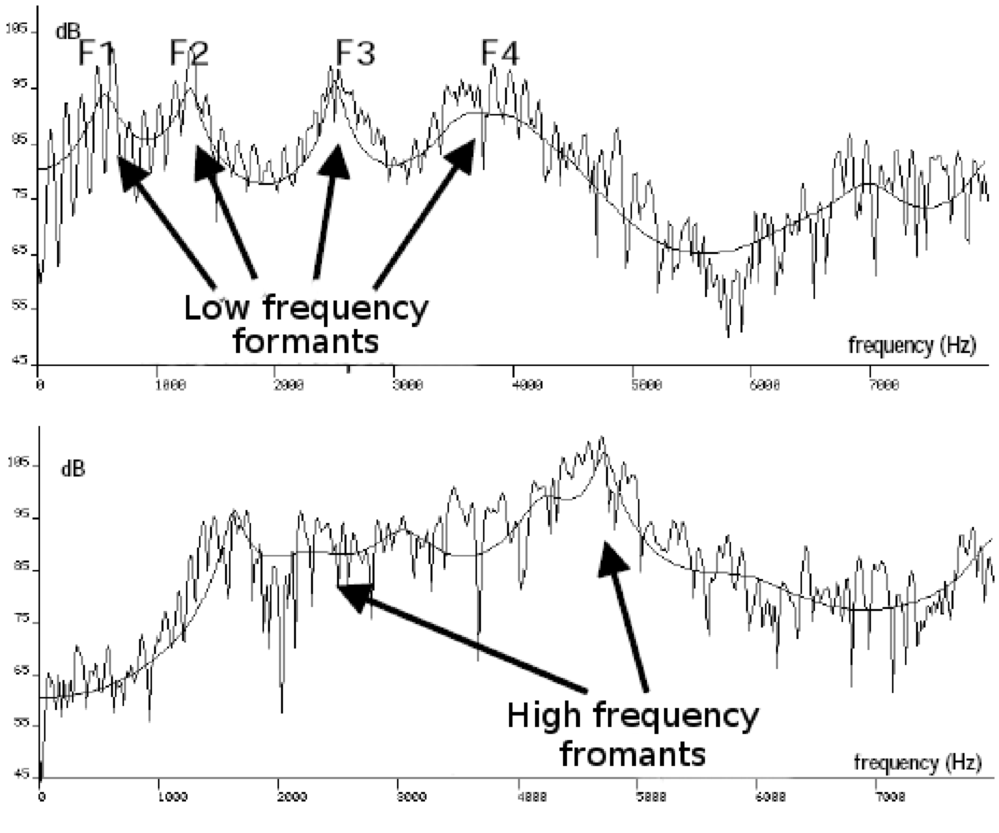

Figure 2.4: Spectral snaphot of speech ([13]). The upper graph corresponds to a ***voiced*** *sound, resulting from the glottal wave generated by vocal cords, thus revealing a fundamental frequency or* ***pitch****, and the lower to a* ***non-voiced*** *sound.*

[2]A formant consists in a vocal tract resonance, resulting in peaks on the speech signal spectral envelope (see figure 2.4 on page 9)

Once the signal is sampled, it is applied through a **pre-emphasis** filter, whose transfer function is :

$$H(z) = 1 - \mu z^{-1} \tag{2.1}$$

This filter plays the role of a high-pass filter, in order to accentuate the high frequency part of the signal spectrum. We know indeed that speech signal spectral intensity globally decreases when frequency increases (see figure 2.4 on page 9). More than simply equally sharing signal power along the entire frequency axis, pre-emphasis provides good algorithm conditioning for further processing. μ is usually taken as 0.95 (see chapter 2 of [6] for more information).

Another problem, which drastically increases the complexity of the speech recognition problem, needs to be pointed. Speech is indeed a non-stationary signal[3], which means that its statistic variables are not fixed in time. Consequentely, its analysis separates it in a succession of elementary sequences supposed to be stationary ([37]). All the theory that will follows in this report will be based on this very restrictive hypothesis. These small blocks are usually called **frames**. Typically, an analysis is applied every 10 *ms* on 30 *ms* frames, with sliding and superposition of the windows to increase the smoothness of the output signal (see figure 2.6 on page 12).

Most of speech recognition features are based on a frame-level spectral analysis, obtained either from Linear Predictive Coding (LPC, see chapter 2 of [6]) or from a Discrete Fourier Transform (DFT). For the latter, DFT coefficients $X[k]$ of the frame $x[n] = \{x[0], \ldots, x[N-1]\}$ can be obtained by :

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp^{-jnk\frac{2\pi}{N}} \qquad (k = 0, \ldots, N-1) \tag{2.2}$$

Various features extraction techniques exist, some of them taking into account human perception system characteristics, in order to generate what we call an **acoustic vector**. So this step converts a word or a sentence into a sequence of acoustic vectors $X = \{x_1, \ldots, x_N\}$, where x_i is an acoustic vector. We distinguish for example :

- the **Mel Frequency Cepstral Coefficients** (MFCC), resulting from a cepstral analysis (Fourier transform of the log-spectra) in a mel-scaled critical band spectrum (see chapter 3 of [6]),
- the **Perceptual Linear Prediction** (PLP) parameters calculated from a spectra representative of the signal frequency content following Bark scaling (taking into account the human auditive

[3]It has to be non-stationnary, otherwise the information carried by the speech signal would be equal to zero !

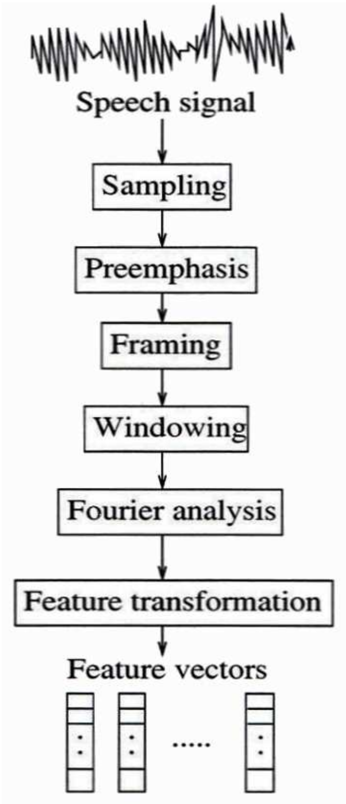

Figure 2.5: Features extraction process in ASR ([26])

system perceptual properties with the critical bands). We will discuss PLP in more details in chapter 4 as they will be the system used features,

- the RASTA-PLP features (PLP modification to achieve robustness to convoutional noises, chapter 22 of [19]),
- the multi RASTA-PLP features ([23]),
- ...

We could also note the use of **dynamic features** in addition to the mentionned features : it consists in extending the acoustic vectors with their first and second temporal derivative estimations, taking thus into account speech dynamic behavior. We speak in this case about **delta features** and **delta-delta features** respectively, where D is the time window length over which the delta is computed :

$$\Delta x_t = \frac{\sum_{d=1}^{D} d(x_{t+d} - x_{t-d})}{2\sum_{d=1}^{D} d^2} \tag{2.3}$$

As we said previously and to summarize, the goal of this first phase of features extraction is to reduct the information intially present into the speech signal and to transform it into a sequence of acoustic vectors robust to acoustic variations, while keeping faithful to the lexical content.

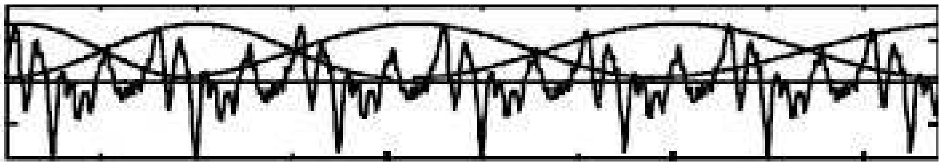

Figure 2.6: Framing and windowing ([13])

2.4 Decoding in ASR

As can be seen on figure 2.3 on page 8, the recognition task in itself is subdivised into several steps we will briefly describe during this section, before making the link between this general approach and existing recognition techniques (deterministic template matching, HMMs, HMM/ANN hybrid models).

The recognition step begin with a **local classification** which consists in estimating for each instant n (thus for each frame) a distance or a local probability which represents the *score* for the frame verifying a local hypothesis. In other words, a **lexical sub-unit** (*e.g.* the **phoneme** or another unit, but we will only consider phonemes in this report[4]) is associated with each acoustic vector, based on a specific criterion. Then, all these scores are kept and integrated in time such as it will be disussed below, by Dynamic Time Warping for instance (section 2.4.1 on page 13) or using the Hidden Markov Model formalism (section 2.4.2 on page 17). So this first phase consists in a classification problem, based on models of the sub-units.

Typically, the matter is not to recognize lexical sub-units but words, which will be defined as a concatenation of the sub-units, or sentences, which are themselves concatenation of words. Thus higher level restrictions have to be taken in consideration. Lexical considerations for instance define each word in terms of sub-units (*e.g.* phonetic transcription) using a dictionnary (**lexical decoding**).

On the other hand, syntactical constraints (**syntactical analysis**) have the role to integrate grammar considerations by introducing valid (or not) words sequences, in the process of making sen-

[4]Phonemes form a set of **unique** sounds that a language uses. For instance, two sounds are associated with two different phonemes if they make a distinction between two words. For information, people have also tried to use articulatory features as lexical sub-units in speech recognition, others syllables,. . .

tences.

For information, more complex grammars have also been developed to include semantic significations in the recognition phase (**semantic analysis**), but this task seems to be very difficult because of the spoken langage complex structure.

It is important to note that figure 2.3 does not represent exactly the reality. Indeed, all the steps are not performed in a sequential way as it has been schematized, but are applied simultaneously during decoding. For example, the last two steps introduce during the decoding procedure some probabilities that penalize (or enhance) transitions between different words. In fact, these penalties are (real time when it is possible, or not in case of very complex grammars) integrated into the decoding algorithm in order to calculate global scores.

To summarize, local scores associated to sub-units are integrated in time to generate global scores corresponding to possible phoneme sequences. For each time frame n, a new word or sentence hypothesis is generated (with respect to the constraints cited above), evaluated and possibly chosen. As we will see after, this temporal integration can be completed by different ways (DTW or HMMs).

2.4.1 Deterministic approach

All the considerations evocated above lead us to think about the integration of the local scores into time domain. Indeed, as we said previously, the goal of the decoding phase is not to recognize sub-units but words or group of words. Moreover, time variations can occur with slower or faster pronunciation. Thus we can say that speech recognition involves both "static" pattern classification and sequence recognition. In this context, the recognition problem consists in associating a sequence $Q = \{q_1, \ldots, q_l, \ldots, q_L\}$ to the sequence to decode $X = \{x_1, \ldots, x_n, \ldots, x_N\}$, with L different from N (time deformation) and where Q is determined in order to minimize an error criterion.

In this section, we will suppose that the sequence Q corresponds to a sequence of acoustic vectors (like X), and plays the role of a **reference** sequence (or **template**) we will note $Y = \{y_1, \ldots, y_j, \ldots, y_J\}$. The problem of template matching approach will be to compare an input sequence X to **all** the references $Y^k (1 \leq k \leq K)$, where K represents the number of reference sequences.

We also obviously need to give correct values to the reference templates Y^k. We have just pointed the so-called problem of *training* of the reference models, which is very important because well-defined models means high performances. Large training databases are indispensable to get robust

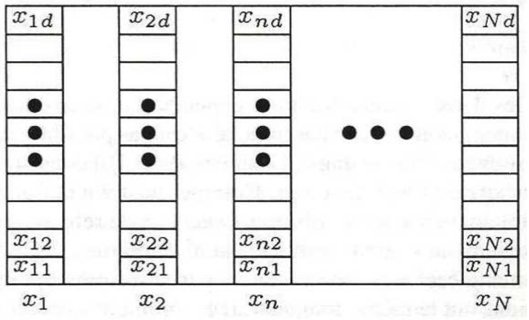

Figure 2.7: Example of template ([6]) used as reference represented by the sequence $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ *of* N *(one each* 10 *ms) acoustic vectors* $x_n = \{x_{n1}, \ldots, x_{nd}\}$

models by taking into account a lot of different elocution contexts, different speaker characteristics (in case of speaker independent recognition),... In this kind of deterministic approach (template matching), training consists only in calculating and stocking references $Y^k = \{y_1^k, \ldots, y_j^k, \ldots, y_{J(k)}^k\}$, $k = 1, \ldots, K$. But other procedures are applied in case of statistical methods (*e.g.* in case of decoding using HMMs, as we will see in section 2.4.2.3, page 23).

But now, how could we compare the sequences ? Following what criterion ? It is indispensable to define a *distance* which will be used as the scores discussed above. Then, the recognizing problem consists in identifying a test sequence $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ to one of the dictionnary words by searching a reference template which **global distance** with X is minimized. Two **local** distances among the most employed are defined as follows (see chapter 5 of [6]) :

- the *Euclidian distance* :

$$d(x_n, y_j^k) = \|x_n - y_j\|^2 = \sqrt{\sum_{i=1}^{d} (x_{ni} - y_j^k)^2}, \tag{2.4}$$

- the *Mahalanobis distance* (gaussian classifier)

$$d(x_n, y_j^k) = (x_n - y_j^k)^T \Sigma^{-1} (x_n - y_j^k). \tag{2.5}$$

We desire to get a method which allows us to take into account time extensions or deformations. A particular form of dynamic programming called *Dynamic Time Warping* (DTW) can be applied. In dynamic programming, we need to be able to formulate the problem in terms of optimal sub-policies, so that the main idea can be formulated as follows : the optimal policy would be made of optimal sub-policies.

In this case, each reference Y^k is associated with a N by $J(k)$ matrix D (where N and $J(k)$ are respectively the number of acoustic vectors in the test sequence and the length of the k^{th} reference). A local distance $d(x_n, y_j^k)$ as defined above is then associated with each element (n, j) of the matrix. At this point, the matter is to search the path through D in order to minimize the sum of local distances to go from an initial point $(1, 1)$ to a final point $(N, J(k))$, corresponding respectively to the beginning and the end of the two sequences. All these considerations lead us to implement the following algorithm :

$$D(n,j,k) = d(n,j,k) + \min_{p(n,j,k)} \{D(p(n,j,k) + t[p(n,j),(n,j)])\} \quad (2.6)$$

where :

- $d(n, j, k)$ replaces $d(x_n, y_j^k)$ as the local distance notation,
- $D(n, j, k)$ represents an **accumulated** distance corresponding to the optimal distance obtained by comparing the n first test vectors with the j first reference vectors of the k^{th} reference,
- $p(n, j, k)$ are all the possible (n, j, k) predecessors in order to get an acceptable trajectory. The most common constraints can be observed on figure 2.8 below,
- $t[p(n, j),(n, j)]$ consists in possible transition penalties, which would differently ponder the various pathes through the matrix (see [38] for more information).

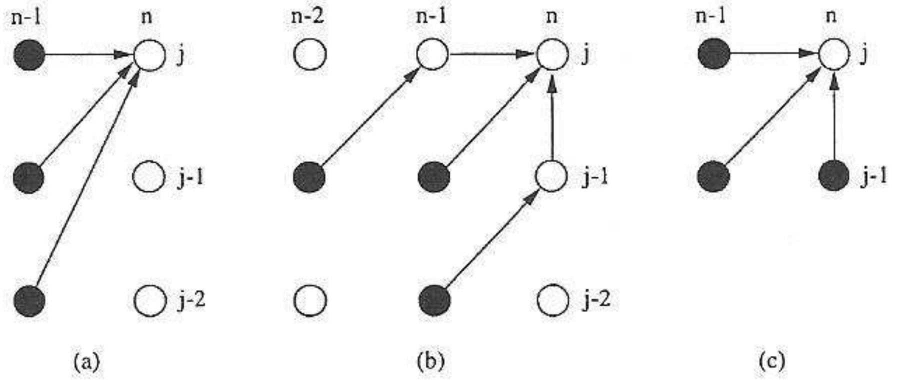

Figure 2.8: Typical ASR local constraints ([6]) ; n is the temporal index of the current input window while j is the index of the reference window.

The implementation of the recursion exposed above leads to the optimal non linear time deformation, as it can be seen on figure 2.9 for isolated word recognition (where words are separated with additive silence). Figure 2.10 below shows an example of DTW path in case of Continuous Speech Recognition. In this case, the sentence pronounced would be $k' - K - 1 - K - k''$. Obviously, a **backtracking** phase is necessary after recursion to find the word sequence associated to this optimal path. Note that this method allows us to **automatically segment the sentence** in terms of reference segments (words in our case).

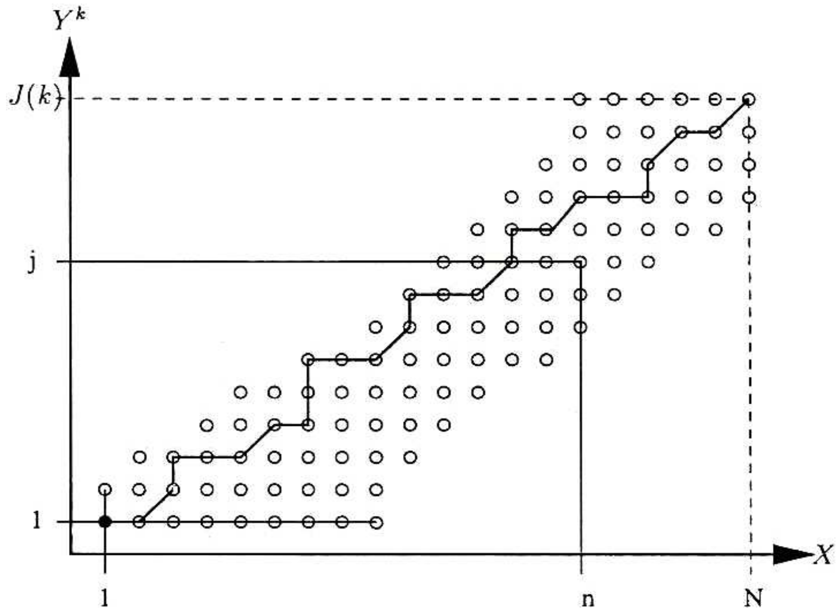

Figure 2.9: Dynamic Time Warping for isolated word recognition ([6])

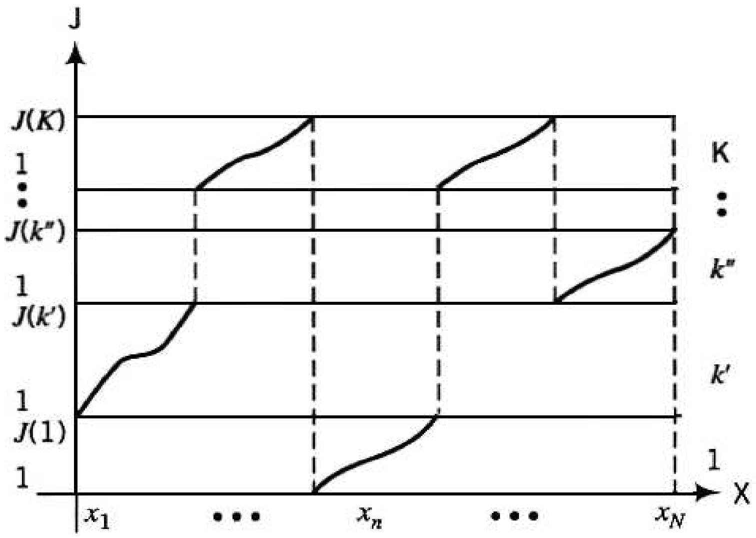

Figure 2.10: Sample of DTW path for connected word recognition ([6]). Local constraints are adapted to handle the case where the current frame could come from other words.

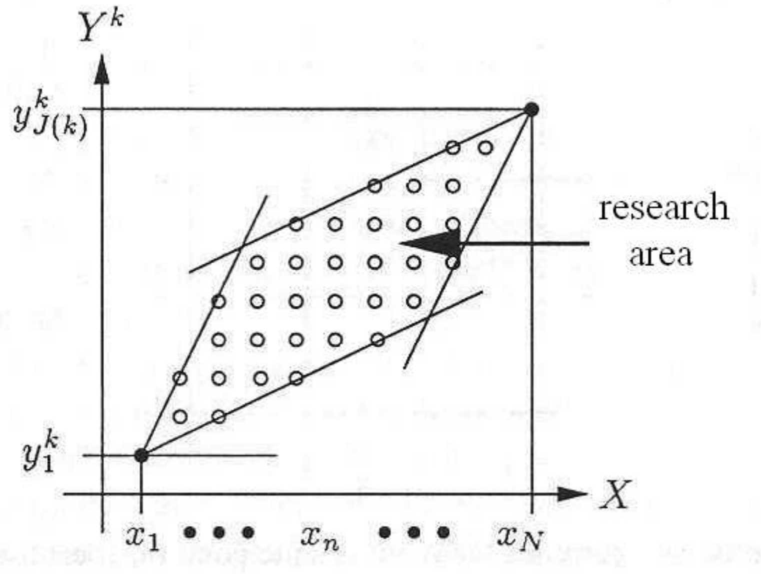

Figure 2.11: Global constraints for DTW ([6]) : reduces the research area around the matrix main diagonal.

2.4.2 Statistical approach

Deterministic speech recognition has been used with success until the mid 1980's, when we saw the arrival of statistical models, more precisely Hidden Markov Models (HMMs), in speech theory. It is indeed easy to understand that statistical models will better handle the speech variability than simply storing reference templates[5]. Moreover, HMMs, as we will see below, provide a way to train their own transition penalties (namely transition probabilities), in opposition to template matching where these values have to be tuned empirically. This section intends to introduce the formalism of statistical speech recognition, and will focus on the use of Hidden Markov Models.

2.4.2.1 Basic concepts

In statistical speech recognition, each linguistic unit is associated with a statistical model, and each model is parametrized by distributions. The recognition approach is based on the Bayes rule :

$$P(M_j|X) = \frac{P(X|M_j)P(M_j)}{P(X)} \tag{2.7}$$

where :

- M_j is the j^{th} model and $j = 1, \ldots, J$

[5]Note that nowadays, template matching approach tends to be used again, as more memory and computational power are available to store and process loads of templates.

- X stands for the data to recognize (sequence of acoustic vectors),
- $P(M_j|X)$ is the probability of being in model M_j **given** the data, or **a posteriori** probability,
- $P(X|M_j)$ is the probability with which the model could generate the data, or **likelihood**,
- $P(M_j)$ is the probability of having the model M_j, or **prior** probability (cause it could be estimated before seeing the data X),
- $P(X)$ is the probability to have the data X.

Note that M_j should stand for a sentence model. As it is not possible to define (and train, see below) a model for each existing sentence (we would have an infinite number of models !), each sentence is decomposed in terms of sub-models associated to smaller linguistic units. For instance, a sentence will be divided into word models, words will be composed of phoneme models, and phonemes will be modeled by tri-state HMMs, leading to a big HMM network.

The optimal classifier (or Bayesian classifier) minimizes the probability of error. According to Bayes decision rule, it maximizes the a posteriori probability ([20]). In other words, we can say that the data sequence $X \in M_{j_{opt}}$ if j_{opt} is evaluated as follows :

$$\begin{aligned} j_{opt} &= \underset{j}{\operatorname{argmax}} P(M_j|X) \\ &= \underset{j}{\operatorname{argmax}} \frac{P(X|M_j)P(M_J)}{P(X)} \\ &= \underset{j}{\operatorname{argmax}} P(X|M_j)P(M_J) \end{aligned} \tag{2.8}$$

The last line of equation *2.8* is written considering the fact that $P(X)$ is a constant independent from the considered model (so that we can remove it from the decision process). Thus, the problem will be to estimate $P(M_j|X)$ for each model M_j in order to perform recognition. We will need to estimate $P(X|M_j)$, which will be done using **acoustic models** (*e.g.* HMMs, section 2.4.2.2 on page 19), and $P(M_j)$, leading to the definition of **language models** (section 2.4.2.4 on page 24).

To be completely right, these probabilities depend on a parameter set Θ estimated via a **training** procedure (section 2.4.2.3 on page 23) which goal is to maximize a posteriori probability. If M_j is the model associated with the training sequence X_j[6], we have :

[6]We talk about **supervised** training : we know the class, or model, each training sample belongs to.**Unsupervised** training also exists, but we will not analyze it during this work (see chapter 5 of [6] for more information).

$$\Theta^* = \underset{\Theta}{\text{argmax}} \prod_{j=1}^{J} P(M_j|X_j,\Theta) \tag{2.9}$$

Typically, parameters which describe acoustic probabilities are supposed to be independent from those describing language probabilities, so that Θ is split into the two sets Θ_A and Θ_L, respectively for acoustic parameters and language parameters. As their joint evaluation is not feasible for practical reasons, the two probabilities are estimated from two different training sets, respectively ε_A and ε_L, and equation *2.9* becomes :

$$\Theta_A^* = \underset{\Theta_A}{\text{argmax}} \prod_{j=1, X\in\varepsilon_A}^{J} P(X_j|M_j,\Theta_A) \tag{2.10}$$

$$\Theta_L^* = \underset{\Theta_L}{\text{argmax}} \prod_{j=1, X\in\varepsilon_L}^{J} P(M_j|\Theta_L) \tag{2.11}$$

Thus, if X represents test data again, Equation *2.8* can be reformulated as :

$$X \in M_{j_{opt}} \Leftrightarrow j_{opt} = \underset{j}{\text{argmax}}\, P(X|M_j,\Theta_A)P(M_J|\Theta_L) \tag{2.12}$$

All the following will be based on this formalism[7].

2.4.2.2 Hidden Markov Models as acoustic models

A *Hidden Markov Model (HMM)* is a stochastic finite state automaton where each state corresponds to a probability density function (whereas for simple Markov models, each state is associated with a deterministic event). In HMMs, states are also linked by transition probabilities. A Hidden Markov Model thus corresponds to a **doubly stochastic process**, to explain the fact that statistics are present at two levels : for the transition between states via the transition probabilities, and at the state level via the probability density functions. HMMs generate stochastic sequences where the corresponding emitting states are not directly observed : given an observation sequence, it is impossible to find a priori the associated state sequence because a state can generate different observations following its own density function. That is why we said *hidden*.

[7]Note that $P(X|M_j) = \sum_{i=1}^{I} P(X|P_i)P(P_i|M_j)$, where P_i allows I different pronunciations for a same word M_j for example. $P(P_i|M_j)$ introduces what we call a **phonetic**, or pronunciation, model.

In speech recognition, acoustic vector sequences are supposed to be produced by this stochastic machine built from a set of stationnary states $\{q_1, \ldots, q_l, \ldots, q_S\}$, where S is the number of HMMs states. The probability density functions, or **emission probabilities**, govern the generation of feature vectors. Transition between states are supposed to be instantaneous.

When **using HMMs for the acoustic model problem**, some hypotheses are usually emitted to make the task computationally acceptable :

1. We use **first order** Markov models, which means that the probability to be in one state only depends on the previous state. Given that $P(q_l^n)$ means "the probability of being in state q_l at time n", and that X_1^{n-1} represents the data from time 1 to time $n-1$ (sequence of acoustic vectors), we have :

$$P(q_l^n|q_k^{n-1}, X) = P(q_l^n|q_k^{n-1}) \tag{2.13}$$

2. We assume the **independency between observations**, which means that acoustic observations don't depend on the past observations and on the past state sequence. The first assumption is equivalent to affirm that acoustic vectors are not correlated in time, which is obviously erroneous[8], while the second says that phoneme pronunciation doesn't depend on the previous phonemes :

$$P(x_n|q_l^n, q_k^{n-1}, X_1^{n-1}) = p(x_n|q_l^n) \tag{2.14}$$

Given these restrictions, a HMM can be defined from its state topology and its parameters set (figure 2.12 summarizes that) :

- $p(x_n|q_l^n, M) = p(x_n|q_l^n)$ are called **state emission probabilities**, where x_n is an acoustic vector part from the sequence to recognize. They are represented by probability density functions, for instance gaussians or weighted sum of single gaussians (Gaussian Mixture Models, or GMM).
- $P(q_l^n|q_k^{n-1})$ are the **transition probabilities**.

Now that we have a good formalism for HMMs, how could we estimate the acoustic likelihood $P(X|M)$? Remembering the hypotheses cited above and that $n = 0, \ldots, N$ refers to time frames, and assuming that Q contains the set of all state paths through the model M, we can write :

[8]This assumptions can be relaxed with the use of autoregressive HMMs (chapter 5 of[6]).

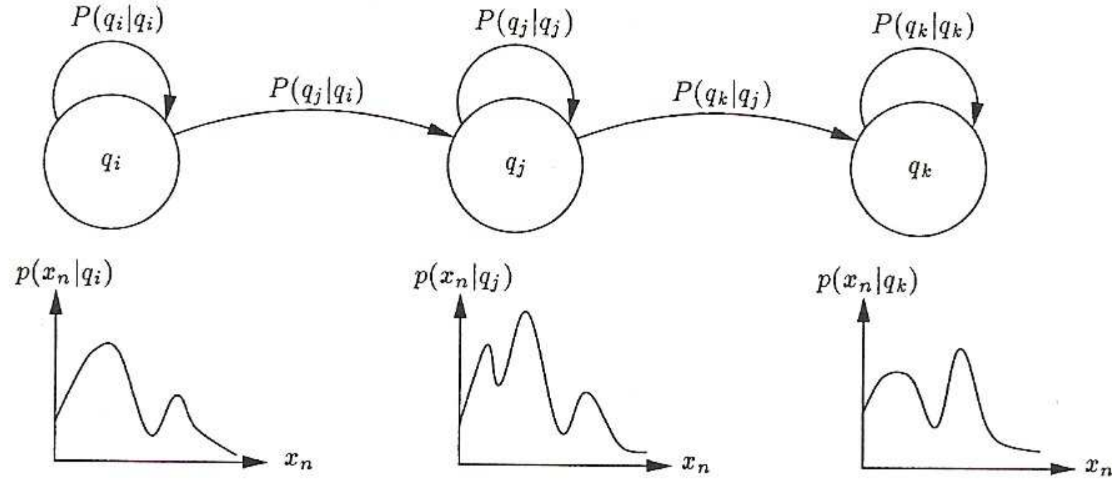

Figure 2.12: Three states hidden Markov model. Each state is defined by a multidimensionnal pdf, which is multigaussian in this case (a pondered sum of single gaussians). This model is one from Bakis models or left-right models which are particularly adapted to words representation (they are indeed considered as succession of stationnary states). Note that it could also contain state jumps to represent possible alternative pronunciation (more info are available on [6],[13]and [20])

$$\begin{aligned}
P(X|M) &= \sum_{Q\in M} P(X,Q|M) \\
&= \sum_{Q\in M} P(X|Q,M)P(Q|M) \\
&= \sum_{Q\in M} P(q_1^0)p(x_0|q_1^0)\prod_{n=1}^{N} p(x_n|q_l^n)\prod_{n=1}^{N} P(q_l^n|q_k^{n-1}) \\
&= \sum_{Q\in M} P(q_1^0)p(x_0|q_1^0)\prod_{n=1}^{N} p(x_n|q_l^n)P(q_l^n|q_k^{n-1}) \qquad (2.15)
\end{aligned}$$

q_1^0 is the first state of the considered path, followed at time $t = 0$, and $P(q_1^0)$ stands for its initial probability. The above equation gives an exact way to compute the likelihood, but requires a high operation amount ($S^N.2N$ operations, S^N possible state sequences with more or less $2N$ computations on each, where S is the number of states, see [6]). Thus, a particular algorithm named **forward backward** algorithm is employed to reduce computational needs (see p.239 of [6] or p.67 of [20] to consult the algorithm presentation and demonstration).

In practise, the so-called **Viterbi approximation** is prefered. In this case, we consider only the likelihood of the best state sequence (whereas for the forward-backward algorithm, all the state sequences are taken into account), and the sum in equation 2.15 is turned into a maximization :

$$P^*(X|M) = \max_{Q \in M} p(q_1^0) p(x_0|q_0) \prod_{n=1}^{N} p(x_n|q_l^n) P(q_l^n|q_k^{n-1}) \tag{2.16}$$

Equation 2.16 can be computed with reasonable computational cost using a simplification of the forward-backward recursion cited above. Thus we have the following**Viterbi recursion**, remembering that S is the number of states in the HMM (see [6] and [20] for a demonstration) :

$$\text{Initialization :} \quad p^*(q_k^0, x_0) = P(q_k^0) p(x_0|q_k^0) \quad \text{for each allowed starting state } q_k \tag{2.17}$$

$$\text{Recursion :} \quad p^*(q_l^n, X_1^n|M) = \max_k \left[p^*(q_k^{n-1}, X_1^{n-1}|M) P(q_l|q_k) \right] p(x_n|q_l) \tag{2.18}$$

with $n = 1, \ldots, N$; for each allowed l

$$\text{Termination :} \quad P^*(X|M) = \max_k p^*(q_k^N, X_1^N|M) \tag{2.19}$$

Moreover, the **Viterbi approximation also gives the optimal state sequence** if we store the best states during the recursion. If the states corresponds to acoustic sub-units, Viterbi approach segments the test data into linguistic units, thus (partially) resolving the recognition problem. Equation 2.18 is often written in the logarithmic domain to avoid underflow during computation. Indeed, in probability domain, we have to multiply a large amount of values nearby zero :

$$-\log p^*(q_l^n, X_1^n|M) = \min_k \left[-\log p^*(q_k^{n-1}, X_1^{n-1}|M) - \log P(q_l|q_k) \right] - \log p(x_n|q_l) \tag{2.20}$$

To finish, we can compare statistical decoding and DTW (exposed on section 2.4.1), in sense that a huge matrix stocking local scores is also created and processed in order to determine the best path, and so the sequence of acoustic units. But in this case, emission probabilities of each state play the role of the local distances, and HMMs transition probabilities the role of transition penalties.

To conclude the recognition process, we need now to incorporate the language model probability $P(M)$ following Equation 2.8. This issue will be discussed in section 2.4.2.4

2.4.2.3 Training of the statistical models

Section 2.4.2.2 below provided a way to estimate acoustic likelihoods $P(X|M_j)$, using HMMs and their parameters (transition probabilities ; mean vectors and covariance matrixes for each state distribution). We now need to find a way to estimate them, refered as Θ_A, from audio training samples : this is the **training** phase. Suppose we have J training sequences $X_1, \ldots, X_j, \ldots, X_J$. As shown in section 2.4.2.1, the idea is to maximize the likelihood of the training samples given the correct associated models :

$$\Theta_A^* = \underset{\Theta_A}{\text{argmax}} \prod_{j=1}^{J} P(X_j|M_j, \Theta_A) \tag{2.21}$$

In the above equation, $P(X_j|M_j, \Theta_A)$ could stand either for a "global" likelihood, in sense that all state paths are taken into account during its estimation, or an approximated likelihood in the Viterbi sense (just as in section 2.4.2.2). This problem cannot be solved analytically and needs iterative procedures. Most current solutions are based on **Expectation-Maximization** (EM) algorithm we will not describe here (please refer to p.191 of [6] for a detailed description) : we speak about **Baum-Welch** training if we consider all the state paths, or about **Viterbi** training if we only take the best path into account for likelihood computations.

The two methods are based on the same scheme. We first begin by defining an initial estimation of the parameters based on some previous knowledge. We then compute at each time n the likelihood of training samples given the appropriate model (Equation 2.21), and use the results to make a reestimation of the parameters. We stop when total likelihood reach a (non-local) maximum. This kind of training is a **Maximum Likelihood** (ML) training (we will cite other criteria in section 2.4.4.1).

As said above, Baum-Welch training uses all the possible state pathes to evaluate the global likelihood. We can show that this kind of training is based on the forward-backward recursion cited previously in section 2.4.2.2 : Baum-Welch training is often named **forward-backward** training.

Viterbi approximation considerably simplifies the training procedure. We begin with a first data segmentation in terms of HMMs states. By simple counting, we can emit a first evaluation of the HMM parameters. For instance, transition probabilities can be estimated by dividing the number of times we observed a transition between two states by the number of times the first state has been visited). A Viterbi alignment (like in Equation 2.20, keeping the best state sequence in memory) is then performed, leading to a new segmentation of the training data. We can now reestimate the parameter set and so on. The entire process is illustrated on Figure 2.13 below.

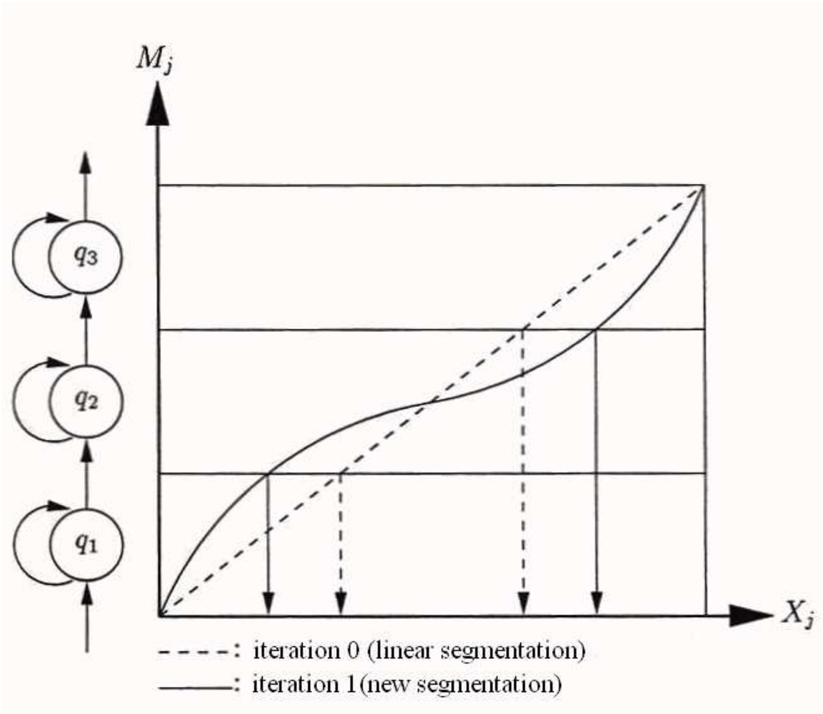

Figure 2.13: Illustration of Viterbi training ([6])

Typically, Gaussians, or multigaussians (thus modifying the training procedure) are used as emission probabilities estimators. But discrete distributions could be used too, thus needing to associate each training sequence with an index in a code book by vector quantization.

2.4.2.4 On the use of language models

We are now able to estimate $P(X|M)$ and acoustic model M parameters. Equation 2.8 also contained another term we have not spoken about yet : the prior probabilities $P(M)$. Such probabilities are estimated by what we call a **language model**, defined by a parameter set Θ_L which is supposed to be independent from Θ_A. Usually, M models sentences, thus $P(M)$ cannot be computed directly and is evaluated by dividing M in shorter sequences (such as word sequences as we will see below).

The three basic problems in language modeling remain :

1. How could we estimate the model parameter set, Θ_L ?
2. How could we estimate $P(M|\Theta_L)$?
3. How could we incorporate these values into the decoding process presented during section 2.4.2.2 ?

We will explain and develop these three basic questions for the currently most used language models : N**-gram statistics**. We speak about N-grams when the probability of having a word can be obtained from the $N-1$ previous words. Let assume that a sentence M can be decomposed in terms of words :

$$M = (W_1, W_2, \ldots, W_k, \ldots, W_K)$$

We can always write :

$$\begin{aligned} P(M) &= P(W_1, W_2, \ldots, W_k, \ldots, W_K) \\ &= \prod_{k=1}^{K} P(W_k | W_{k-1}, W_{k-2}, \ldots, W_0) \end{aligned} \tag{2.22}$$

In case of N-gram models. Equation 2.22 can be reformulated as ($P^*(M)$ means that we approximate $P(M)$) :

$$P^*(M) = \prod_{k=1}^{K} P(W_k | W_{k-1}, W_{k-2}, \ldots, W_{k-N}) \tag{2.23}$$

For bigrams, N is equal to 1 ,and is equal to 2 for trigrams. Equation 2.23 above has just answered to question 2 : we can estimate $P(M)$ from N-gram probabilities $P(W_k|W_{k-1}, W_{k-2}, \ldots, W_{k-N})$ which play the role of the parameters. These parameters can easily be evaluated by simple counting on very large written text databases, thus resolving question 1 (more elaborated techniques obviously exist to handle particular problems : smoothing procedures, backoff models,... - see chapter 6 of [29] for more details).

But now, how incorporate these values into the recognition procedure ? We could imagine to use the language model probabilities as transition probabilities between HMM word models (namely the probability of exit from a word-final state to the first state of another word) during forward-backward or Viterbi decoding. Obviously, such an approach could be only applied to bigram probabilities (following also the assumption made on section 2.4.2.2 on page 19 that we use first order HMMs), which mean by definition the probability of having a word given the previous. Longer-span language models would not respect the Markovian assumption in this case ; trigrams, for instance, refer to the two previous words. But other techniques allow the use of higher order language models :

- Use higher order Markov models. Such models can always be divided into first order Markov models, but the total number of states would explode, leading to computational problems.

- Use of a **stack decoding** technique, based on A^* search (see chapter 7 of [29] for more information) where allowable word sequences are stored in a tree. This technique possesses the advantage that the language model is **decoupled** from the acoustic model and is not used to generate hypotheses (not as the other approaches, where probability transitions between HMM word models have to be included during the Viterbi search !).

- Multiple-pass decoding (see [43]) : the most probable hypotheses are generated from a decoding first pass using a simple bigram language model and are stored in **N-best lists** or **word lattices**. A eventual further pass applies higher order language models on these hypotheses. We will analyse the word lattices into more details on chapter 4.

2.4.2.5 Hybrid HMM/ANN models for ASR

It is also possible to combine Artificial Neural Networks (ANNs) with HMMs in speech recognition, as proposed in [9]). Before exposing this method we will briefly define ANNs. ANNs try to roughly copy the function of brain neurons and synapses. Typically, a neural network consists in connecting a large number of pseudo-neurons called *perceptrons*, each of them calculating a balanced sum of its inputs before passing the results through a non-linear function (a sigmoïd in most of cases) which simulates a logical threshold. The ANNs generally used in speech recognition are called *MLP* (Multi Layer Perceptron, see figure 2.15 on page 27).

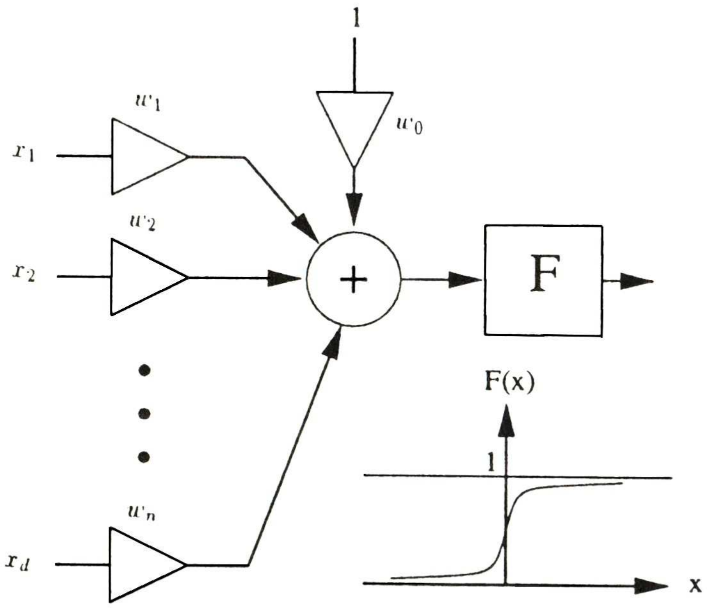

Figure 2.14: A perceptron calculates a balanced sum of its inputs and introduces an offset via w_0, before passing the results through a non-linear function, a sigmoïd in this case.

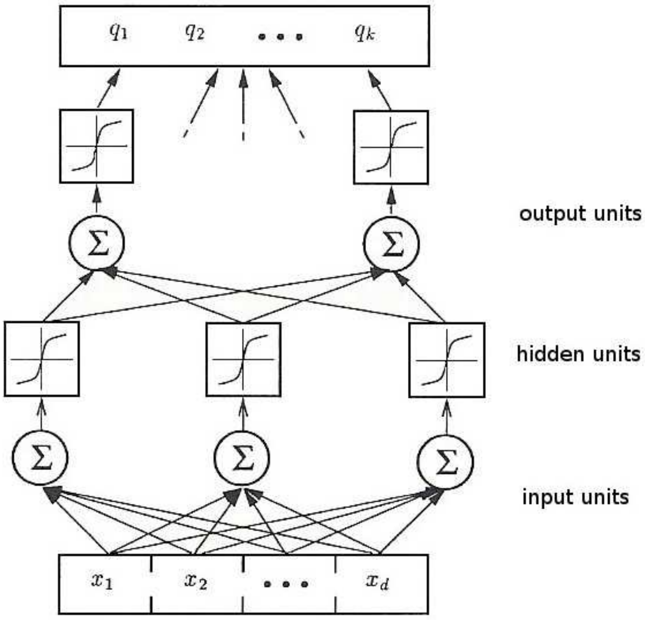

Figure 2.15: MLP architecture : it can model any non-linear function !

We can show that an MLP with one hidden layer outputs an **approximation of the state posteriors** $p(q_k|x_n)$ ([10]) if it is trained under the following specific conditions :

1. **the network is trained as a classifier, with one output neuron per class (per phoneme class in our case). It is trained to produce** 1 **at the output corresponding to the presented sample and** 0 **at the other outputs,**

2. **we use the Least Square Error (LSE) criterion (or the Relative Entropy criterion) as the training optimization function,**

3. **the network has enough hidden units,**

4. **the algorithm does not converge towards a local minimum.**

In this case, the ANN gives the phoneme posteriors $p(q_k|x_n)$[9](if we assume that states correspond to phonemes, what we will do during this report). Following Bayes law, we can write :

$$\frac{P(q_k|x_n)}{P(q_k)} = \frac{p(x_n|q_k)}{p(x_n)} \tag{2.24}$$

[9]Strictly, posteriors have to sum up to 1 to respect probability definition. To make certain, the output layer sigmoïd function is usually replaced by the **softmax** function ([10]) $\Phi(y_i) = \frac{\exp^{y_i}}{\sum_{k=1}^{K} \exp^{y_k}}$, where y_k stands for the input weighted sum of the k^{th} output neuron.

Equation 2.24 tells us that we can obtain **scaled likelihoods** by dividing the posteriors by the priors $P(q_k)$, which can be estimated manually from training databases (*e.g.* by counting number of appearances of each phoneme and divide by the number of all phonemes). $p(x_n)$ is constant and class independant, thus we obtain a value proportionnal to likelihood, the scaled likelihood, that can be used as the HMM states emission probabilities in equation 2.20. Figure 2.16 below summarizes the procedure.

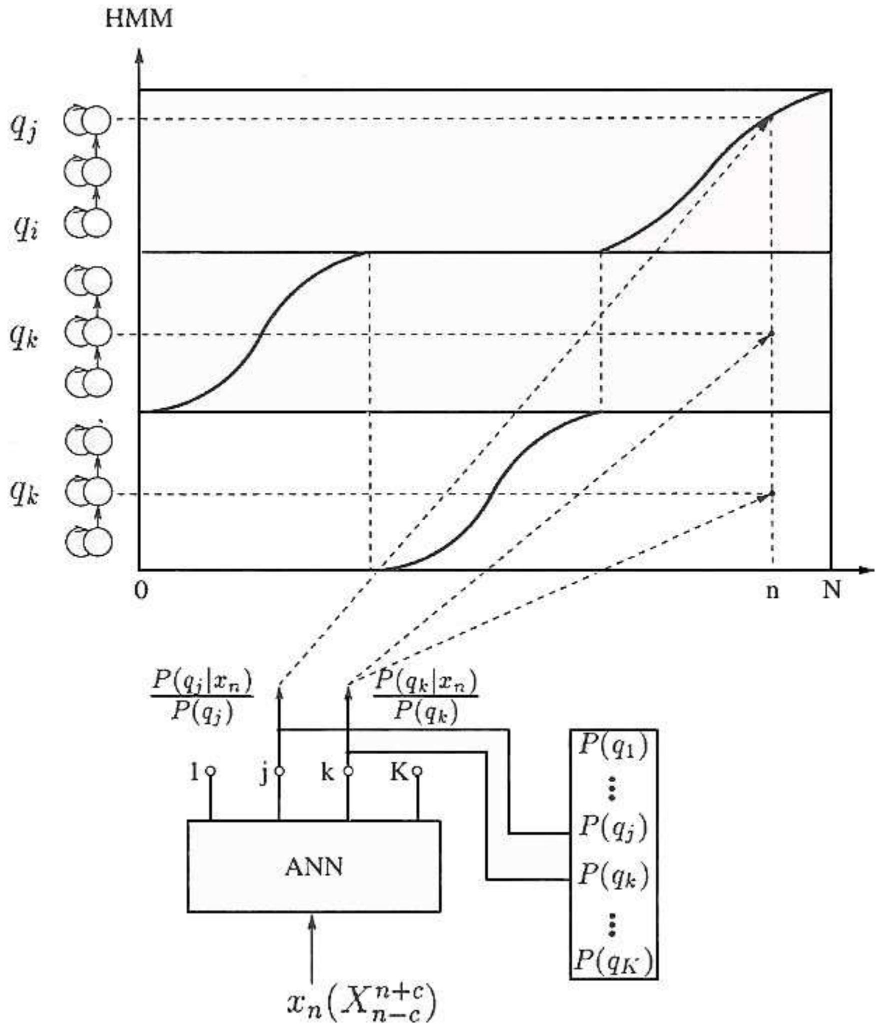

Figure 2.16: HMM/ANN hybrid decoding

The main advantage of using ANNs in conjunction with HMMs is that they provide a **discriminant** training. Indeed, the ANN training procedure boosts the output of the right class **while penalizing the other classes**, whereas training HMMs was performed by using standard Maximum-Likelihood procedures. Moreover, ANNs can include **long-context temporal information** by taking as input several acoustic vectors around the current frame, X_{n-c}^{n+c}, instead of only one, x_n (it has been shown that 9 yields good results). Finally, neural networks **don't require to make previous assumptions on the data distribution** (unlike HMMs, where we often start from a multigaussian hypothesis) : the training phase allows us to model any continuous data distribution.

To conclude, hybrid systems give another way to estimate the observation likelihoods than HMMs. They combine ANN advantages to the ability of HMMs to handle sequential information. Indeed, the training procedure has to be adapted (see [6] for further information). In most cases, hybrid HMM/ANN approaches, also refered as **connectionist** approaches, lead to better (or at least equal) performance than classical HMM systems.

2.4.3 Word Error rate (WER) : a performance measure

This section simply intends to present the performance criterion usually employed in speech recognition : the **Word Error Rate** (WER). This measure compares how the output word sequence differs from the reference sequence. First, the two sequences are aligned in such a way that the distance between them is minimized. This step gives us the minimum number of **deletions**, **insertions** and **substitutions** necessary to match the sequences. The WER is then computed as follows :

$$WER = \frac{S + I + D}{N} 100 \tag{2.25}$$

where :

- S is the number of substitutions (substitution of an original word by another in the original sequence),
- I is the number of insertions (insertion of an additionnal word in the original sequence),
- D is the the number of deletions (suppression of a word in the original sequence),
- N is the number of correct words in transcripts.

2.4.4 Advanced concepts

In this section, we will enumerate some advanced concepts currently used in ASR, without entering into details. Some of them need to be cited as they will be used in further chapters.

2.4.4.1 Advanced training methods

The training procedure exposed in section 2.4.2.3 corresponds to a Maximum Likelihood criterion, and thus lead to an algorithm lack of discrimination. Indeed, ML training adapts the model parameters to maximize the likelihood of the corresponding training sample, but without penalizing the likelihood of other models.

Other criteria have consequently been developed, in order to directly maximize the a posteriori probabilities (thus minimizing the probability of error) : we speak about Maximum A Posteriori (MAP) training [10]. The **Maximum Mutual Information** (MMI), the Minimum Phone Error (MPE) and the Minimum Word Error (MWE) criteria belong to this category (see [3], [48]).

We have also showed that ANNs could be used for a direct estimation of posteriors in combination with HMMs, thus offering their discrimant aspect to the entire system training (see section 2.4.2.5 on page 26).

2.4.4.2 Advanced decoding strategies

Classical Viterbi drawbacks We can denote two major drawbacks for Viterbi decoding. First, the recursion does not actually give us the optimal word sequence but the optimal state (*e.g.* phonemes) sequence. This may not be important, but sometimes, the most probable sequence of phones does not correspond to the most probable sequence of words. This case may occur when the lexicon contents words with multiple pronunciations : the probability to transit to such a word is split among the different pronunciation variants (as the transition probabilities leaving a state have to sum up to one), thus decreasing the word score ; and the decoder could select another wrong word with only one pronunciation as it has to choose only one path (the best). The second main disadvantage of Viterbi procedure is that it does not allow the use of high order language models, as we discussed in section 2.4.2.4 on page 24.

These problems are often solved according to two different approaches. We could first use **multiple-pass search** decoding techniques (see figure 2.17 below) : the most probable hypotheses are generated from a Viterbi first pass using a simple bigram language model and are stored in **N-best lists**

[10]Ideally, such training procedures would also have to estimate the optimal HMM topology.

or **word lattices**. A eventual further pass applies higher order language models on these hypotheses. The second solution implies a completely different decoding algorithm : we speak about **stack decoding** techniques, or A^* search, which relies on a complete likelihood estimation (summing all the path contributions, in the forward-backward decoding sense) on a tree which contains allowable word sequences (see chapter 7 of [29]). In this case, language models are decoupled from acoustic models and are not used to generate new recognition hypotheses.

Figure 2.17: Multiple pass decoding ([29])

Other approaches than MAP Most of ASR systems employ MAP classification based on Bayes law that presupposes equal cost to all misclassifications. **Minimum Bayes Risk** (MBR) classifiers intend to associate various weights to different type of errors, in order to build task dependent decoding strategies (see [18] for more information).

2.5 Conclusions

We have learned throughout this chapter the basic concepts we need to know to design a speech recognizer. This short conclusion will try to act as a (very) short summary. We have seen that a classic ASR system is divided into two main parts : the **acoustic front end** and the **decoder**.

The first phase, also named features extraction step, aims to reduct the useful information initially present in the speech signal, and to transform it into a sequence of acoustic vectors robust to variations, while remaining faithful to the lexical contents. The **Mel Frequency Cepstral Coefficients** (MFCC, see [19]) and **Perceptual Linear Predictive** (PLP, see [24]) features, extended with their first and second derivatives, are all commonly used in current ASR systems.

The decoding part intends to transform the sequence of acoustic vectors into a recognized word sequence. To perform it, the former **deterministic approach** stores reference templates (acoustic vector sequences) and applies dynamic programming to align them with the test sequences. A further backtracking phase yields the recognized word sequence by finding the selected templates. The **statistical approach** intends to represent words as statistical models, typically **Hidden Markov Models** (HMMs). We need in this case to formulate the recognition problem in term of probabilities, and the recognized word sequence is the one with the highest posterior probability (**Maximum A Posteriori approach**, or MAP). If X stands for the data and M_j for the statistical models, we have thanks to Bayes law :

$$\begin{aligned} j_{opt} &= \operatorname*{argmax}_j P(M_j|X) \\ &= \operatorname*{argmax}_j \frac{P(X|M_j)P(M_J)}{P(X)} \\ &= \operatorname*{argmax}_j P(X|M_j)P(M_J) \end{aligned}$$

Recognition with HMMs implies the estimation of $P(X|M_j)$, or **likelihood**, which can be performed through the **forward-backward** procedure (which sums the scores along all the possible state pathes), or thanks to the **Viterbi** approximation (which only take into account the most likely HMM state sequence). We speak about **acoustic** modeling.

HMMs are parametric models, thus before performing a Viterbi recursion for instance, a **training** algorithm is applied : parameters are estimated from audio data so as to maximize the likelihood of the models (Maximum Likellihood criterion, but other criteria, such as MAP training, also exist). The most common procedures are **Baum-Welch** or **Viterbi** training.

$P(M_j)$, or **language model** probabilities, are often estimated via N-**gram** models from large text databases.

An **hybrid** approach, which combines Artificial Neural Network (ANN) and HMM advantages (discriminant training, use of context information during recognition, no assumption on the distribution shapes for ANNs ; ability of handling sequential information for HMMs), has proved itself to be at least as efficient than HMM-based recognizers. Under specific training conditions, ANNs give state posteriors that are injected, after dividing by the state priors, in HMMs as state local scores.

The recognizer accuracy is often evaluated thanks to the **Word Error Rate**. Today ASR system WER reach more or less 25% for the most complex tasks (no elocution constraints, 50000 words vocabulary, different speakers).

Chapter 3

State-Of-The-Art in Keyword Spotting

The role of this section is to expose the word spotting problem, using the basic concepts previously exposed, and to speak about several related issues. We will also present in a few words the main available techniques and algorithms.

3.1 Keyword Spotting : General concepts

3.1.1 What is Keyword Spotting?

As we said during the introduction (chapter 1 on page 1), word spotting systems are designed to search words or group of words embedded in unconstrained speech. We said that such kind of systems were crucial for many applications : they could be used in dialog machines when Text-To-Speech (TTS) engines prompt a user to give a specific information for instance. In all the cases, the system role is to extract this useful information without taking into account the rest of the sentence. To a certain extent, we can say that the subject shares some features with the task of topic, speaker and language identification, where some patterns have to be detected and others have to be ignored.

In a keyword spotting (KWS) task, several issues need to be treated.

First, we have to decide which events will be significant. In a word spotting system, they consist in either a **true keyword detection or a false alarm** (when the system detects a keyword though it has not been pronounced). Considering these putative hits, thresholds are often used to distinguish true keywords from high scoring false alarms, as we will see in chapter 4.

We have also to think about which **acoustic unit** using in the system : phonemes, sub-phoneme units, syllables or even entire words ? In case of HMM-based keyword spotting for instance, three state HMMs could model phonemes, and words would be created by concatenating of such smaller models.

The size of the keyword vocabulary is also important. In most cases, 20 or 30 keywords are considered, keyword spotting is thus a **small vocabulary problem**, with consequently a **simple syntax compared to Large Vocabulary Continuous Speech Recognition** (LVCSR) systems[1]. Obviously, smaller the vocabulary size will be, better the system will work.

We said that in word spotting, only the keywords matter in a sentence. But what about the rest ? The system has obviously to detect the keywords but has also to be able to deal with extraneous speech ! In that way, it is indispensable to model non-keyword speech by using **garbage (or filler, or sink) models**. The methods to model them, as well as the type and size of the training data used to train them consist in a main issue in the field of word-spotting, leading to an incessant flow of paper publications.

Spotting words can be very time consuming, and this phenomenon increases with the vocabulary size. The **system speed** does not have to be neglected ! Sometimes, applications in word spotting require the ability to spot large amount of data at many times faster than real time (*e.g.* message retrieval, processing of meeting databases,. . .). In this context, solutions have been developed (James & Young, 1994 in [27])

The **vocabulary flexibility** has also to be considered. Indeed, in some applications, such as message retrieval, the keyword vocabulary is not fixed and changes at each user request ! Techniques have been investigated to deal with that problem ; among them, we will briefly speak about James & Young's fast lattice-based approach (see section 3.3 on page 40).

Note that word spotting techniques vary with the use of **language modeling** (see section 3.1.3 below).

Several databases had been widely used to develop and test word spotting systems. In the U.S.A, we can denote two. The *Roadrally* corpus, distributed by the National Institute of Standards and Technology (NIST, 1991), containing conversations, read paragraphs and/or read keyword sentences by more than 150 speakers, has been recorded with the hope of creating a standard word spotting database. The *Switchboard* corpus (1992) consists in 250 hours of spontaneous conversational speech collected under computer control without human intervention. It is a DARPA-sponsored

[1]This affirmation is correct for most KWS systems, but not for LVCSR-based keyword spotters. We will study them in more details in chapter 4.

corpus developed at Texas Instruments, and has been selected by the NIST as a standard for future word spotting tasks. Note that multi-national databases have also been recorded, such as the SAMOGO (Speech Activated Manipulation Of Graphical Objects) database which has been used to study garbage modeling with one or several garbage models (1992). Today, the current tendance is to search for keywords in large meeting, or conversational, databases.

3.1.2 Garbage models

We emphazised above the importance of garbage models, which role is to represent non-keyword speech. The system performances will easily depend on the way it will be modeled : better modeling of non-keywords will of course reduce the amount of false alarms. We can for example use template fillers, as Higgins & Wohlford did in 1985 ([25]), but the HMM nature will provide a better way than templates to model garbage. Indeed, HMMs offer the possibility of being less selective (by playing with the connections betweeen states for example) and handle better the characteristics of extraneous speech, namely its large variety.

The main issues in using filler models remain the training procedure and the number of models to use. Three approaches have been investigated :

1. training one or several HMMs to model the entire background environment (noise and extraneous speech),
2. using the models which describe lexicon words to define the garbage,
3. computing garbage scores directly "on-line" without explicitly defining garbage models.

The first approach seems to be the most natural : we can for example use a fully-connected HMM to model the entire background. In this case, the danger consists in a risk of confusion between keywords and garbage. Note that a few number of universal garbage models could outperform a system with a large number of models (Wilpon *et.al.* in 1990, [51]).

Thanks to the progress made in CSR, we can use now acoustic speech units to model garbage : Rose & Paul used triphones (phonemes that handle left and right context, see *e.g.* the lower part of figure 3.3 on page 41) and monophones in 1990 ([42]) and Bourlard *et.al.* used context-independent phonemes in 1994 ([5]). In his paper, the latter showed that it is often necessary to define keyword entrance penalties, and garbage entrance penalties in case of several garbage models, to improve recognition rate. For example, in this last case, these factors will penalize the transitions between the garbage classes and thus improve the keyword score. The figure 3.1 below shows such a grammar network with the two kind of penalties.

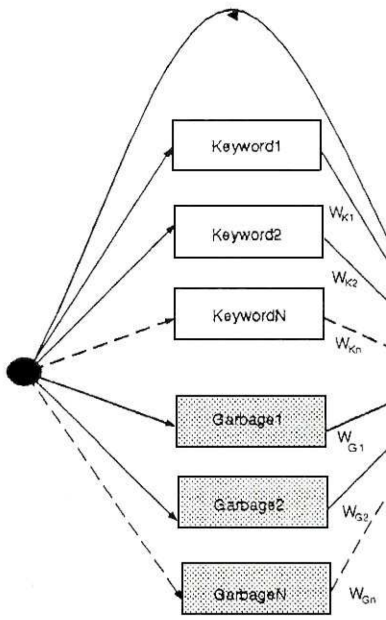

Figure 3.1: A grammar network with keyword and garbage entrance penalties (chapter 10 in [28]). Bourlard et.al. showed in 1994 that they need to be optimized empirically to get the best performances.

The third approach, developed by Bourlard *et.al.* in 1994 ([4]), computes the garbage local scores *on-line* for each frame by taking the average of the N-bests other acoustic units local scores. Using this method, we understand easily that the garbage score will never be the best one at the **local** level, and that garbage will only be recognized thanks to its **global** score, *i.e.* if the global match of keywords with the utterance is bad. Moreover, we can say that garbage is not **explicitly** modeled in the way that no garbage HMMs are trained at the beginning, but are used during the Viterbi decoding phase. It is important to add that N appears in this approach as a parameter which needs to be tuned to get the best performances. This method has been shown to work as well or better than more traditionnal methods which explicitely define garbage models. The interested reader can refer to my internship report, where we developed a system based on this technique, to consult some results ([12]).

3.1.3 Keyword Spotting and Large Vocabulary Continuous Speech Recognition

Large Vocabulary Continuous Speech Recognition (LVCSR) and Keyword Spotting seem to be very close disciplins : KWS is often considered as a continuous phoneme recognition problem on spontaneous speech with no (or simple) syntax model and with a small vocabulary size. Besides, some word spotting techniques are based on a further search on the hypotheses previously generated by a LVCSR recognizer ([47] for a search on word lattices, [45] for a search on phoneme lattices, [49] for a search on N-best lists, and chapter 4). But several differences subsist (*e.g.* by assuming that in KWS, no speaker training is allowed, unlike some LVCSR cases, increasing the problem complexity), especially in the field of training, defining garbage models and language modeling.

We spoke about the garbage modeling issue above during the section. But it is important to add that in word spotting, training garbage models become one of the most important problems. For example, when the non-keyword vocabulary is large and when there is a limited training corpus (a few keyword training tokens), garbage models (but it is also valid for keywords models) can be shared across similar context using context-dependent or context-independent phonemes (Rose et Paul, 1990, [42] and Rohlicek *et.al.*, 1993, [40]). But on the other hand, when a large amount of only keyword training data is available, whole-word modeling can be used and lead to very good performances (Rohlicek *et.al.*, 1993, [40]).

Note that it is also possible to use simple semantic constraints to improve the performances in word spotting (simpler than in LVCSR). We can for example include an *a priori* knowledge of the application domain. Syntax constraints can also force the system to detect at least one keyword per utterance, depending on the application type, in order to get a better error rate. All this improvements concern the language modeling issue. Besides, some techniques work on a lattice representation of the initial search space, generated using an LVCSR system for instance, and thus include complex syntactic and lexical information in the word spotting task.

3.2 Performance measures

To evaluate a system designed to spot words or group of words, we need to introduce some kind of performance measurement.

The NIST has defined the Figure Of Merit (FOM) as the average of the scores up to 10 false alarms per keyword per hour :

$$FOM = \frac{(p_1 + p_2 + p_3 \ldots + p_N + a p_{N+1})}{10T}, \tag{3.1}$$

where :

- p_i is the percentage of true hits found before the i^{th} false alarm,
- T is a fraction of an hour for test talkers,

- N is the first integer $\geq 10T - \frac{1}{2}$,
- $a = 10T - N$ is a factor that interpolates to 10 false alarms per hour.

With this evaluation tool, we don't need to compare scores across keywords anymore ; and we get also a more stable measure than the detection rate at zero or one false alarm per hour, which has a large variance. For example, such a FOM was introduced in 1989 by Rohlicek *et.al.* ([39]), which can be observed on figure 3.2 below.

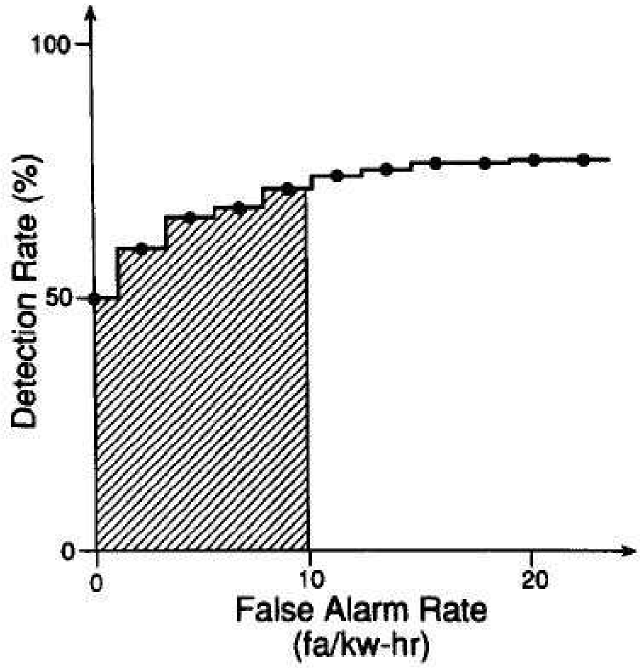

Figure 3.2: By setting and varying a threshold compared with a score (typically a global likelihood) they obtained during the decoding phase, Rohlicek et.al. got operating points which consist in various detection rate for different false alarm rates per hour of speech (ROC curves, see below). They took the average value of the shaded area to approximate the FOM ([39]).

Figure 3.2 introduces another way to evaluate the system performances : the ROC curves, or Receiver Operating Characteristics, which consist in plotting true detection rate versus false alarm rate. Marcus *et.al.* proposed in 1992 ([33]) to take the area beneath the curve as a performance measurement to get a measure independent from the operating point : the larger the area, the better the performance is ; and the word-spotter is able to better discriminate keywords from garbage. Note that to get various operating points to plot ROC curves, we need to make some model parameters (such as keyword or garbage entrance penalties or the value of N in case of on-line garbage modeling) or some thresholds (as explained on figure 3.2 above) values varying, depending on the technique used to design the system[2].

3.3 Available algorithms and techniques

Several techniques are available to deal with the word spotting problem ; and we can try to classify them following various criteria.

We can first distinguish **template-based** word spotters. Higgins & Wohlford used in 1985 concatenated templates ([25]), obtained from various ways, to model the entire input speech. In this pionneer approach, keyword templates competed for the best score with the other templates representing the alternatives. According to that, a keyword can be detected even if the input utterance generates a bad match with the keyword template but higher than with the others.

HMM-based techniques seem to provide a better alternative thanks to their natural ability to handle the entire background, thus to better model garbage leading to better performances (Wilpon *et.al.*, [51]). For example, Rohlicek *et.al.* used left-right HMMs to model keywords, with one or three states for each phoneme, and used alternate models for non-keyword speech (phoneme loops or single HMM state trained on the entire database), as it can be seen on figure 3.3 below (1990, [39]). This figure, proposed by Szöke *et.al.* in [45] and selected because clearer than the original, consists in a adaptation of the system proposed in [39] in which they introduced some simplifications to run it on-line.

Other HMM-based techniques also exist, but mainly differ by **the way garbage is modeled** (Bourlard *et.al.* in 1994, [5]), as it has been said on section ; where we exposed the three approaches usually used : training one big HMM to model the entire background (non keyword speech and noise), use lexicon units to model garbage, and not explicitly defining garbage models. The latter is an approach particularly investigated by Bourlard : we note his on-line garbage modeling as previously defined ([4]) but also techniques using specific confidence measures based on posterior probabilities (Bourlard & Silaghi searched the segment maximizing the average observation posterior along the most likely path in the hypothesized keyword model in [11]). Note that techniques also differ by the way scoring methods are defined (*e.g.* thresholding), by the way acoustic units are specified (context dependant or independent phonemes, whole-word modeling. . . see below) and by the way the overall HMM structure is defined. (Rohlicek *et.al.*, 1993, [40])

[2]Other scoring techniques exist, taking for example into account the difference of duration between true keyword detections and false alarms, but are more specialized and will not be developed in this short chapter ([54] for instance).

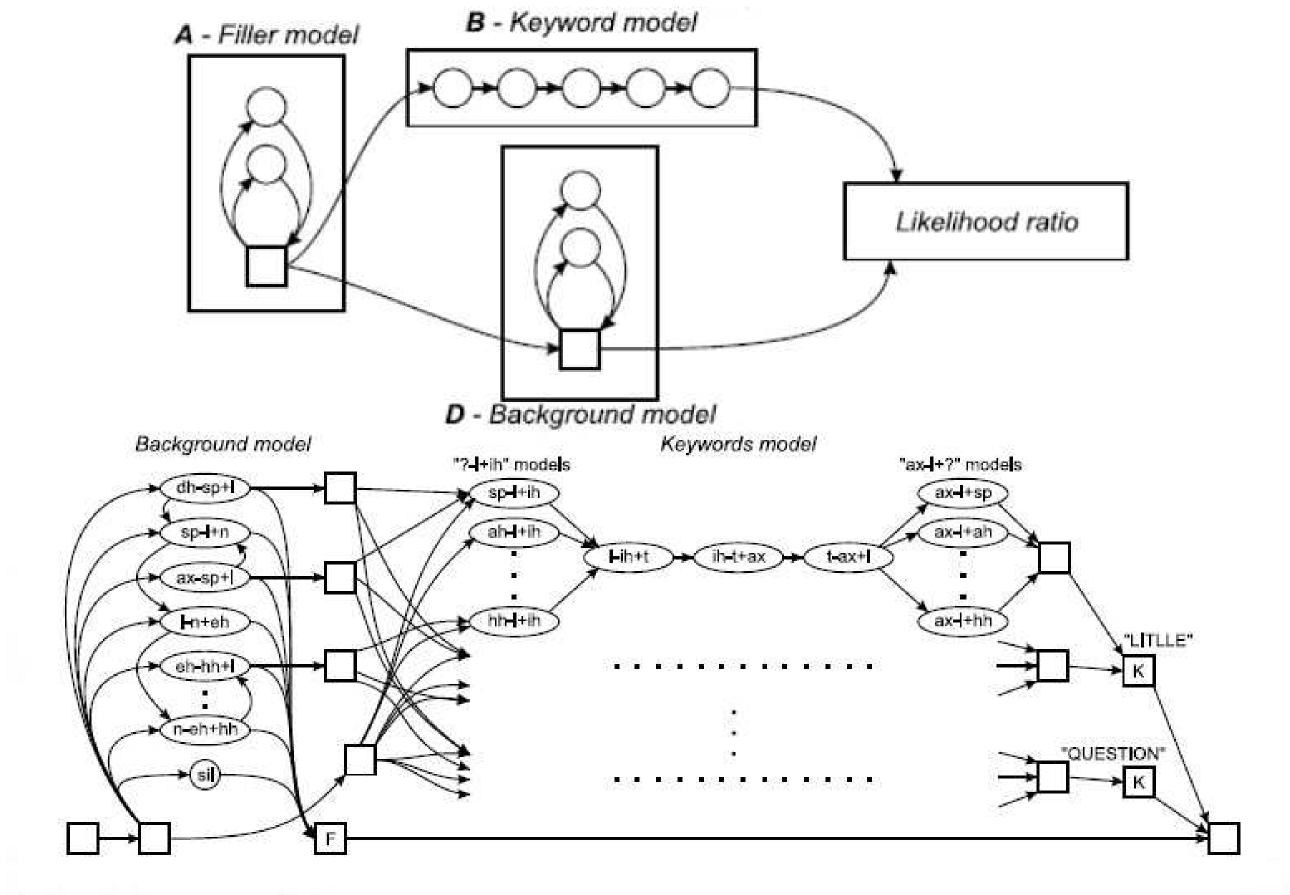

Figure 3.3: On the upper part of the figure can be observed the general structure of an HMM based keyword spotter, following [39], where alternate models (filler and background models) are made from loop on keyword units. Under you can see an adaptation given in [45] in the case of context independent phonemes (triphones here) used as keywords and garbage units.

We can also denote **hybrid neural network/HMMs** based techniques. Zeppenfeld *et.al.* ([54]) developed in 1993 a system based on MS-TDNN, for Multi-State Time Delay Neural Network (*cfr.* figure 3.4 below). Szocke *et.al.* also developed the LCRC Feature Net system, for Left Context Right Context Feature Net system, which combinates neural networks with Viterbi decoding without any language model ([45]).

A classification criterion can also be found if we consider that some techniques use **whole-word modeling** (*e.g.* Rose & Paul, [42]), and others **sub-word models** (*e.g.* Rohlicek *et.al.* in 1993, [40]). The first method has been shown to perform well in cases where only the keyword locations are known, and not their phonetic transcription, and when a large amount of only keyword training data is available (*e.g.* when we spot digits). In the second approach, all the words are decomposed into phonetic units, and different words can benefit from the training data related to the other words

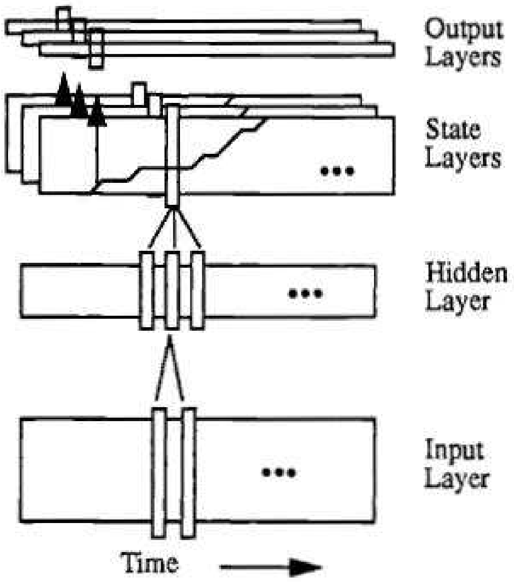

Figure 3.4: This keyword spotting network consists in an input layer and a hidden layer, connected to a state layer and an output layer for each word to be spotted. The activations for all units and states in the hidden and state layers are found using a standard TDNN feedforward network algorithm. At the state layer, each keyword to be spotted is represented by a series of independent states. A dynamic programming algorithm is performed starting at each time frame in order to find the best path through these state activations. The score of this optimal path represents the output score for the keyword at the time frame in question. The network thus outputs a score for each keyword at each time frame ([54]).

which share same phonetic pieces. This method works good when we have a large non-keyword vocabulary and when we dispose transcribed speech data with a few keyword tokens.

Some particular applications, such as message retrieval, necessit a trade-off between speed and keyword vocabulary flexibility. In these cases, the lexicon words change at every query, and the spotter need to work faster than real time ! Techniques have been developed to deal with this vocabulary independence and the high processing speed required. Among them, we denote some **lattice based** techniques (James & Young in 1994, [27]). In this approach, spoken messages are first processed to obtain vocabulary independent content (with a modified Viterbi phone recognizer here), and stored in a compact intermediate form once for all for each message (a phoneme lattice, in which phones hypothesis are stored for each time point). Then, the word spotting step consists in a fast symmetric dynamic programming phase which tries to match the keyword pronunciation with the hypothesis, following a specific criterion. There is thus no use to rerun a slow keyword-dependent word spotter each time the user wish to search new words. In [47], log-likelihood ratios are computed from LVCSR word lattices (while they are computed from N-best lists in [49]) in order to generate keyword spotting hypotheses. Usually, word lattices give more accuracy for the spotting task, but are not able to deal with Out Of Vocabulary (OOV) words. The technique we will study in chapter 4 belongs to this category.

To finish this section, we will mention that good performances imply good rejection (or acceptance) capabilities. In other words, true keyword hits have to be separated from high scoring false alarms. To deal with the algorithms lack of discrimination (the word spotter decision is indeed based on how well a word hypothesis is *compared to others*), several **post-processing** techniques have been introduced, including so additional knowledge and processing. A post-processor takes as its input the scores from the main recognizer (the wordspotter first-pass) and computes alternative scores to make the final decision : keyword name, and its acceptance or rejection. Several techniques exist and differ by their complexity. Figure 3.5 below shows the five main categories ; but we will not explain each of them into details (you can refer to chapter 10 of [28] for more information).

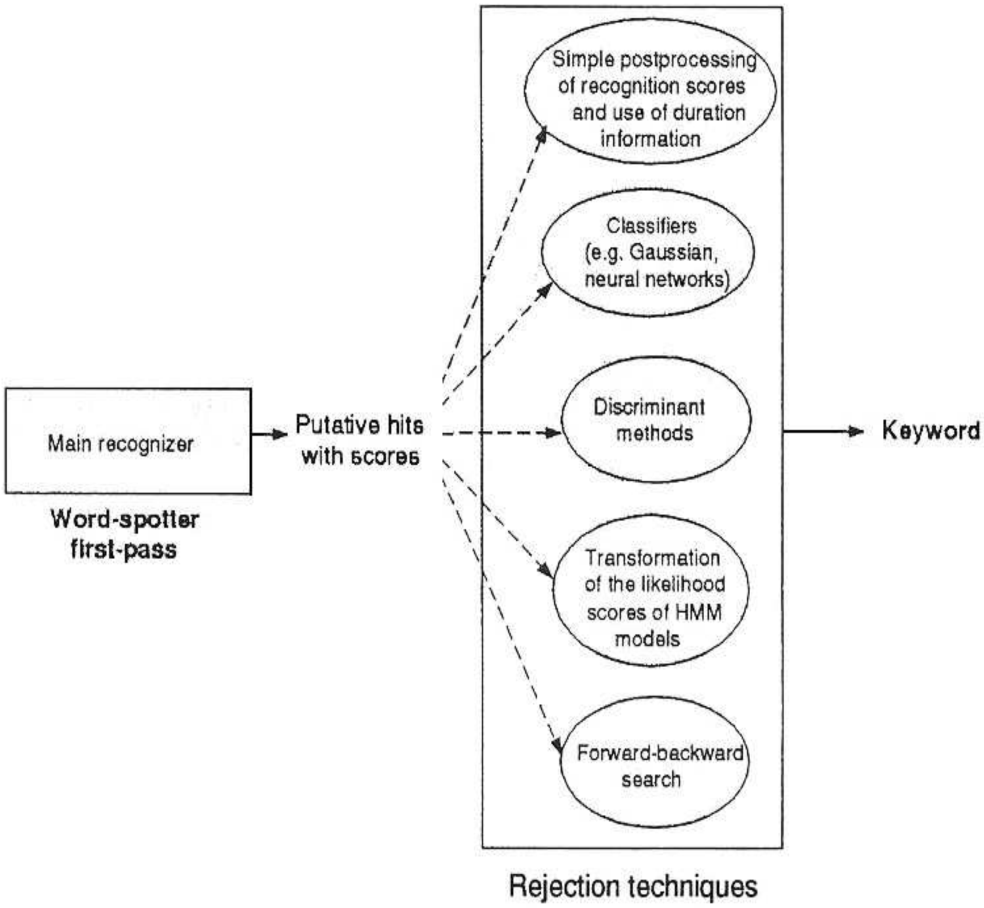

Figure 3.5: Classification of post processing techniques into five categories ([28]).

3.4 Out-Of-Vocabulary words : the OOV problem

Even when using a large vocabulary speech recognizer, it is not possible to have a complete coverage of the vocabulary that will be pronounced. In this context, the system will often have to deal with words it doesn't know. However, following the techniques currently used in LVCSR, even when it faces an unknown word, it will always recognize a word inside its existing vocabulary. Moreover, mistakes can also occur in the neighboring area of the unknown word. We can easily understand that techniques used to model garbage in keyword spotting will be very useful to deal with this kind of problem ! Generally, the task can be divided into two parts independently solvable : the detection of the new word, and its addition to the existing vocabulary (see figure 3.6 below).

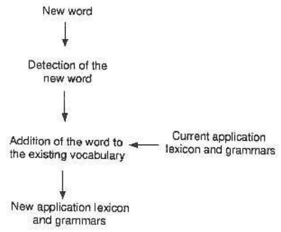

Figure 3.6: Dealing with out-of-vocabulary words (chapter 10 of [28]).

For the detection phase, garbage models are used to locate and detect new words. Asadi *et.al.* ([1]) used in 1990 explicit models to represent new words. They used context-independent phonemes, each of them modeled by a three state HMM, and created a three state explicit model for the new word containing all the phoneme transitions in parallel from the first to the second state, the same from the second to the third state, and all phonemes in parallel looping on the second state. Such a model shows good performances detecting new words, and does not reach high false alarms rates (this kind of model does not recognize words from the existing vocabulary).

Once a new word is detected, we have to add it to the existing vocabulary. This process will highly depend on the type of speech recognizer used. If whole words are modeled, we can add a new word by training the system on it, but if the system is phoneme-based, we need phonetic transcriptions of the new word (*e.g.* from its spelling, but also from text-to sound rules as used in a text-to-speech synthetizer, *cf.* Asadi *et.al.*, 1991, [2]). In the latter case, HMM models can be built once the phonetic transcription is obtained, and the new word can be added to the existing vocabulary, without forgetting to adapt the language model (new syntax transition probabilities,. . .).

3.5 Conclusions

We spoke during this chapter about a really topical subject : the Keyword Spotting (KWS) problem, which consists in spotting a word, or a group of words, embedded in extraneous speech and noise. Current applications really need to deal with this problem : more and more systems need to interact with their users, and voice tends to become a highly appreciated man-machine communication mode. In this context, we understand the utility of the system finding useful words among all the extra data given, often involuntarily, by the user.

We saw that many issues were involved in design and conception of word spotting systems, offering us classification criteria to differentiate them :

- the main issue in keyword spotting remains the so-called **garbage models**, used to model non-keyword speech ; and better they are estimated, better the system can discriminate keywords from false alarms (when the system detects a keyword though it has not been pronounced). We saw that Hidden Markov Models (HMMs) gave a natural way to handle the entire speech variability. Three approaches are widely used :
    1. training one single HMM that model the entire background, *i.e.* extraneous speech, noise and silence (Wilpon *et.al.*, [51]),
    2. using the acoustic units from the lexicon, such as context-dependent or context-independent phonemes (Bourlard *et.al.* in [5], Rose & Paul in [42]. . .),
    3. not explicitly modeling garbage classes, and playing with the way scores are defined for recognition, as in *on-line garbage modeling*, (Bourlard *et.al.*, [4] and [11]),
- **speed** is also an important factor ; some systems (*e.g.* message retrieval systems) need to work many time faster than real time. Techniques have been developed to perform this task ; a preprocessing step can for instance store and decompose all the spoken messages in **phoneme or word lattices, or N-best lists** , used after during a fast decoding phase (James & Young in [27], Szöke *et.al.* in [47], Weintraub in [49]),
- KWS is obviously related to the general speech recognition problem, and its algorithms are based on the ones from ASR. We denote :
    1. **template-based** word spotters (Higgins & Wohlford in [25]),
    2. **HMM-based** word spotters (Bourlard *et.al.* in [5], Rohlicek *et.al.* in [39] and [40], Rose & Paul in [42], Wilpon *et.al* in [51],. . .),
    3. **hybrid HMM/ANN based** word spotters (Zeppenfeld *et.al.* in [54]),

- techniques also vary with the type of acoustic unit. **Whole word** modeling is used when only the keyword locations are known in a large keyword training database (Rose & Paul in [42]) ; **subword acoustic units**, such as context dependent (triphones) or independent phonemes, are used with a large non-keyword vocabulary when a few keyword tokens in the phonetic transcribed database are available (Rohlicek *et.al.* in [40]),
- **scoring methods** greatly influence the variety of algorithms, because they are always based on some kind of thresholding on various scores to make the final decision. But two performance measures have emerged, even if a lot of papers are now published to impose more robust measurement techniques :
  1. **Figure Of Merit** (FOM) represents the average of the scores up to 10 false alarms per keyword per hour (Rohlicek*et.al.* in [39]),
  2. **Receiver Operating Characteristics** (ROC) curves plot true detection rate versus false alarms, and the area beneath the curve is taken as a performance measure (Marcus in [33]).

  Note that **post-processing** techniques are now emerging, getting as input the classical scores out from the word spotter and applying some processing to give the final decision and increase the rejection rate,
- some applications do not require a fixed vocabulary : in a message retriever, vocabulary words change at each user query. We have just pointed the problem of **vocabulary flexibility** (James & Young in [27])

All these issues could appear to be too many to perform a clear classification. Thus, to summarize, we will now restrict the classification by defining three larger categories :

1. **garbage-based** techniques. Such word spotters intend to explicitely model non-keyword speech, thanks to HMMs or templates. The main variations into this category remain the garbage HMM topology, the acoustic unit choice,. . . We speak about **acoustic keyword spotting**,
2. **no-garbage modeling**. With these techniques, garbage is not explicitly modeled (in sense that we do not train a HMM on non-keyword speech for instance). The **On-Line Garbage Modeling** approach ([4]) is the most important among them,
3. **lattice-based** techniques. A LVCSR first pass restricts the initial search space by storing the hypotheses under the form of **lattices** (phoneme based, like in [27], or word based, [47]), or N-best lists ([49]). Keyword spotting algorithms are further applied on them. Such kind of

approach is for instance well-adapted when we search for keywords in a large vocabulary database, within an information retrieval framework for instance.

Moreover, we saw that KWS can help Continuous Speech Recognition (CSR) by offering its garbage modeling techniques to deal with the out-of-vocabulary (OOV) words problem. Indeed, current CSR systems recognize a lexicon word even when an OOV word is pronounced. To overcome this difficulty, systems have to be able to first detect the new words (thanks to KWS techniques !) and after to add it to the existing vocabulary (Asadi *et.al.*, [1] [2]) .

Chapter 4

Posterior-based Keyword Spotting on Word Lattices

4.1 Context

We exposed during the previous chapter state-of-the-art keyword spotting (KWS) strategies. Three principal categories emerged from this analysis. Acoustic keyword spotting intends to train garbage models on non-keyword data in order them to generate higher scores than keyword models when non-wanted data is presented at the system input. The main issue in this category remains the way garbage models are defined (HMM topology, acoustic units...). Another class of systems does not explicitly model non-keyword speech ; garbage scores are computed from acoustic keyword local scores in such a way that they are never the best ones at the local level, while they can lead to a match at the global level. These are on-line garbage-like techniques. Finally, we can also note **lattice-based** KWS approcahes, where the initial search space is reduced using a Large Vocabulary Continuous Speech Recognizer (LVCSR) first pass by storing the hypotheses under the form of **word lattices** or **phoneme lattices**, on which further KWS algorithms are applied. The main advantage on using phoneme lattices is that they consist in a vocabulary independent representation of the search space. In other words, such systems can search for any desired keyword by using its phonetic transcription. On the other hand, we saw that word lattices could help us to get more accurate results.

In this chapter we intend to focus on the last category by developing a word lattice based keyword spotter. Such approaches are particularly well adapted when searching for keywords in large real databases (see figure 4.1), such as meeting or telephone conversation databases, while simpler techniques, as On-Line Garbage modeling and acoustic KWS, are not sufficient to deal with the large vocabulary.

Figure 4.1: Keyword spotting on large databases, using word lattices as an intermediate representation of the search space. We could imagine using this technique in a system which aims to find all the meetings, or all the telephonic conversations, where a specific keyword would have been pronounced.

The main idea consists in first transforming and reducing the initial search space using a multi-pass LVCSR system, storing the most probable hypotheses under the form of word lattices. By doing so, more knowledge (lexical and syntactic) is taken into account than tradionnal single-pass, one-best approaches. That search space pruning allows us to secondly estimate keyword hypotheses posteriors from it. As posteriors minimize, by definition, the probability of error, we thus aim at minimizing the false alarm rate while maximizing true detection rates.

Such an approach is however more demanding in computational ressources, as a full LVCSR multi-pass is first run on the data. But this apparently drawback can be overtaken if we use our system in the context of an information retrieval task. Indeed, messages can be preprocessed immediately during recording and stored in word lattices. A further faster than real time, and more elaborated, keyword spotting algorithm can be after applied at each user request. In this context, experiments will be performed on Conversational Telephone Speech (CTS) database, more precisely on CTS data given within the NIST Spoken Term Detection evaluation framework (2006).

This chapter will be organized as follows. We will first begin with a global system description, before exposing each step with more details (section 4.2). We will also analyze the impact of lattice acoustic scores estimation. In a first attempt, we will run our system on the original lattices, the ones generated by the LVCSR first pass, before replacing all their acoustic scores by their estimation using an HMM/ANN forced Viterbi alignment. Section 4.3 on page 63 will adress this issue. Then, other considerations, such as tuning of scaling factors needed to estimate the posteriors, or the measure of the initial search space lattice coverage, will be discussed. The chapter will end with the system evaluation based on experiments performed on CTS data, and with its comparison with the On-Line

Garbage Modeling approach.

4.2 A posterior based LVCSR Keyword Spotting system

4.2.1 System overview

Figure 4.2 on page 50 gives an overview of the entire system. It can be divided into 3 main steps.

First, Perceptual Linear Predictive (PLP) coefficients are extracted from the speech waveform and are gathered in acoustic vectors (section 4.2.2 on page 51).

Then, word lattices are generated by using a multi-pass LVCSR system we will describe in section 4.2.3 on page 52.

Keywords hypotheses are further extracted from the lattices and scored by estimating the keyword posteriors given long context data $p([KW; t_s, t_e]|X_1^T)$ (where $[KW; t_s, t_e]$ represents a keyword hypothese begining at time t_s and ending at time t_e, and X_1^T the entire data, from time $t = 1$ to $t = T$). Then, each keyword hypothesis is rescored by accumulating the posterior probability mass splitted among the hypotheses which overlap in time (section 4.2.4 on page 57). Hypothesis scores are then compared to a threshold in order to take the final decision.

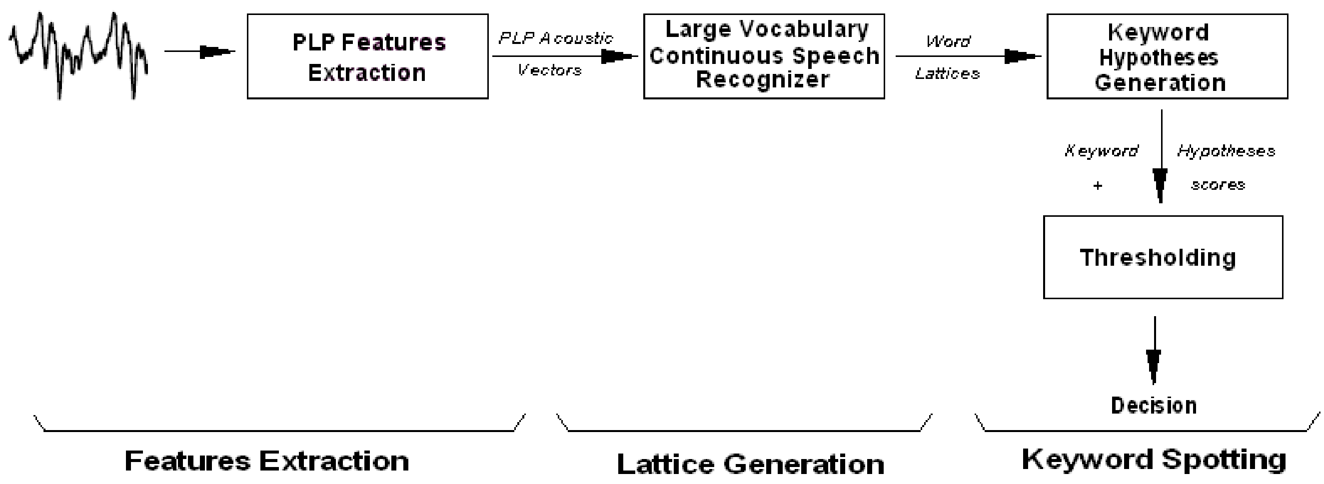

Figure 4.2: Posterior based Keyword Spotter, on word lattices generated by a Large Vocabulary Continuous Speech Recognizer. Keyword hypotheses are extracted from lattices and then scored by estimating $p([KW; t_s, t_e]|X_1^T)$.

4.2.2 Features extraction : the PLP features

In our system, speech features are extracted from the speech waveform using a Perceptual Linear Predictive analysis. This section tries to give an overview of the procedure, based on [24]. PLP speech analysis models the auditory spectrum, which is a spectrum that takes into account perceptual properties of human auditive system, by the spectrum of a low-order all-pole model. As can be seen on Figure 4.3 below, PLP features computation can be divided into two major steps, each of them composed from several substeps.

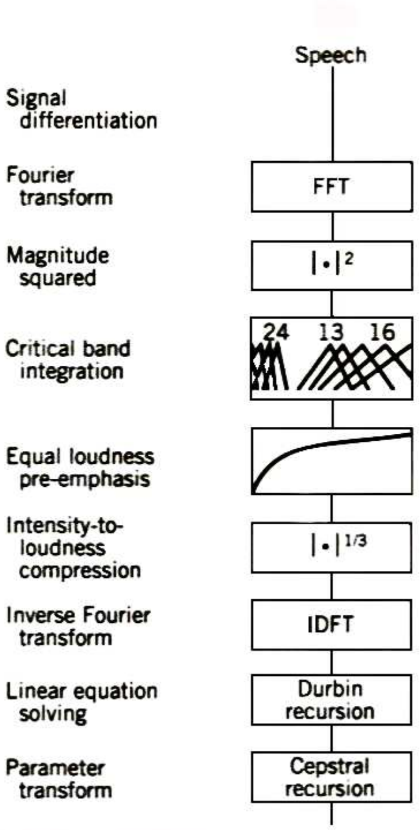

Figure 4.3: Steps in the computation of the PLP features ([19]).

First, the auditory spectrum is obtained. To that end, the power spectrum is estimated by computing the square magnitude of the windowed-signal Fast Fourier Transform (FFT). Then, the power spectrum is integrated within overlapping critical band filter responses. To do this, the frequency axis is warped along Bark scale (which take into account perceptual properties of human ears), and the power spectrum is weighted by trapezoidal functions (equally spaced in Bark frequencies), simulating the filter bank. Each output (one for each simulated filter, often 18) is after multiplied by an equal loudness curve to approximate the unequal sensitivity of human ears at different frequencies. The spectral amplitudes are then compressed following the Steven's power law, to reduce amplitude variations of spectral resonances. To finish, all these outputs are linearly interpolated to give a representation of the auditory spectrum. To summarize, we started from a time/amplitude domain (speech waveform) to reach a tonality (Bark scale)/loudness domain (the auditory spectrum).

The second step aims to approximate the auditory spectrum by the spectrum of an all-pole model (usually fifth order). An inverse FFT is performed on the spectrum, giving autocorrelation-like coefficients (they come indeed from a compressed spectrum). Then, Durbin recursion is applied (as for all-pole LPC speech modeling) to smooth the compressed critical band spectrum, leading to the auditory spectrum enveloppe, with usually one or two major peaks. The last step converts the autoregressive coefficients to orthogonal variables thanks to cepstral recursion.

In this work, we used 39**-dimension PLP acoustic vectors** : 13 PLP parameters (including log-energy), 13 for the first derivative, 13 for the second derivative.

4.2.3 Word lattices generation

4.2.3.1 The concept of word lattices

We have previously defined the word lattices as a compact way to store multiple hypotheses generated by a recognizer. We will now try to define them in a more formal context.

According to the formalism exposed in [50], word lattices can be viewed as directed, acyclic and weighted graphs, defined by a set of **nodes** and a set of **links** (or edges). The nodes represent discrete points in time, and their links word hypotheses $[w_i; t_s, t_e]$, for the hypothesis to have a word w_i starting at time t_s and ending at time t_e. With each edge is associated a word label, an acoustic likelihood $p(x_{t_s}^{t_e}|w_i)$ (the probability with which test data X between times t_s and t_e is generated by the model of word w_i) and a language model probability $P(w_i|w_{i-1}, \ldots, w_{i-(n-1)})$ (in case of an n-gram language model).

A word lattice can thus be regarded as a **reduced representation of the infinite space of possible solutions** for the maximization problem defined in Bayes' decision rule.

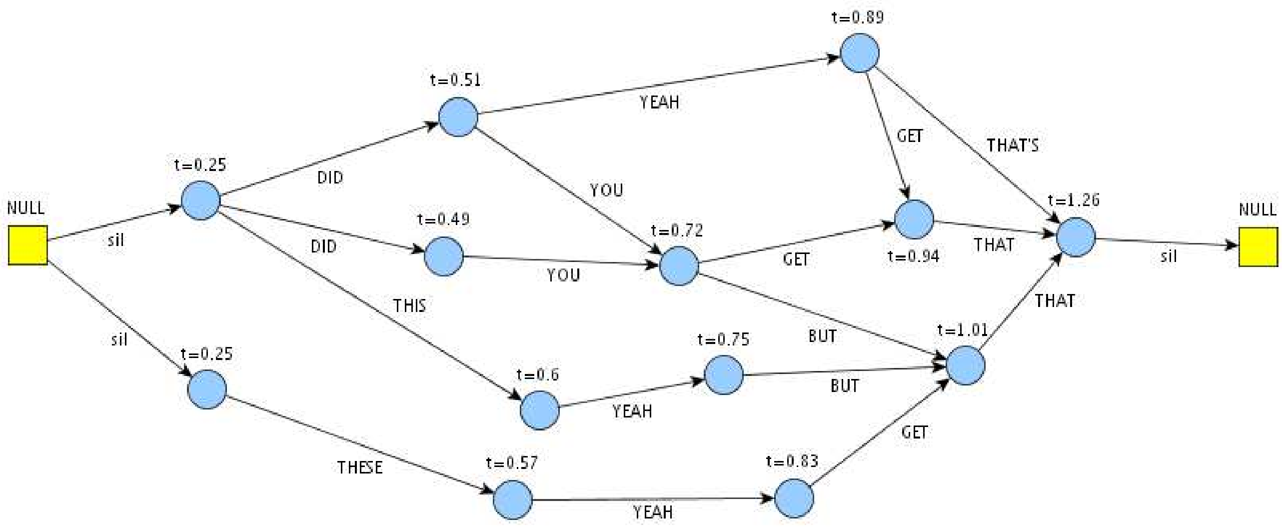

Figure 4.4: A word lattice. Acoustic likelihoods $p(x_{t_s}^{t_e}|w_i)$ *and language model probabilities* $P(w_i|w_{i-1},\ldots,w_{i-(n-1)})$ *have not been represented for convenience.*

Figure 4.4 shows a word lattice for the sentence "Did you get that ?". We can see that NULL nodes are introduced (they are used for a lattice computer representation). We can moreover observe the presence of silent edges ("sil").

Before finishing this section, we will adress a question that may come at mind. We said during chapter 2 that high-order language models could be applied on word lattices. But basing on the above definition of lattice, how could we know which predecessors language model scores refer to, in case of a trigram for instance ? Actually, lattices are often generated by a decoding pass using a bigram language model, before being expanded with higher order knowledge sources. The expansion process keeps acoustic scores while replacing language scores and expanding lattice nodes as required by the structure of the language model. Figure 4.5 below illustrates this procedure in case of a tri-gram model expansion.

Normally, the lattice has to expand as it needs to incorporate the dependecies present in the new language model as well as those already present in the original lattice. For instance, arcs are duplicated in order to ensure that each node will have the unique two words history needed to estimate tri-gram likelihoods. In our example, we can observe that the instance of word IT has been duplicated five times for the five tri-gram contexts SIL IT, IN IT, TO IT, AN IT and A IT. By doing so, language model probability for word DIDN'T, which depends on the two previous words, are placed on different lattice arcs, while time stamps and acoustic likelohoods are not modified. Thus, by assuring that for each link the relevant context is unique, we obtain a lattice with a large number

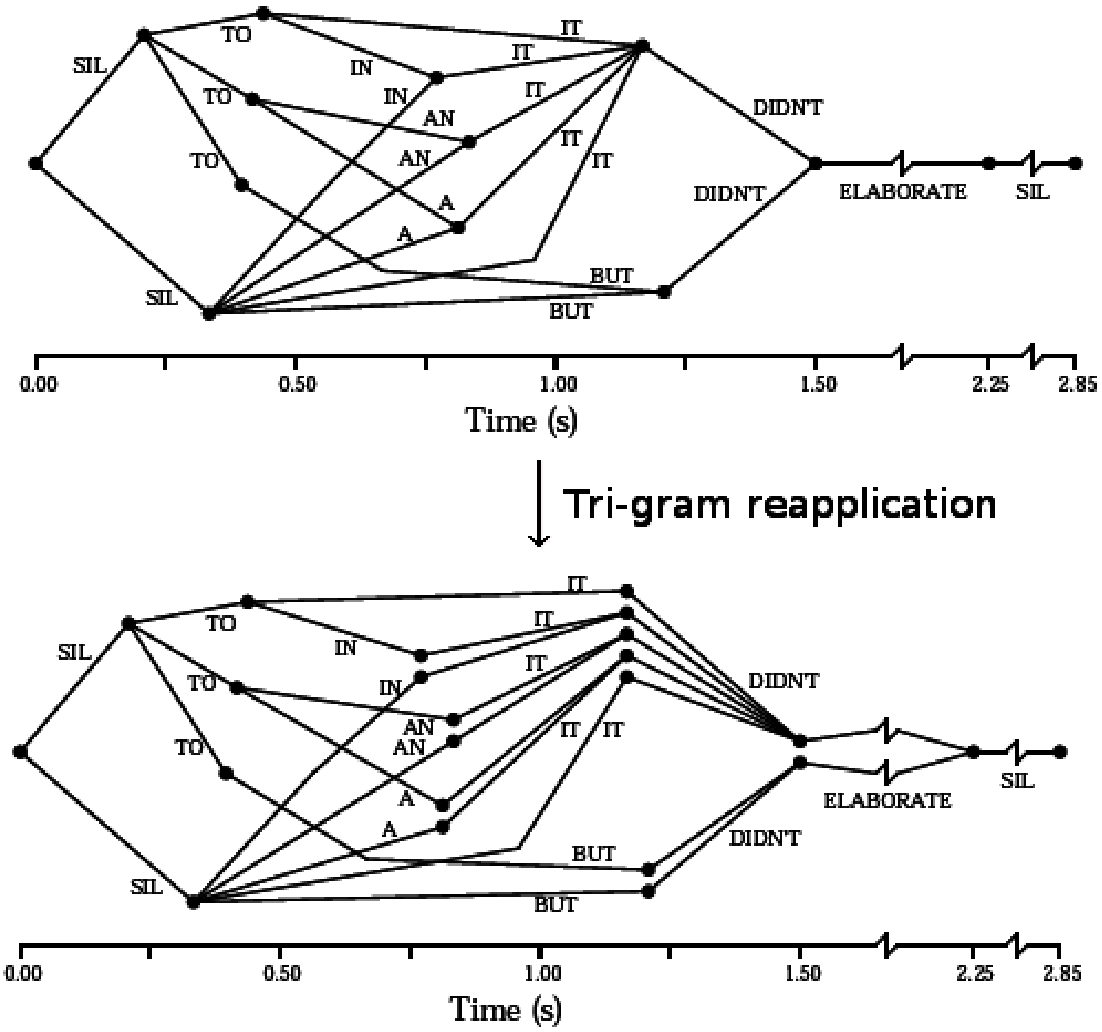

Figure 4.5: Lattice expansion using a tri-gram language model ([34]).

of links labelled with the same word but corresponding to different segmentations and contexts.

4.2.3.2 Generating word lattices : LVCSR system architecture

The classical decoding procedures have to be modified in order to keep in memory more than the best hypothesis. To generate the word latices that will be further processed, we used a multi-pass LVCSR recognizer derived from the AMI speech meeting transcription system[1] ([21], [22]). The modified system was developed by Brno University of Technology in the context of the NIST Spoken Term Detection 2006 workshop, which took place in Gaithersburg (MD, US) on 14-15 December 2006 ([46]). This section intends to briefly expose the recogniser flow-chart, without entering into deep

[1]The AMI project, for Augmented Multi-party Interaction, is a European project that targets to substantially advance the state-of-the-art in computer enhanced multi-modal interaction in the context of meetings (*http://www.amiproject.org*).

details. More specific informations (such as training data, dictionnary size...) will be given in section 4.6 on page 70.

The LVCSR system is based on a HTK (Hidden Markov Model Toolkit for speech recognition, see [53]) architecture. Figure 4.6 on page 56 shows that it is a 3-pass decoder. Each decoding pass uses the Token Passing algorithm ([52] and [53]) to perform Viterbi-based recognition. This algorithm intends to reformulate classical Viterbi-decoding procedure in a process where tokens, which hold log-scores and word boundaries, are passed around a transition network. In LVCSR, as the number of models which are investigated during decoding is quite high (we often reach 50000 words in a LVCSR dictionnary !), strategies which limit the search space need to be imagined to avoid explosion of computational times. The token algorithm allows such a beam search by deactivating all the tokens which scores are below a certain threshold, thus constraining the search space and accelerating the decoding (**beam search** algorithms). Moreover, by playing with the number of passed tokens, multiple hypotheses can be stored under the form of word lattices or N-best lists.

The **first pass** starts from the 39 dimesion PLP acoustic vectors to perform Cepstral Mean Normalization (CMN) and Cepstral Variance Normalization (CVN). CMN is a technique used to offer robustness to convolutional noise induced by the channel. Indeed, the transmission channel transfer function is multiplied with the speech signal spectrum (convolution in time domain, which explains the name convolutionnal noise for such a disturbance). The convolutional effect turns in a summation in the log power domain. Moreover, as the channel spectrum is constant over time, and as we do not care about the constant part of speech, the mean of the total log-spectrum can be removed to supress channel effect. Alternatively, one can compute the FFT of the log-spectrum, yielding cepstral coefficients, and substract the mean in this domain. CVN consists in normalizing the feature variance to 1 to deal with noises and channel mismatch. After these normalizations, a first recognition is conducted on acoustic models trained with the Maximum Likelihood (ML) criterion (see section 2.4.2.3 on page 23), and with application of a trigram language model. Acoustic models are 3-state monophones, extended to context-dependent triphones (so phones including left and right contexts), with a tree clustering of the contexts (to handle unseen contexts). They are trained up to 16 gaussian mixture components (GMMs), on 277 hours CTS data. The first pass outputs a one-best decoding hypothese.

The **second pass** uses the transcription got from the first pass to perform Vocal Tract Normalization (VTLN). Indeed, shape and size of human vocal tracts differ from speaker to speaker (especially between female and male), and VTLN aims to warp the frequency axis to shift the formant to their "canonic" position. In this case, PLP features were used in combination with the first pass best hy-

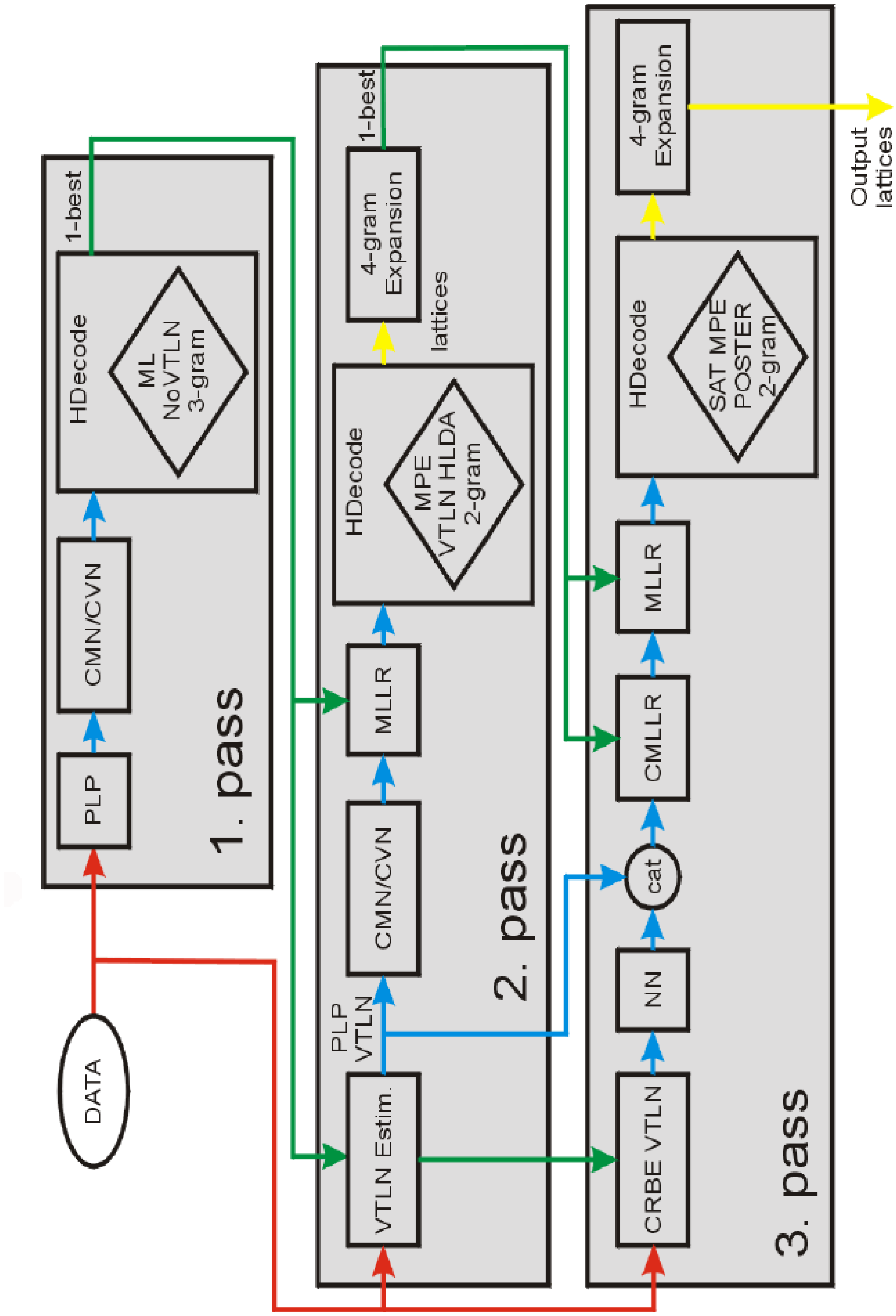

Figure 4.6: Overview of the LVCSR system ([46]).

pothesis to estimate the warping factors, and recode data following them. After a new CMN/CVN pass, a Maximum Likelihood Linear Regression is applied to perform more general speaker and environment adaptation : it uses linear transformations and transcripts got from first pass to adapt the acoustic model means and variances (in order to maximize likelihood of the adaptation data). Note that a Heteroscedastic Linear Discriminant Analysis (HLDA) is also applied using the reference given from the first pass to reduce the feature dimensionality (from 52, standard plus third derivative, to 39), while keeping the maximum of information. Then a second decoding pass is applied using bigram models and acoustic models trained on VTLN data following the Minimum Phone Error criterion (MPE), a discriminative training method which intends to maximize the accuracy at the phone level. The decoding phase generates lattices, which are then expanded with a 4-gram language model.

In the **third and last pass**, critical band VTLN warping coefficients are re-estimated, and are used to recode data. The latter is then passed to a Neural Network (NN) to estimate state posteriors. VTLN PLP created during second pass and VTLN posteriors are after concatenated, then passed through HLDA for dimension reduction. Features are transformed by constrained MLLR (CMLLR, trained per speaker) and by MLLR (global transformation). The decoding phase is then applied with bigram language model on acoustic models trained with a Speaker-Adapted Training which minimizes the phone errors (SAT MPE). The generated lattices are then 4-gram expanded.

References about the concepts briefly exposed above can be found in [21].

4.2.4 Keyword hypotheses generation

4.2.4.1 Keyword posteriors estimation

The LVCSR first pass goal was to prune the initial search space by keeping only the most probable solutions, and storing them under the form of word lattices, allowing us to apply more complex algorithms on them. In this section, we describe how these reduced representations are employed to estimate the keyword hypotheses posterior probabilities $p([KW; t_s, t_e]|X_1^T)$. All the following mathematical developments are based on [50] and [15].

Assuming that $p([w; t_s, t_e]_1^M|X_1^T)$ stands for the posterior probability of having a sequence of word hypotheses $[w; t_s, t_e]_1^M = [w_1; t_{s_1}, t_{e_1}], \ldots, [w_M; t_{s_M}, t_{e_M}]$ ($[w_i; t_{s_i}, t_{e_i}]$ standing for a word hypothesis with word label w_i that begins at time t_{s_i} and ends at time t_{e_i}) and following Bayes rule, we can write :

$$p([w;\, t_s,\, t_e]_1^M | X_1^T) = \frac{p(X_1^T | [w;\, t_s,\, t_e]_1^M) P(w_1^M)}{p(X_1^T)} \tag{4.1}$$

In the above equation, w_1^M stands for the word sequence $w_1, \ldots, w_M$. From now we will suppose that the acoustic observations are independently generated among words of the sequence. In other words, that means that the generation of $x_{t_{sm}}^{t_{em}} = x_{t_{sm}}, \ldots, x_{t_{em}}$ will only depend on the word m. Considering that fact, equation 4.1 can be reformulated as :

$$p([w;\, t_s,\, t_e]_1^M | X_1^T) = \frac{\prod_{m=1}^M [p(x_{t_{sm}}^{t_{em}} | w_m) P(w_m | w_1^{m-1})]}{p(X_1^T)} \tag{4.2}$$

Following it, the posterior probability of a given word (or keyword in our case, refered as KW) hypothesis can be computed by summing on all the word hypotheses sequences $[w;\, t_s,\, t_e]_1^M$ which contain the hypothesis $[w;\, t_s,\, t_e]$:

$$\begin{aligned} p([KW; t_s, t_e] | X_1^T) &= \sum_{\substack{[w;t_s,t_e]_1^M | \\ [KW;t_s,t_e]=[w_i;t_{s_i},t_{e_i}];\, i\in\{1,\ldots,M\}}} p([w;\, t_s,\, t_e]_1^M | X_1^T) \\ &= \sum_{\substack{[w;t_s,t_e]_1^M | \\ [KW;t_s,t_e]=[w_i;t_{s_i},t_{e_i}];\, i\in\{1,\ldots,M\}}} \frac{\prod_{m=1}^M [p(x_{t_{sm}}^{t_{em}} | w_m) P(w_m | w_1^{m-1})]}{p(X_1^T)} \end{aligned} \tag{4.3}$$

In our case, the argument of the summation sign in Equation 4.3 refers to all possible (and relevant) word sequences contained in the restricted search space, namely in the word lattices. Hence, in our word spotting context, the idea will be to "grep" hypotheses with a keyword label into the lattices, thus generating **keyword hypotheses**, and then to associate a score with each of them by **estimating** $p([KW;\, t_s,\, t_e] | X_1^T)$ from the reduced space.

Equation 4.3 gives a direct way to estimate the posteriors from the lattices : $p(x_{t_{sm}}^{t_{em}} | w_m)$, the acoustic likelihoods, are part from the lattice link weights. Moreover, if the lattices have been expanded (following a 4-gram model in our case), their language model scores can stand for $P(w_m | w_1^{m-1}) \approx P(w_m | w_{m-3}^{m-1})$. However, to speed up the computational procedure, a forward-backward algorithm is usually applied, except that it is now performed at the word level instead of the state level (as for HMMs likelihood estimations, section 2.4.2 on page 17). It requires the definition of two variables, the forward probability $\alpha([KW;\, t_s,\, t_e])$ and the backward probability $\beta([KW;\, t_s,\, t_e])$. The former holds the probability that a sequence of word hypotheses, among all the sequences allowed by the lattice topology, will end with $[KW;\, t_s,\, t_e]$ while the latter stores the probabilty that a word sequence (with the same restriction as above) will begin with $[KW;\, t_s,\, t_e]$. The two variables can be computed recursively as follows :

$$
\begin{aligned}
\alpha(link) &= A_{score}^{link} L_{score}^{link} \sum_{all\ possible\ previous\ links} \alpha(previous\ link) \\
\beta(link) &= A_{score}^{link} L_{score}^{link} \sum_{all\ possible\ next\ links} \beta(next\ link)
\end{aligned} \tag{4.4}
$$

In equations 4.4, A_{score}^{link} and L_{score}^{link} stand respectively for the link acoustic likelihood $p(x_{t_{s_{link}}}^{t_{e_{link}}}|w_{link})$ and the link language model score. Thus, α can be computed chronologically in an ascendant order while β in a descendant order. Then, following the definitions of α and β and equation 4.3, the desired posterior probability is given by :

$$
p([KW;\, t_s,\, t_e]|X_1^T) = \frac{\alpha([KW;\, t_s,\, t_e])\beta([KW;\, t_s,\, t_e])}{A_{score}^{[KW;\, t_s,\, t_e]} L_{score}^{[KW;\, t_s,\, t_e]} p(X_1^T)} \tag{4.5}
$$

$A_{score}^{[KW;\, t_s,\, t_e]}$ and $L_{score}^{[KW;\, t_s,\, t_e]}$ appeared in the denominator of 4.5 for algorithmic reasons (because they appeared twice in Equations 4.4). $p(X_1^T)$ can be approximated by summing over all possible pathes through the lattice, and using the forward and backward variables, we have :

$$
\begin{aligned}
p(X_1^T) &= \sum_{w}\sum_{t_s} \alpha([w; t_s, T]) \\
&= \sum_{w}\sum_{t_e} \beta([w; 1, t_e])
\end{aligned} \tag{4.6}
$$

An interessant property of our link posteriors (equation 4.5) should now be established. The posterior probabilities of all parallel links that contain a specific point in time t always sum up to 1 :

$$
\sum_{\substack{[w;\, t_s',\, t_e'] \,|\\ t_s' \le t \le t_e'}} p([w;\, t_s',\, t_e']|X_1^T) = 1 \qquad \forall t \in \{1, \ldots, T\} \tag{4.7}
$$

In other words, if we imagine to draw a vertical line on the lattice at a specific time, the total intersected probability will sum up to 1. This behavior will be used in next section for a further processing of hypotheses.

4.2.4.2 Keyword hypotheses processing

Previous equations gave us a way to compute a score for each keyword hypothesis grepped in the lattice[2] under the form of a posterior probability $p([KW; t_s, t_e]|X_1^T)$. But these scores cannot be used directly for a comparison with a threshold. Indeed, the question is : how could we deal with overlapping keyword hypotheses (with same keyword index obviously) ?

The keyword hypotheses generation process can thus be decomposed in two steps, as figure 4.7 shows. First, all the keyword links are "grepped" in the lattice and their posteriors are computed following the described procedure. Then, the overlapping keyword links are processed and rescored following various criteria in order to generate the final keyword hypotheses. The end of this section will adress the problem of what criterion choosing to process the overlapping links.

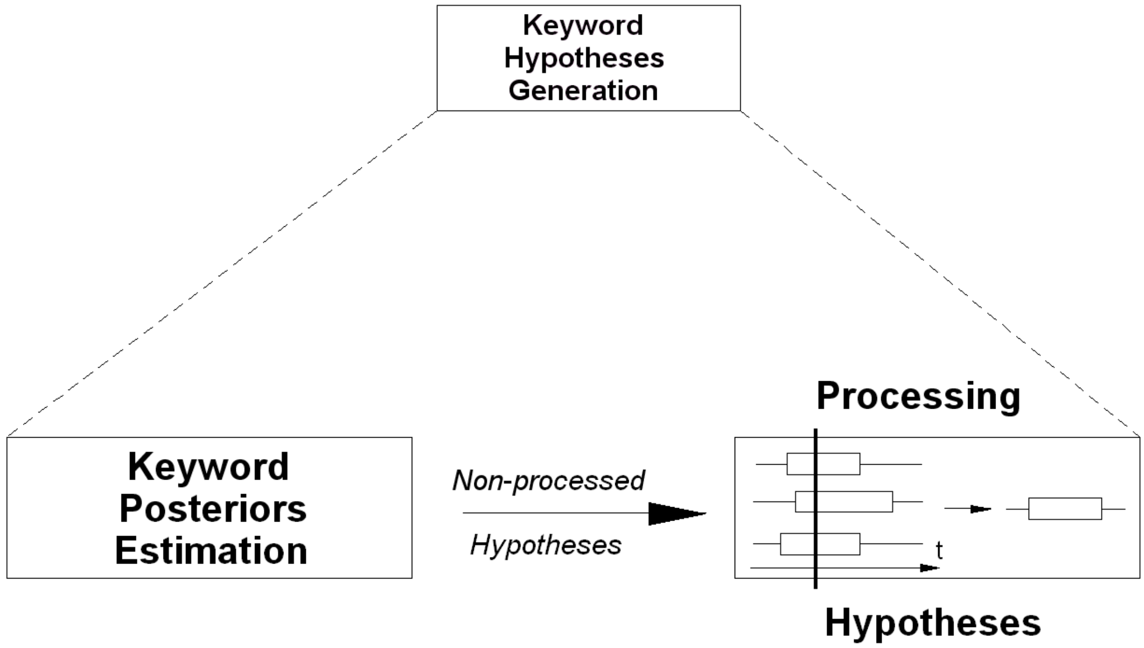

Figure 4.7: Generation of keyword hypotheses. Posteriors are first computed for each keyword hypothesis, then overlapping hypotheses are merged and rescored following different criteria.

The first solution that comes at mind consists in only keeping the one with the highest posterior probability, before comparing with a threshold to make the decision (see Figure 4.8 below). This approach will be later on refered as $\mathcal{S}_{max}$.

[2]Keyword hypothese posteriors were computed during this work thanks to the SRI Language Modeling Toolkit ([44]), and calculations were applied in the log domain.

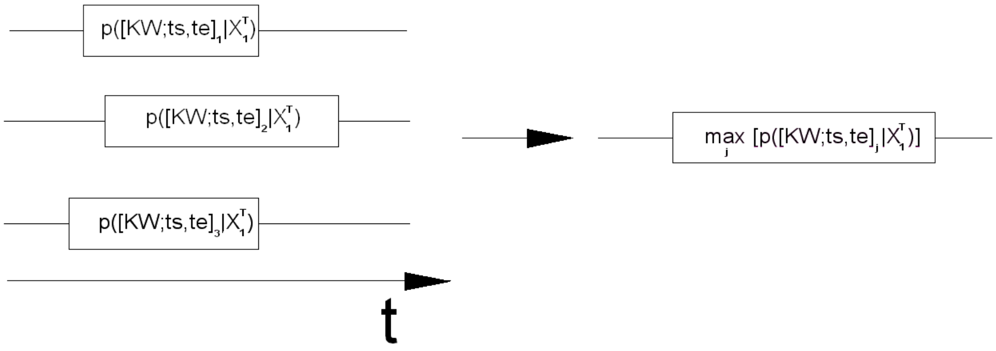

Figure 4.8: One simple way to deal with overlapping keyword hypotheses.

However, we could foresee that such a method will not be sufficient. Indeed, following the property exposed in equation 4.7, the intersected posteriors at a specific time point in the lattice must always sum up to 1. Consequently, the posterior probability mass is splitted among the overlapping hypotheses, leading us to waste some weight by only taking the maximum. As hypotheses with slightly different starting and ending times often correspond to the same word, a smarter approach would be to **accumulate** the posteriors among the overlapping hypotheses.

In the following, three more criteria are investigated, based on [50]. The first one proceeds as follows. A keyword hypothesis grepped from the lattice is rescored by summing the posteriors of all hypotheses for which intersection of time interval is not empty. This criterion will be later on refered as :

$$\mathcal{S}_{acc}([KW;\, t_s,\, t_e]) = \sum_{\substack{[KW;\, t'_s,\, t'_e]| \\ [t_s\ t_e]\cap[t'_s\ t'_e]\neq\emptyset}} p([KW;\, ,t'_s,\, t'_e]|X_1^T) \tag{4.8}$$

Once each hypothese is accumulated, we choose among the overlapping hypotheses the one with the highest score $\mathcal{S}_{acc}$. If ϑ is the set of hypotheses with same keyword index and which overlap, we have :

$$Final\ hypothesis = \underset{[KW;\, t'_s,\, t'_e]\in\vartheta}{\text{argmax}}\ \mathcal{S}_{acc}([KW;\, t'_s,\, t'_e]) \tag{4.9}$$

However, it is straightforward to understand that accumulating the posteriors over all the overlapping hypotheses could lead to a score greater than 1. To avoid this missing normalization, a second criterion has been investigated :

$$\mathcal{S}_{med.acc.}([KW;\, t_s,\, t_e]) = \sum_{\substack{[KW;\, t'_s,\, t'_e]|\\ t'_s \leq t_s + \frac{t_e}{2} \leq t'_e}} p([KW;\, ,t'_s,\, t'_e]|X_1^T) \tag{4.10}$$

The second criterion $\mathcal{S}_{med.acc}$ restricts the accumulation of posteriors to one single frame, in this case the hypothesis medium frame t_{med}. Again, $\mathcal{S}_{med.acc}$ is computed for each keyword hypothesis, and then we keep the one with the highest score, as explained in Equation 4.9 but by replacing $\mathcal{S}_{acc}$ by $\mathcal{S}_{med.acc}$.

One may wonder now how the choice of the accumulating frame (t_{med} above) could affect the results. This lead us to define the third and last criterion :

$$\mathcal{S}_{max.acc}([KW;\, t_s,\, t_e]) = \max_{t_{max} \in [t_s\; t_e]} \sum_{\substack{[KW;\, t'_s,\, t'_e]|\\ t'_s \leq t_{max} \leq t'_e}} p([KW;\, ,t'_s,\, t'_e]|X_1^T) \tag{4.11}$$

The last criterion just accumulates the posteriors for each keyword hypothesis frame, and then chooses the highest score. The three last exposed criteria introduce one supplementary step in the overall keyword hypotheses generation process, the **posterior rescoring** step, as can be seen on Figure 4.9 below.

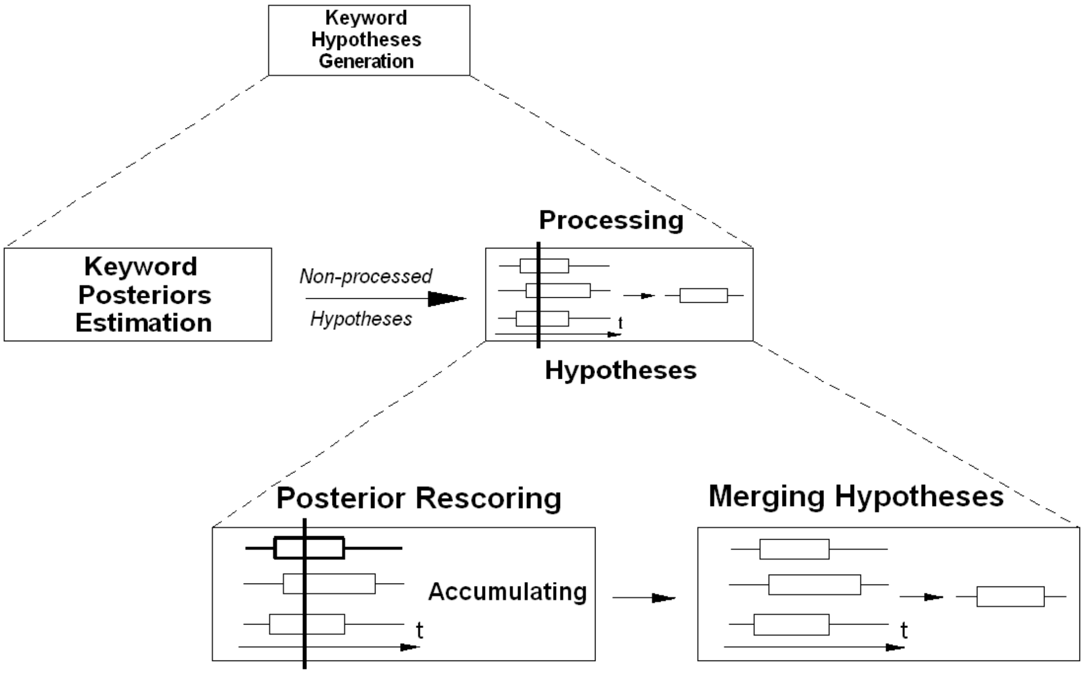

Figure 4.9: Complete process of keyword hypotheses generation.

4.3 Hybrid HMM/ANN lattice rescoring

The difference between "calculation" and "estimation" may *a priori* appear obscure to students like us. During our work, and our numerous engaged discussions with scientists, we argued more than one time about the feckless use of the term "probability calculation". Indeed, all we can do in statistical sciences is **estimating** probabilities, basing on some more or less restrictive hypotheses. Thus, one smart approach (among others...) for a researcher who wants to improve a technology would be to list all the baseline hypotheses, and then choose to study one of them.

We will not review and study all the hypotheses that have been posed since the beginning of this report. In this work, we will only adress one of them.

In chapter 2, we saw that HMMs, in spite of their qualities to estimate acoustic scores, implied a lot of constraints. Among them, we said that they required previous assumptions on the state probability density function shape. The most employed is the gaussian (or multigaussian) assumption, for the simplicity offered when developing training algorithms. However, we cannot say that speech true distribution follows a gaussian. Moreover, HMMs work by assuming that acoustic vectors are not correlated in time (see Equation 2.14), which is obviously far from reality !

One possible solution to these constraints could be the use of hybrid HMM/ANN models. ANN are indeed trained to approach data distribution without any assumption on their shape. The second cited HMM restriction could moreover be softened if we consider the fact that ANN are able to handle context information by taking multiple acoustic vectors as input.

Following these thoughts, we propose to re-estimate the lattice acoustic scores, namely to change likelihood **functional approximation**, coming initially from GMMs (see section 4.2.3.2), by applying a forced Viterbi alignement on hybrid models. The procedure can be observed on figure 4.10.

First, relevant data is selected using the original lattice link time stamps. The state posteriors $P(q_k|x_n)$ coming from the MLP (see figure 4.12) are then divided by the priors $P(q_k)$ to obtain scaled likelihoods $\frac{p(x_n|q_k)}{p(x_n)}$ following Bayes rule as explained in section 2.4.2.5 on page 26 :

$$\frac{P(q_k|x_n)}{P(q_k)} = \frac{p(x_n|q_k)}{p(x_n)} \tag{4.12}$$

These scaled likelihoods play afterwards the role of emission probabilities on (phoneme tri-state) HMM models generated using word phonetic transcripts, during a Viterbi alignment (see equations

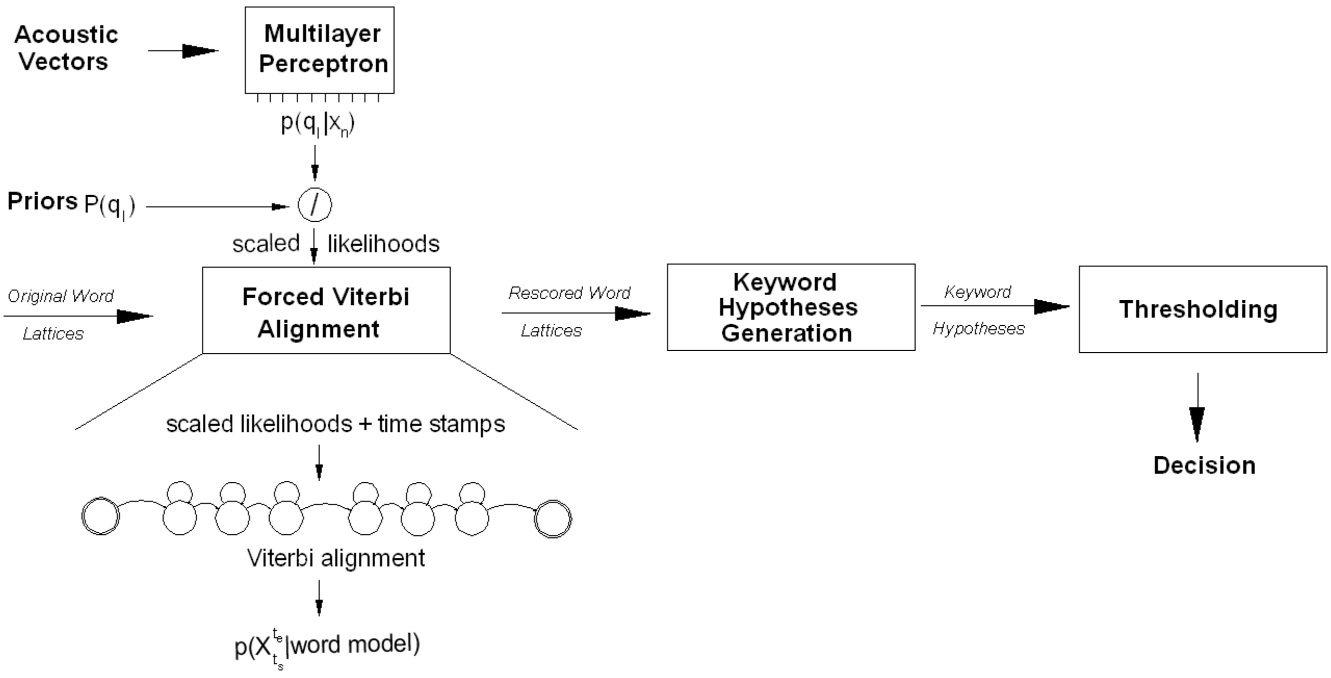

Figure 4.10: Rescoring lattices with hybrid HMM/ANN models.

2.18 on page 22) which yields the desired word acoustic score[3]. Finally, each lattice link acoustic liklelihood is replaced with the new estimation. Rescored lattices are then ready to be spotted.

To summarize, our word spotting algorithm described in section 4.2.4 was applied on lattices generated by GMM LVCSR system, and on the same lattices rescored using hybrid models following the above procedure (see figure 4.11 below).

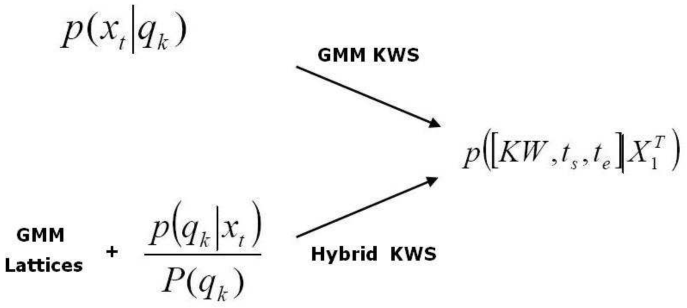

Figure 4.11: On one hand, keyword hypothesis posteriors are estimated from GMM LVCSR lattices (where $p(x_n|q_k)$ are the state GMM emission probabilities). They can also be estimated from hybrid rescored lattices (where $p(q_k|x_n)$ stands for the state posterior probabilities coming from the MLP and $P(q_k)$ stands for the prior probabilities).

[3]All these tasks have been brought to fruition by using the *Dr. Speech* software (http://www.drspeech.com).

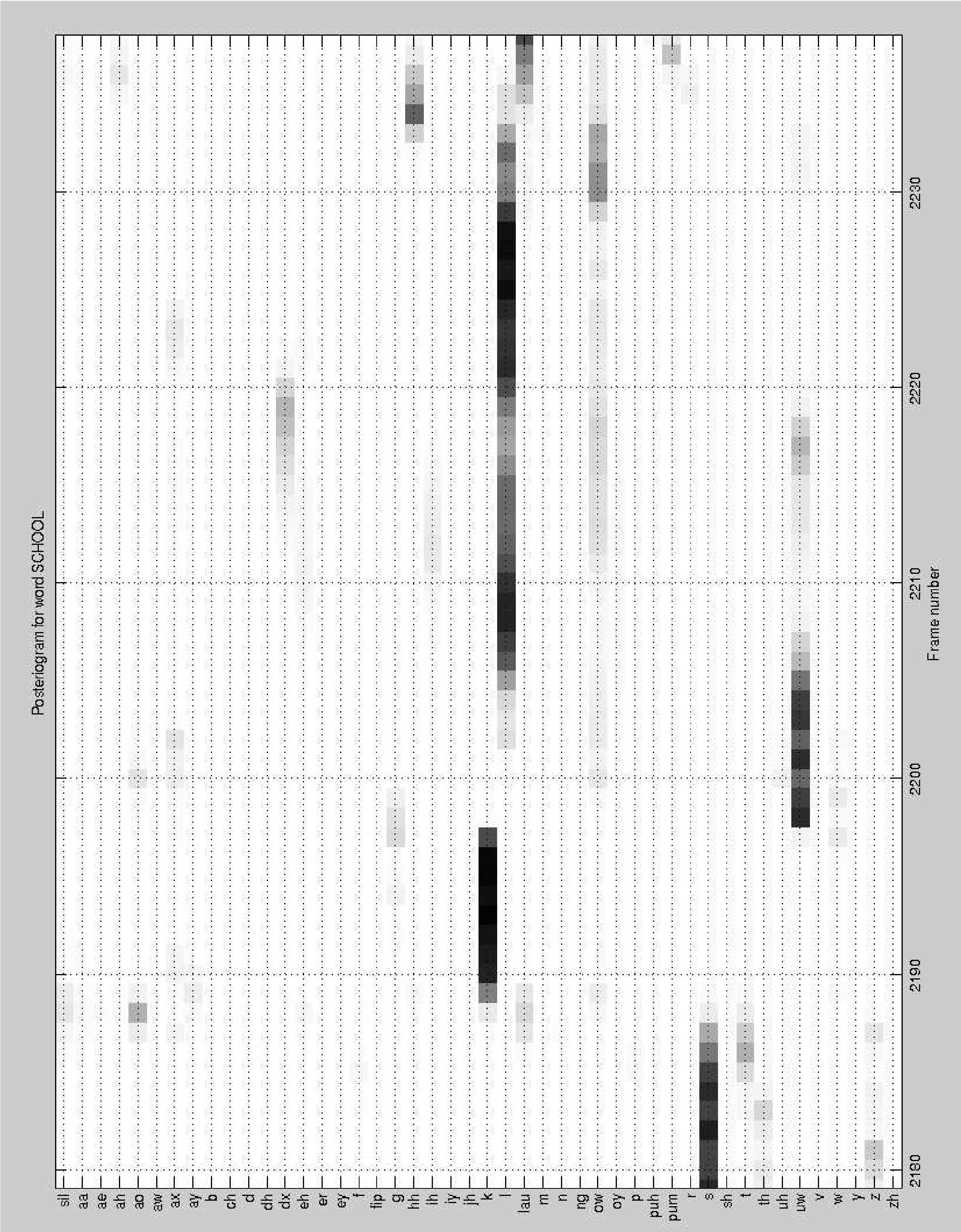

Figure 4.12: Posteriogram for word "school" (PLP based MLP, see section 4.5.2 later on). Gray levels attest for state posterior values : darker areas refer to higher phoneme posteriors.

4.4 About lattices generation

At this point, some words need to be said considering the word lattice generation. Indeed, they have to be as faithful as possible to the initial search space. In other words, a word lattice has to contain all the important hypotheses while remaining compact. Consequently, a question that naturally came into my mind was : how to figure out the lattice richness ?

To that end, we followed this simple line. We said during previous section (4.2.4 on page 57) that processed keyword hypothesis scores were compared with a threshold in order to take the final decision. Thus, we immediately understand that different operating points (which will be employed to draw ROC curves) can be obtained by varying the threshold value between 0 and 1. More precisely, when the threshold will be set to 0, all the lattice keyword hypotheses will be accepted as keywords. The last consideration gives us a way to estimate lattice accuracy. Indeed, if we run the keyword spotter with a 0 threshold, and without any keyword hypotheses processing, dividing the number of correct keyword detections by the number of keyword occurences present in reference transcripts will give a percentage that could be used as a measure of lattice richness for the spotting task.

So, the keyword spotter was run on lattices derived from CTS data given within the NIST's Spoken Term Detection evaluation framework (2006), with a 0 threshold. The keyword list had a size of 625, and contained short words as well as longer terms. As mentionned above, we counted the number of "correct" detections for all the keywords and the number of keywords occurences in reference data, and obtained a value of 97.28%. This high value provides evidence that the lattices generated with the LVCSR system cover nearly all the search space, and thus can be used for further keyword spotting experiments.

Another question could rise in our minds at this step. We exposed during previous section a way to change acoustic likelihoods functional approximation (see section 4.5.2 on page 68), by simply replacing lattice acoustic scores with hybrid HMM/ANN estimation. One could have noticed that, to be completely right, the lattices would need to be generated thanks to the hybrid system itself, leading to redesign an entire hybrid LVCSR recognizer. But we can avoid that long process[4] if the original lattices (or **GMM lattices**) contain all (or nearly all) the search space hypotheses. Basing on the previous paragraph result, we will admit this assumption, and new acoustic scores will simply replace the older ones in original lattices, creating rescored lattices later on refered as **hybrid rescored lattices**.

[4] Actually, it would have been really interesting to do it, but time always defaults when you have ideas…

4.5 Scaling factors tuning

In practise, as dynamic ranges of acoustic and language model scores are very different, scaling factors are applied to obtain satisfactory results. Let's consider the acoustic scores first. They are often estimated via a Viterbi approximation, by accumulating state emission probabilities and transition probabilities among the HMM topology. This procedure involves that longer words, thus with more states, will have bigger scores than short words. Likelihoods are not normalized. However, it is not true for language model scores : the way N-gram are estimated naturally includes normalization.

Moreover, during keyword posterior estimations, both recognizer acoustic and language model scores are combined, as we saw during section 4.2.4. Following [32] and [50], scaling factors are also indispensable in this case : if scores are not scaled appropriately, the summations in all of the equations presented in section 4.2.4.1 are dominated by a few word links because of the large dynamic range of the acoustic scores. Consequently, two scaling factor sets need to be tuned in our case : one for lattice generation (during HTK LVCSR passes), and another for posterior estimations (using the SRILM toolkit).

We will first adress the case of posterior estimation. The scaling procedure is usually accomplished by multiplying all the acoustic and language log-scores respectively by α and β. Basing myself on experiments performed in [32] and [50], we found that the best scaling constant (within a word error rate minimization framework) is equal to unity for language model scores. This is exactly what we would expect because language model scores are normalized as we said above. These experiments also showed that acoustic log-scores need to be scaled by the inverse of the LVCSR language model weight in order to obtain the best performances.

Consequently, for the purpose of link posterior estimations, lattice language model log-scores are not modified ($\beta = 1$), while acoustic log-scores are multiplied by $\alpha = \frac{1}{\lambda}$, where λ stands for the LVCSR language model weight.

However, one could ask for the value of λ. In the following, we will analyse this issue, beginning with the case of the original lattices, coming from the LVCSR system (GMM lattices). Secondly, we will see that it will need to be re-tuned if we want to spot the hybrid rescored lattices, as acoustic scores are re-estimated following the procedure exposed in section 4.3.

4.5.1 GMM lattices

Scaling factor λ was tuned in order to minimize the WER on the GMM lattices (by extracting the best sentence hypothesis from each lattice) generated from 3 hours of CTS data given by NIST STD (same data used in section 4.6 on page 70). We can see on figure 4.13 that the curve reach a minimum at 24.86% for a value of 29, value that we will keep to run the spotting experiments later on.

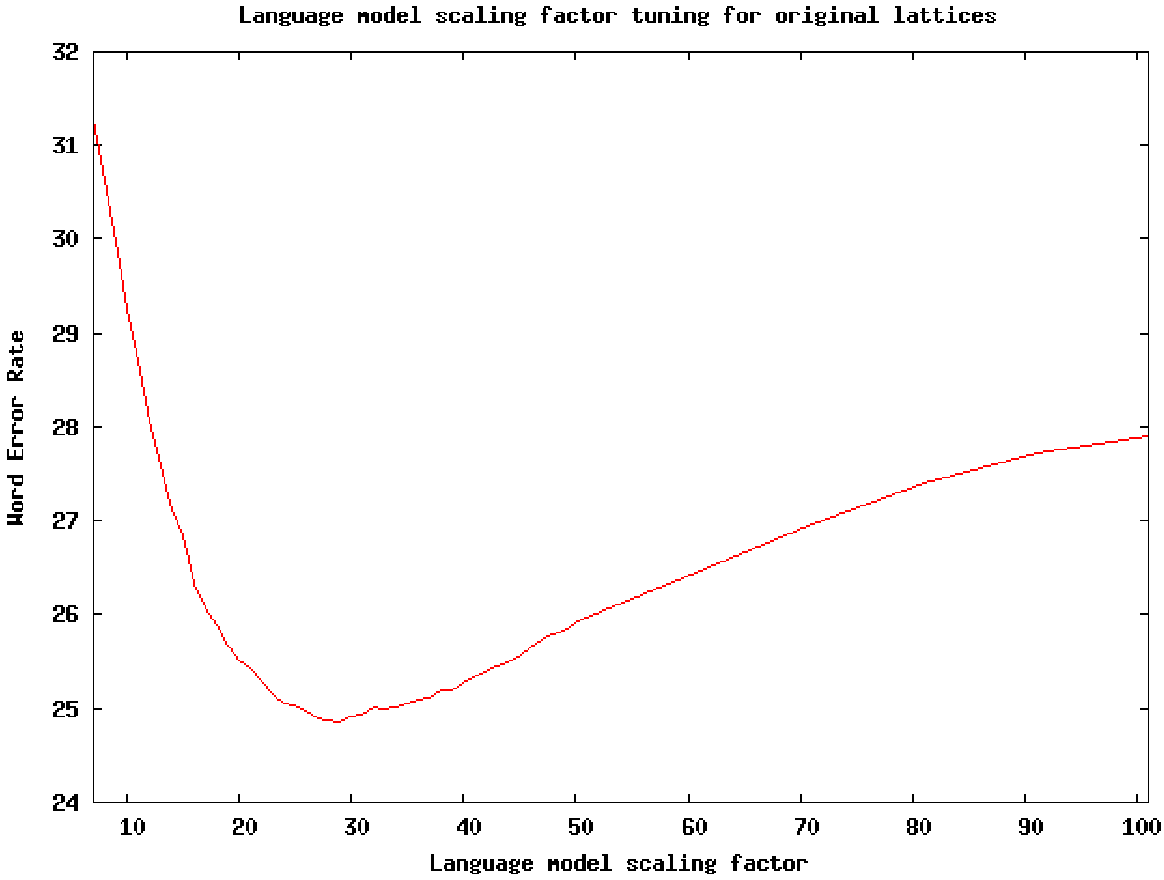

Figure 4.13: Language model scaling factor tuning for GMM lattices.

4.5.2 Hybrid HMM/ANN rescored lattices

For hybrid rescored lattices, original link acoustic scores were replaced by an estimation through a forced Viterbi alignment on HMM/ANN models. Consequently, the optimal λ value will differ from the one found above for GMM lattices as acoustic score ranges are different.

To generate rescored lattices, two available MLPs were used for a comparison purpose. The first one, obtained from the International Computer Science Institute in Berkeley (ICSI), was trained on 2000 hours of CTS fishers data with VTLN. It took as input 9 39-dimensions PLP acoustic vectors (12 standard plus energy, Δ, ΔΔ, for an input layer of 351 units) to yield 46 state posterior probabilities. The size of the hidden layer was 20800. The second one, developed at IDIAP, was only trained on 30 hours of CTS fishers data, but used 448 multi-RASTA (see [16]) acoustic vectors as input. Its hidden layer contained 2000 units, whereas 46 for the output layer. The non-linearity function employed was the softmax function.

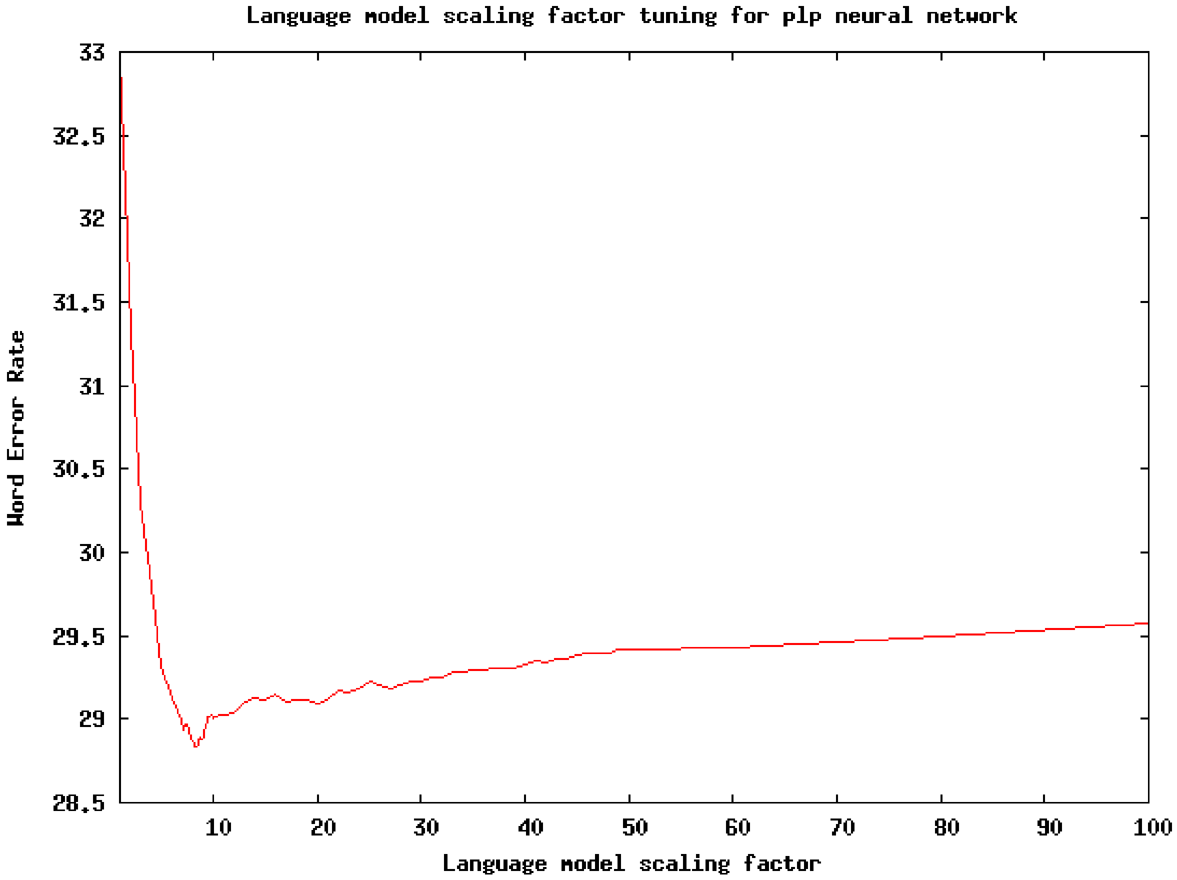

Figure 4.14: Language model scaling factor tuning for hybrid rescored lattices, for ICSI MLP

Again, λ was tuned on the rescored lattices to minimize WER on the NIST STD CTS data, to obtain the two figures 4.14 and 4.15. As we could have expected, the first MLP gives better results (28.84% for $\lambda = 8.3$) than the second (29.38% for $\lambda = 24$) : the high amount of data used for training has obviously something to do with it. Consequently, the lattices rescored thanks to the ICSI neural network will be further used for the word spotting experiments.

We will end this section with the following comment. A further look at the three figures also demonstrates that hybrid systems are less sensitive to scaling factors than GMM based. In other words, hybrid curves are flatter than the GMM one on a large range of scaling factor values. This

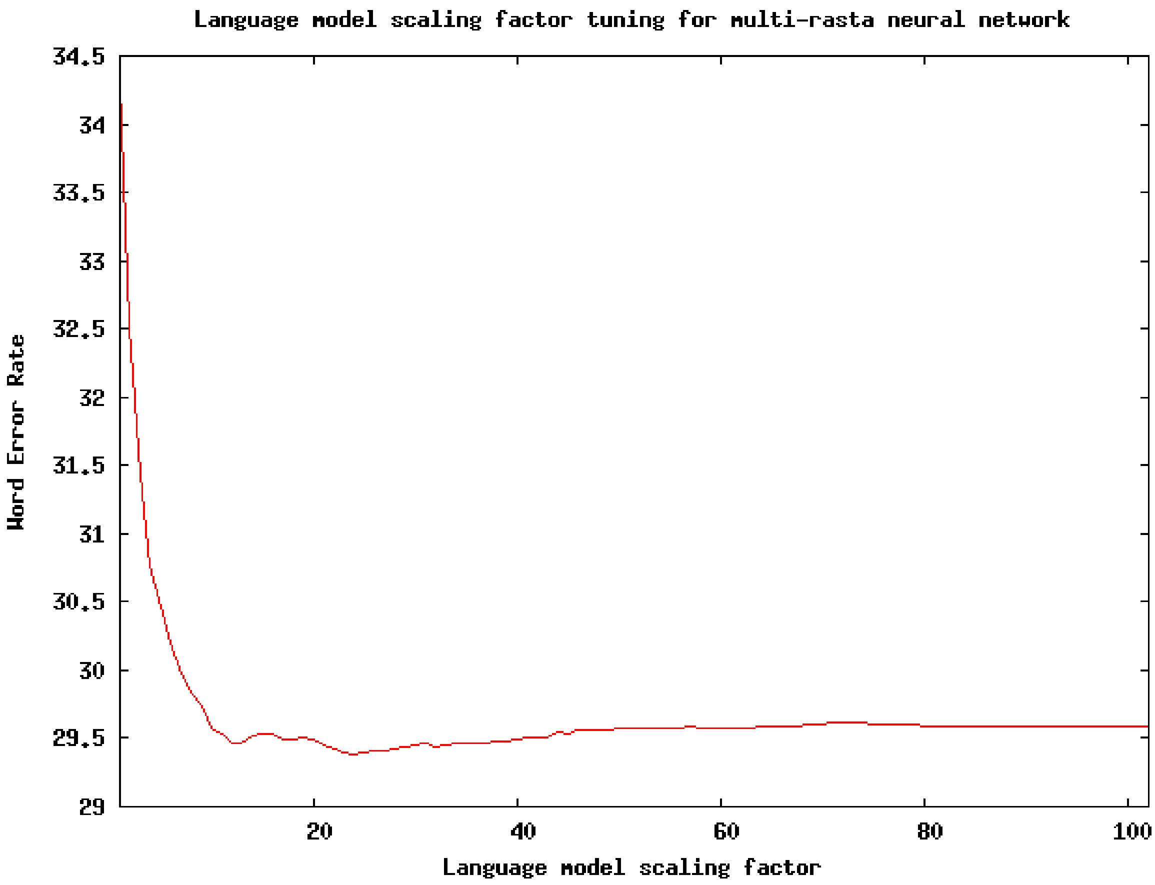

Figure 4.15: Language model scaling factor tuning for hybrid rescored lattices, for IDIAP MLP

could easily be explained. Indeed, acoustic scores on rescored lattices have been estimated by integrating **scaled** likelihoods along word HMM topology, and thus are less affected with the missing normalization than GMM lattices.

4.6 Experiments

4.6.1 System and database characteristics

We will begin this section by listing the multi-pass LVCSR system characteristics. The system acoustic training was performed on the *ctstrain04* training set defined at the Cambridge University as a training set for Conversation Telephone Speech (CTS) recognition systems. It contains about 278 hours of well transcribed speech data from Switchboard *I*, *II* and Call Home English. Language models were trained by interpolation from : Switchboard *I*, *II* + Call Home English (3.5 *Mwords*), Hub4 (Broadcast news, 220 *Mwords*) transcriptions, Fisher (10.5 *Mwords*) data, Web data, BBC (33 *Mwords*), SDR99 (39 *Mwords*), Enron Email (152 *Mwords*) and ICSI/ISL/NIST/AMI

(1.5 *Mwords*) data, for a total amount of more or less 1.5 *Gwords*. Standard 39 dimension PLP feature vectors were used (13 PLP coefficient including energy, extended with their first and second derivative). A classic 3-state left-to-right phoneme setup (44 base phonemes + 2 silences), with 16 gaussian mixtures per state, was used, and then expanded to context-dependent phonemes (triphones) by decision tree clustering. The dictionnary contained 50 *Kwords*s. Bi-gram lattices were generated on 3 hours of NIST STD English CTS data (2006) by keeping 48 tokens per state, before being expanded with 4-gram language models.

To rescore the lattices with hybrid models, an ICSI MLP, trained on 2000 hours of CTS Fishers data with VTLN, was employed. It took as input 9 39 dimension PLP acoustic vectors (12 standard plus energy, Δ, $\Delta\Delta$, for an input layer of 351 units) to yield 46 phoneme posterior probabilities. The hidden layer contained 20800 units. Forced Viterbi alignment was then applied on 3-state left-to-right HMM models.

4.6.2 Performance measures

Keyword spotting experiments were performed on expanded lattices, using the 4 posterior rescoring methods ($\mathcal{S}_{max}$, $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc}$) exposed during section 4.2.4.2 before. The results relative to LVCSR lattices will be refered as **GMM lattices** while those associated to hybrid rescored lattices will be labeled **hybrid rescored lattices** in the following. The keywords, picked in the NIST STD 2006 term list, were separated in 3 categories based on their number of syllables —short (103 keywords, 3591 occurences, *e.g.* "too"), medium (159 keywords, 2014 occurences, *e.g.* "pretty") and long (73 keywords, 496 occurences, *e.g.* "something")— to analyse the length impact on results.

For each case, averaged (on all the terms of the considered list) ROC curves will be plotted by varying the threshold value. The y-axis will represent the probability of true detection while the x-axis will display the number of false alarms per keyword per hour of speech ($FA/kw/h$). Such curves will allow us to observe the trade-off between true detections and false alarms.

Ideally, a word spotting system aims to maximize the detection probability at the lowest false alarm rate. Consequently, we are particularly interested in the ROC curve values below 10 $FA/kw/h$, and the curve values, from 1 to 10 $FA/kw/h$, will be averaged and used as a figure of merit (FOM) in each case.

4.6.3 Comparison with On-Line Garbage Modeling

Our word spotter was compared with an On-Line Garbage Modeling based system, developed at IDIAP for the NIST STD 2006 evaluation. This system used the same ICSI MLP we employed to rescore our lattices (PLP based, trained on 2000 hours of CTS Fishers data, 46 phoneme set). For each keyword list, the value of N (*i.e.* the number of local phoneme posteriors which were averaged at each time frame to get garbage local score) was tuned in order to get the best results, and various operating points were obtained by varying keyword entrance penalties.

4.7 Results and discussions

The two first tables (table 4.1 and 4.2) show FOMs respectively for the short keyword list and for the medium keyword list. In the two cases, results for the four posterior rescoring method we investigated are reported ($\mathcal{S}_{max}$, $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$). To make things clear, we will just remind that $\mathcal{S}_{max}$ simply keeps the best keyword hypothesis (the one with highest word posterior) when they overlap, while $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$ **accumulates** posteriors among parallel overlapping hypotheses, following slightly different criteria. Results are also reported for lattices generated from GMM emission probabilities (*GMM lattices*), and for the same lattices rescored using hybrid HMM/ANN models (*hybrid rescored lattices*).

Averaged Figure of Merit for short keywords				
FOM	$\mathcal{S}_{max}$	$\mathcal{S}_{acc}$	$\mathcal{S}_{med.acc.}$	$\mathcal{S}_{max.acc.}$
GMM lattices	80.4%	82.3%	82.3%	82.3%
Hybrid rescored lattices	68.7%	76.2%	76.7%	76.7%

Table 4.1: Averaged Figure of Merit for short keyword list, and for the four posterior rescoring methods we employed. $\mathcal{S}_{max}$ simply keeps the best keyword hypothesis (the one with highest word posterior) when they overlap, while $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$ accumulate posteriors among parallel overlapping hypotheses, following slightly different criteria. Results are also reported for lattices generated from GMM emission probabilities (GMM lattices), and for the same lattices rescored using hybrid HMM/ANN models (hybrid rescored lattices).

The first conclusion we can draw by observing these tables is the necessity to accumulate keyword hypotheses posteriors among time overlapping keyword hypotheses (with the same keyword label) : we obtain 2% and 8% more FOM with short keywords, for GMM lattices and hybrid rescored lattices respectively, when we compare the values for $\mathcal{S}_{max}$ (without posterior accumulation) and the three other criteria (with accumulation). Besides, the system working on hybrid lattices seems to be more sensitive to that posterior accumulation. However, this FOM improvment decreases with the keyword length (0.2% for GMM lattices and hybrid rescored lattices, for medium keywords).

Averaged Figure of Merit for medium keywords				
FOM	$\mathcal{S}_{max}$	$\mathcal{S}_{acc}$	$\mathcal{S}_{med.acc.}$	$\mathcal{S}_{max.acc.}$
GMM lattices	91.3%	91.5%	91.5%	91.5%
Hybrid rescored lattices	89.6%	89.8%	89.8%	89.8%

Table 4.2: Averaged Figure of Merit for medium keyword list, for the two type of lattices (GMM lattices and hybrid rescored lattices), and for the four posterior rescoring methods we employed ($\mathcal{S}_{max}$, $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$).

Figures 4.16 and 4.17 show the ROC curves for short and medium keywords respectively, for GMM lattices and the four posterior rescoring methods. From them, we can see again that accumulating posteriors yields better scores, as the areas beneath the corresponding ROC curves are higher than the one beneath the curve for $\mathcal{S}_{max}$. However, results do not differ significantly among the three "accumulating" criteria $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$. Consequently, for further tables, we have only kept values for the $\mathcal{S}_{max.acc.}$ criterion, as it mostly gives the best FOMs.

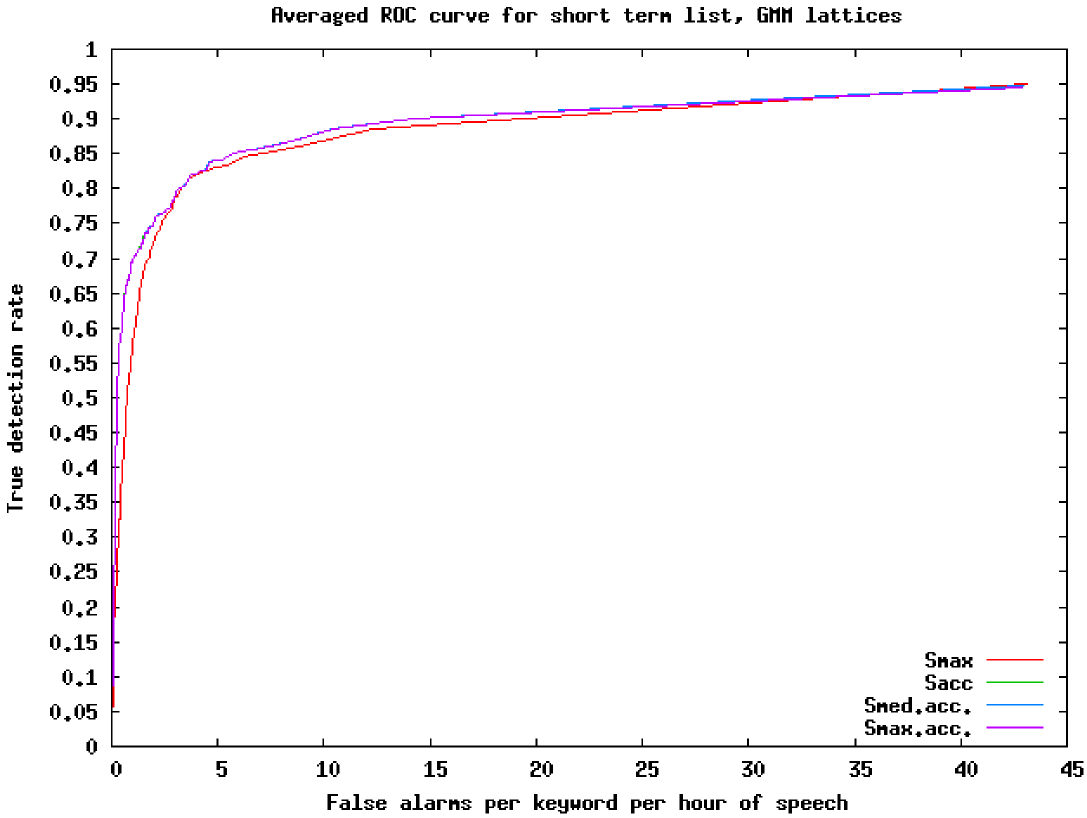

Figure 4.16: ROC curve for short keyword list, using GMM lattices, for the four posterior rescoring methods.

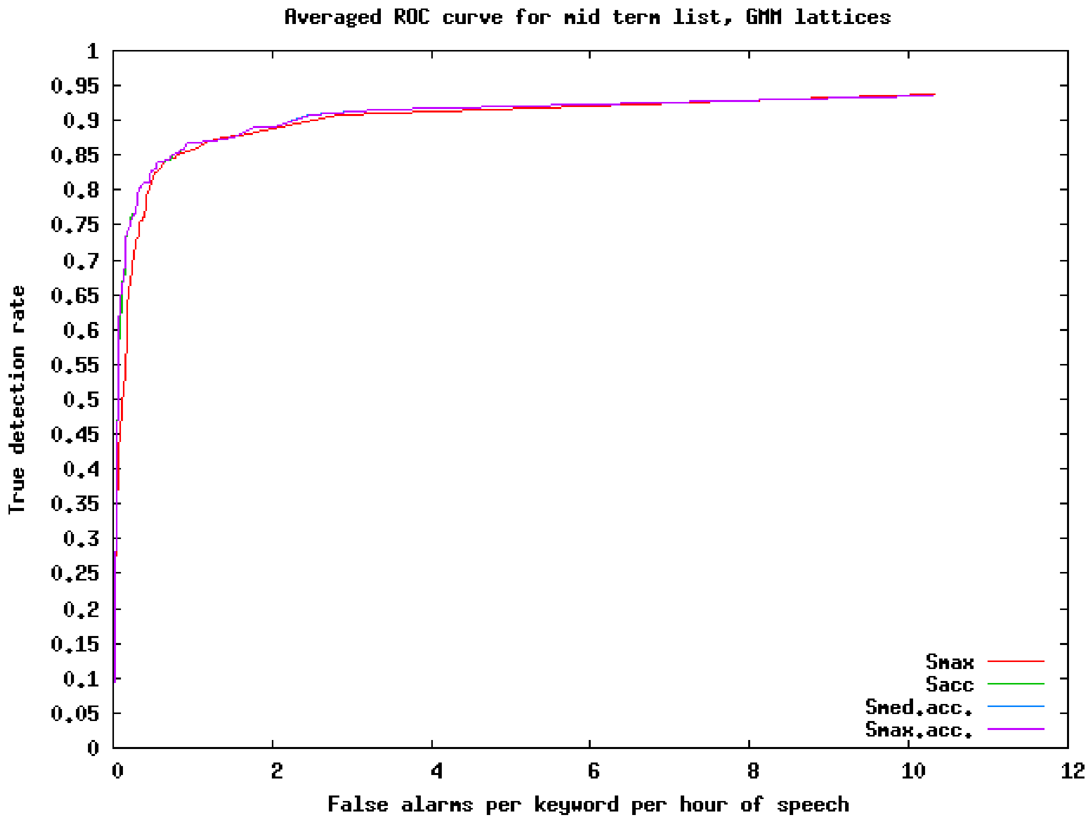

Figure 4.17: ROC curve for medium keyword list, using GMM lattices, for the four posterior rescoring methods.

From the two previous tables, we can also observe that running our word spotting system on hybrid rescored lattices seems to yield worse results than GMM lattices, and that this difference tends to decrease with the keyword length. At this level, it is difficult to draw a general conclusion, and explaining that fact will certainly require further investigation. The following ROC curve (figure 4.18) illustrates these considerations when searching for long keywords.

For the long keyword list, we were not able to compute FOMs. Indeed, we defined FOM as the average of true detection rates from 1 to 10 $FA/keyword/hour$. As figure 4.18 shows, we did not reach more than 1.6 $FA/keyword/hour$ by varying our threshold, and that is why we reported the true detection rate at 1 $FA/keyword/hour$ instead of FOM for long keywords. However, we cannot put as much confidence in this measure as in the FOMs : following table 4.3, the system using hybrid rescored lattices would work better than the other for long keywords, while it is obviously not the case if we observe the corresponding ROC curve on figure 4.18.

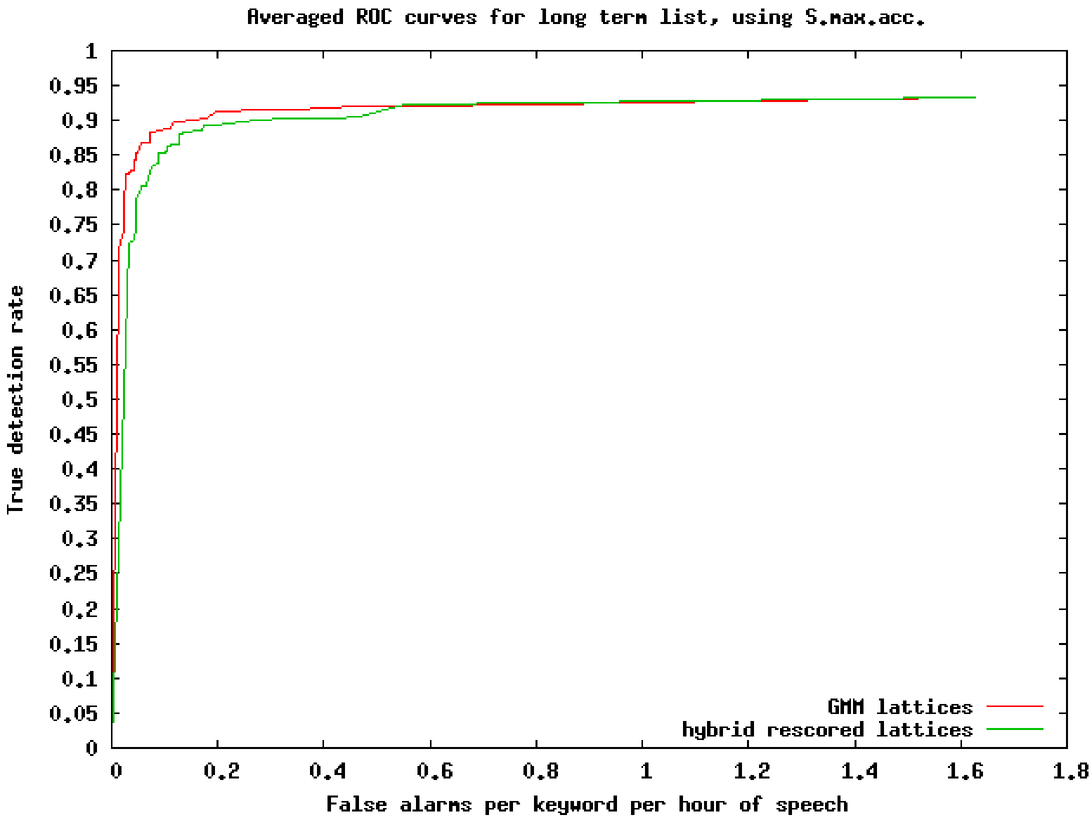

Figure 4.18: ROC curve for long keyword list, comparing GMM lattices coming from LVCSR system and hybrid rescored lattices using an MLP, for the $\mathcal{S}_{max.acc.}$ criterion. In this case, we were not able to reach more than 1.6 $FA/keyword/hour$*, so we could not compute FOMs !*

Averaged True Detection rate at 1 $FA/kw/h$**, for long keywords**		
FOM	$\mathcal{S}_{max}$	$\mathcal{S}_{max.acc.}$
GMM lattices	93.4%	92.5%
Hybrid rescored lattices	92.7%	92.8%

Table 4.3: Averaged True Detection rate at 1 $FA/kw/h$*, for long keywords, and for two posterior rescoring methods we employed. $\mathcal{S}_{max}$ simply keeps the best keyword hypothesis (the one with highest word posterior) when they overlap, while $\mathcal{S}_{max.acc.}$ accumulates posteriors among parallel overlapping hypotheses. Results are also reported for lattices generated from GMM emission probabilities (GMM lattices), and for the same lattices rescored using hybrid HMM/ANN models (hybrid rescored lattices).*

We have also plotted ROC curves for the 3 keyword lists, the two types of lattices, and using $\mathcal{S}_{max.acc.}$ in order to compare our approach to the On-Line Garbage Modeling system[5] (from figure

[5]Unfortunately, at the time of finishing this report, we had only the time to process 21% of the 3 hours of our CTS database for the On-Line Garbage based system. Thus, the associated curves which can be observed correspond to these

4.19 to 4.21). We can see that for all the lists, our system outperformed the On-Line Garbage approach, which lead us up to conclude that our posterior based LVCSR keyword spotting system is better adapted to detect keywords in large databases. Moreover, it is important to note that, in the three cases, we were not able to get enough operating points below 10 $FA/keyword/hour$ to compute FOMs. That is why we only compared ROC curves.

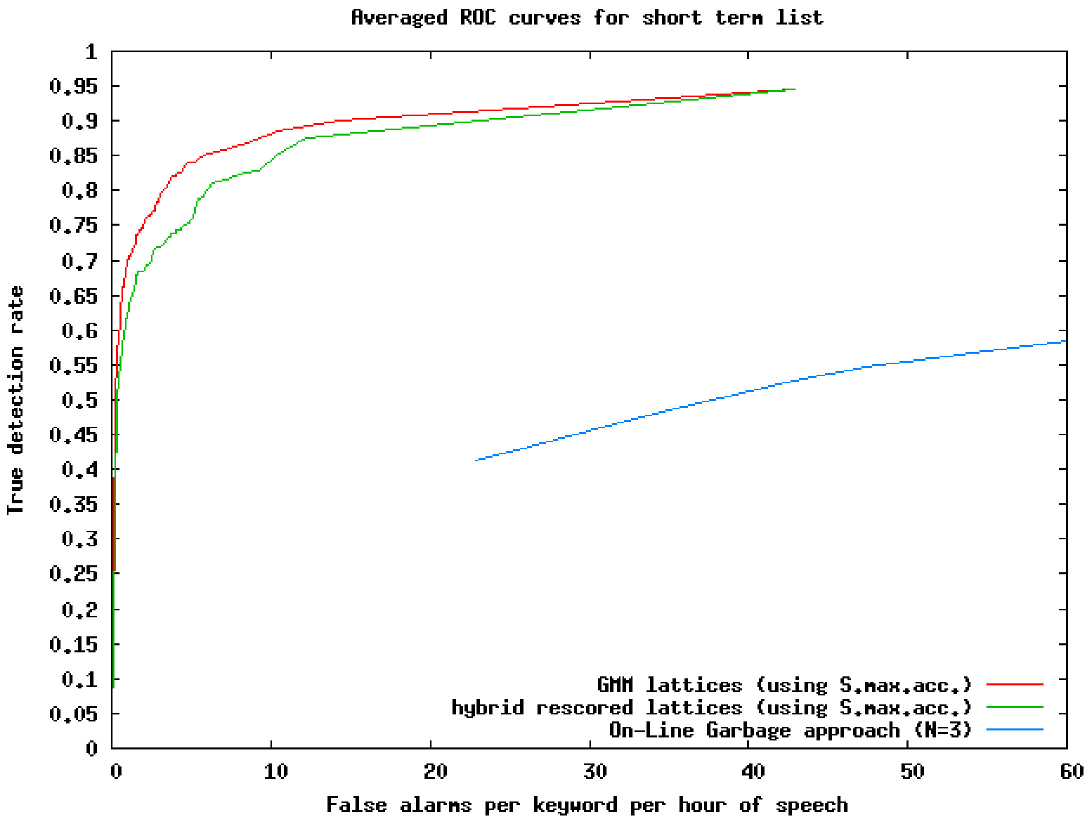

Figure 4.19: ROC curve for short keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

To finish, we have gathered our system scores in table 4.4. We can see that very accurate results are reached, especially with the long keyword list (that could be easily understood by considering the fact that short keywords are associated with a high confusion risk) : we get more than 92% detection rate for hardly 1 $FA/keyword/hour$. Detection rates for shorter keywords can also be considered as really good when compared to curves from the On-Line Garbage based system (figures 4.19 and 4.20).

21%, which are sufficient to give an idea of the results general trend.

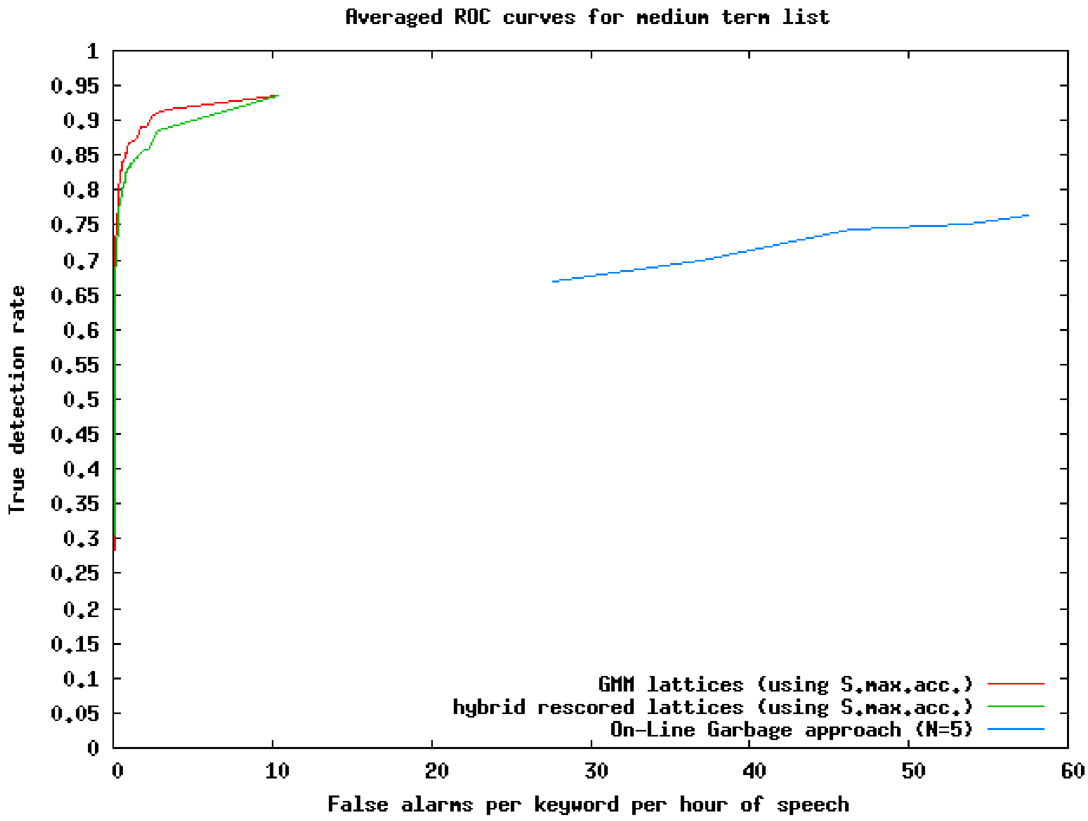

Figure 4.20: ROC curve for medium keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

Using $\mathcal{S}_{max.acc.}$	**FOM**		Det. rate at 1 $FA/kw/h$
	Short	Medium	Long
GMM lattices	82.3%	91.5%	92.5%
Hybrid rescored lattices	76.7%	89.8%	92.8%

Table 4.4: Figure Of Merits for our system, using $\mathcal{S}_{max.acc.}$ criterion (with posterior accumulation), for the 3 keyword lists (short ,medium and long), and for the two type of lattices (GMM lattices and hybrid rescored lattices).

More ROC curves were also plotted but not included in this section for clearness. To compensate, the interested reader may consult them by going to appendix A on page 81.

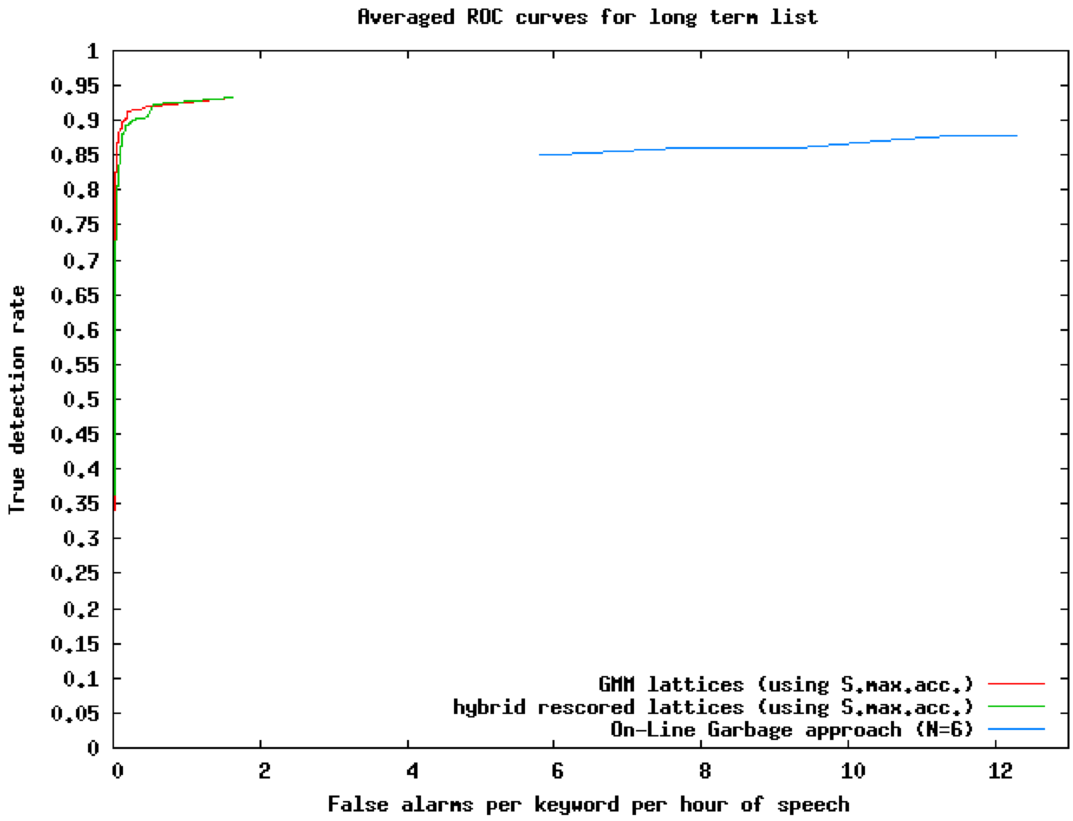

Figure 4.21: ROC curve for long keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

4.8 Conclusions and extensions

During this chapter, we have described a system which performs keyword spotting on word lattices previously generated by a Large Vocabulary Continuous Speech Recognizer (LVCSR). Keyword hypotheses were extracted from the word lattices, and associated with scores. To that end, keyword hypotheses posteriors were estimated from the lattices using a word level forward backward algorithm, and accumulated among time overlapping keyword hypotheses to take into account the posterior probability mass dispersion in parallel lattice edges (in the maneer of [50]). Then, candidate hypotheses with their scores were compared to a threshold to take the acceptance/rejection decision.

Moreover, lattice acoustic likelihoods were estimated following two ways in this work. On one hand, we started from Gaussian Mixture Model (GMM) state emission probabilities, and on the other hand, we used local state posteriors that were obtained from a Multi Layer Perceptron (MLP).

In the two cases, our word spotting system proved itself to perform well for searching keywords in large vocabulary databases, and significantly better than the well-known On-Line Garage approach, as experiments conducted on 3 hours of Conversationnal Telephone Speech (CTS) data given within the 2006 NIST Spoken Term evaluation framework have shown. Our results convinced us that within an information retrieval context, when spotting words in large databases, it was indispensable to introduce syntactic and lexical knowledge using LVCSR early passes, before working on pruned representations of the initial search space (word lattices here), in order to get accurate results. Garbage based techniques, even if they were less time and resources consuming, were not sufficient in this case, and would be restricted to simpler tasks, for dialog machines for instance, where the vocabulary is really task defined and where the system has only to deal with a few extraneous words around keywords.

More specific to our system, we have seen that results were improved when keyword hypotheses were rescored by accumulating posteriors among parallel overlapping keyword hypotheses, instead of only keeping the best one (namely the one with the highest posterior) among them. Highly accurate values were reached, especially for long keywords (that could easily be understood by considering that short keywords are associated with a high confusion risk among the entire vocabulary) : we got more than 92% of detection rate at barely 1 false alarms per keyword per hour of speech when searching for long keywords.

Besides, the results obtained from lattices rescored using an MLP, even if they were still significantly better than those from the On-Line Garbage system, seemed to be lower than the ones from

GMM lattices. We also observed that this difference tended to decrease with the keyword length. Nevertheless, it is difficult to draw a general conclusion at this level, and further investigations are needed to figure out a better explanation.

In the future, we also intend to modify our system in order to run it "on-line" : speech utterances will be turned into word lattices directly during meeting recording for instance, so that our word spotting algorithm can be applied later at each user request. We are moreover interested in our system behavior when high amount of noise is present in the speech waveform, especially when searching in lattices rescored using local state posteriors from an MLP, and part of our future work will be oriented in that direction.

Appendix A

Receiver Operating Characteristics

In this appendix, four sets of ROC curves are available for each keyword list. For instance, for the short keyword list, four ROC curves are plotted :

- one for GMM lattices using the 4 posterior rescoring criteria ($\mathcal{S}_{max}$, $\mathcal{S}_{acc}$, $\mathcal{S}_{med.acc.}$ and $\mathcal{S}_{max.acc.}$),
- one for hybrid rescored lattices also for the 4 posterior rescoring criteria,
- one for GMM lattices, hybrid rescored lattices at the same time, using only $\mathcal{S}_{max.acc.}$ criterion,
- one for comparing with the On-Line Garbage approach.

A.1 Short keywords

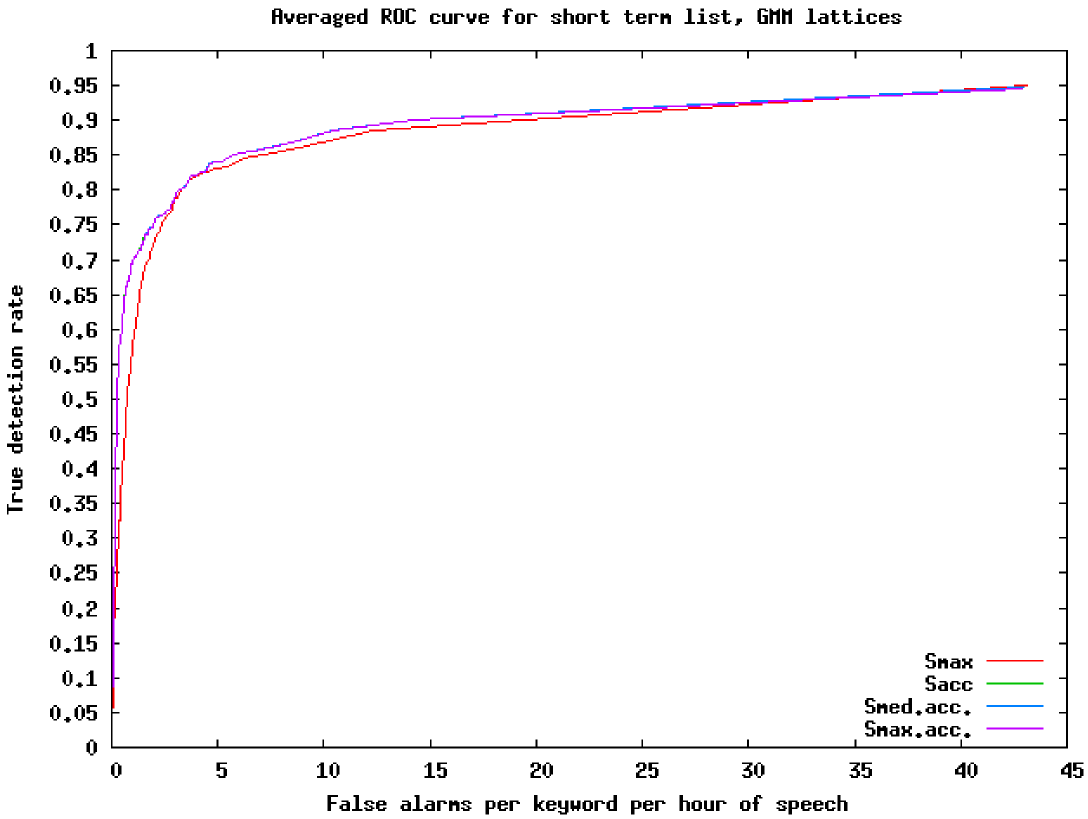

Figure A.1: ROC curve for short keyword list, using GMM lattices, for the four posterior rescoring methods.

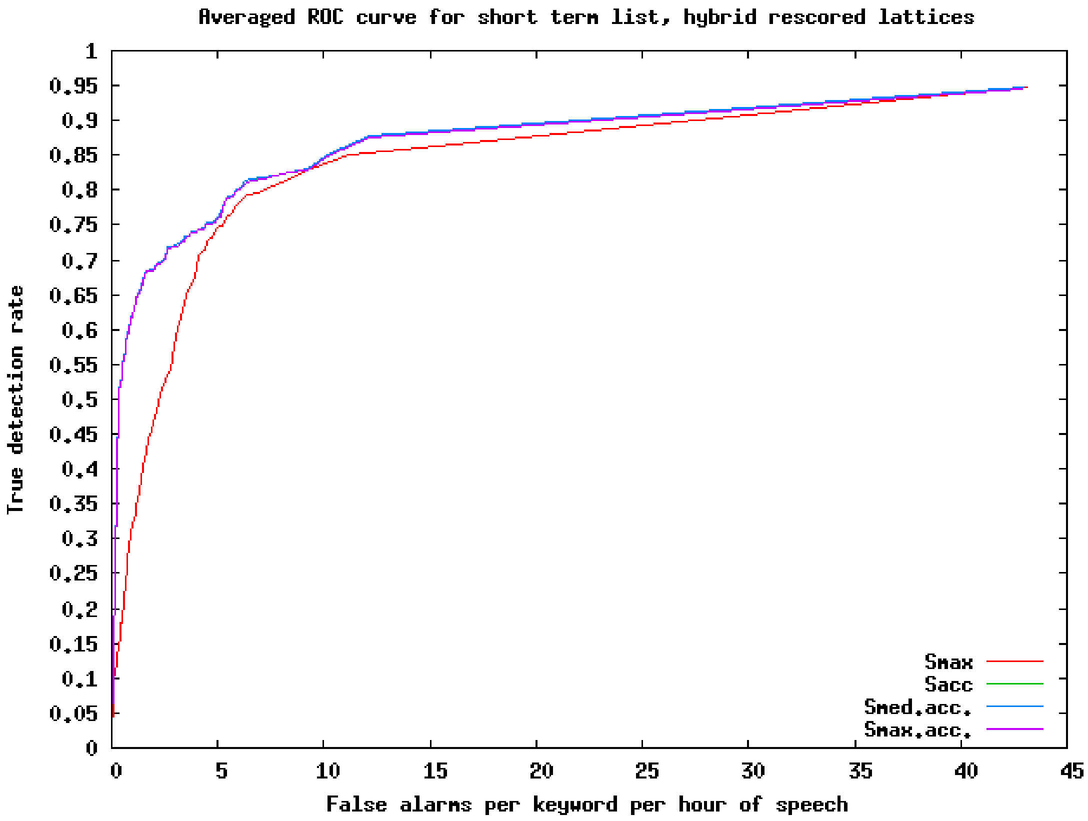

Figure A.2: ROC curve for short keyword list, using hybrid rescored lattices, for the four posterior rescoring methods.

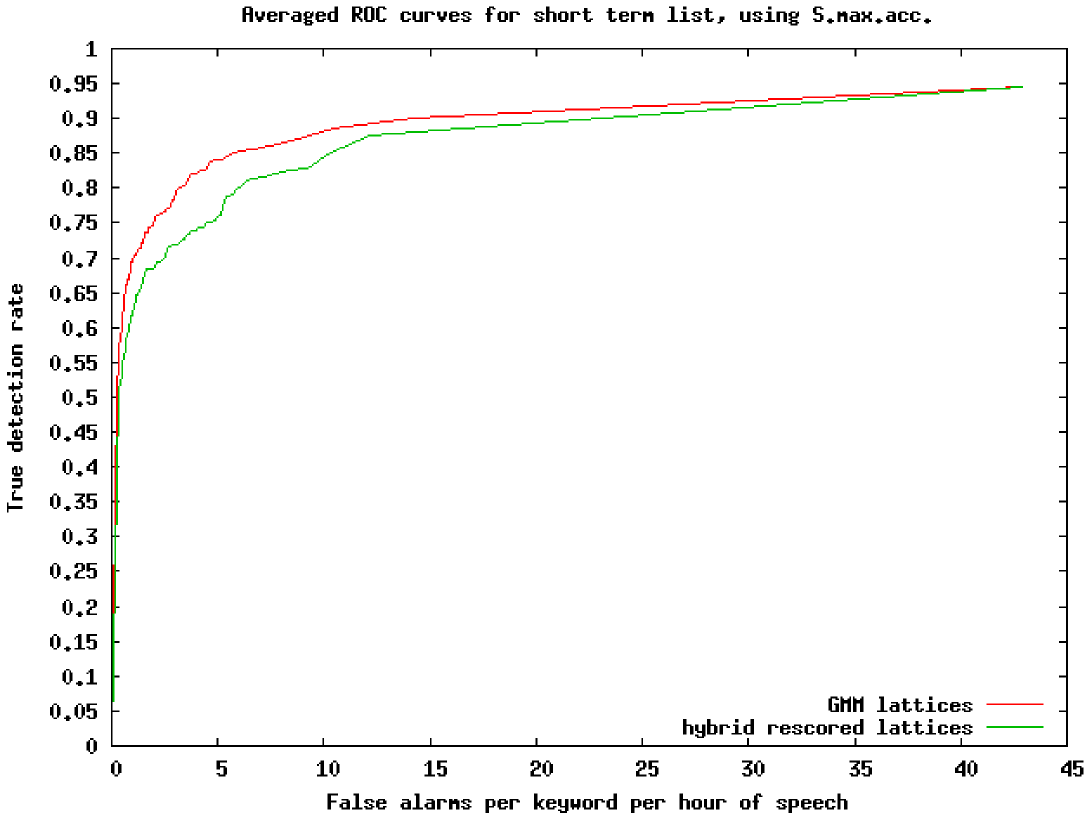

Figure A.3: ROC curve for short keyword list, comparing GMM lattices coming from LVCSR system and hybrid rescored lattices using an MLP, for the $S_{max.acc.}$ criterion.

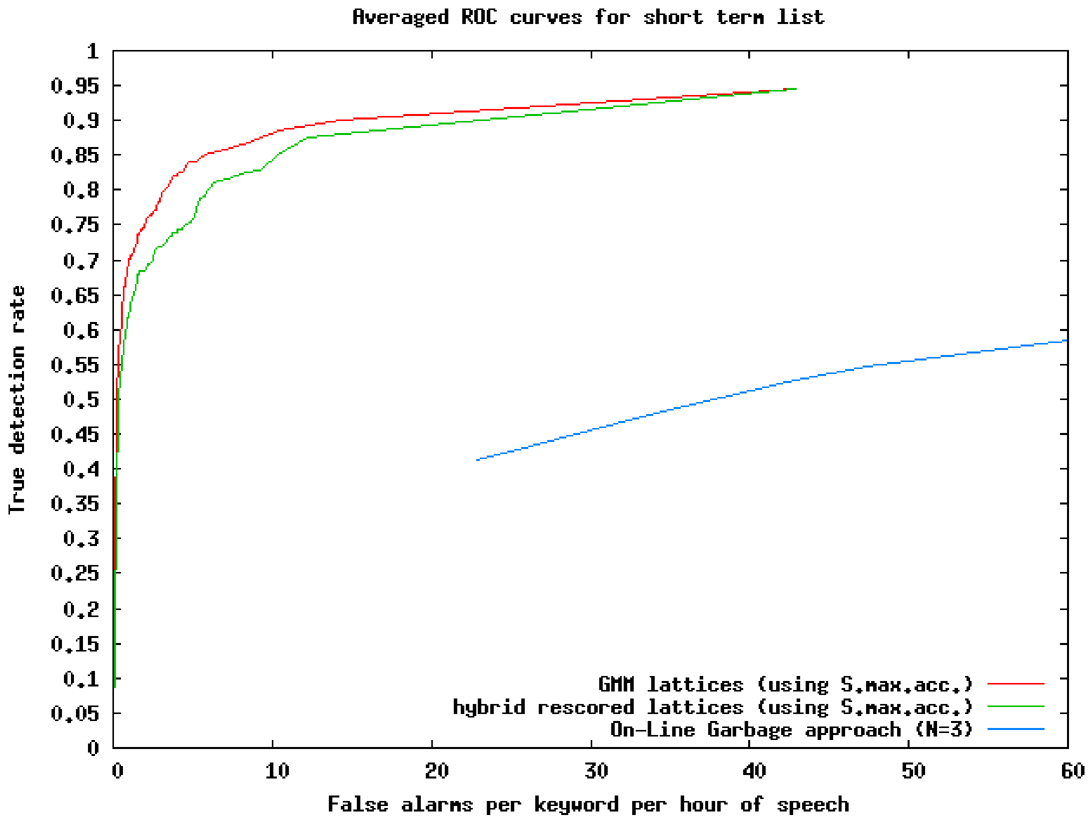

Figure A.4: ROC curve for short keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

A.2 Medium keywords

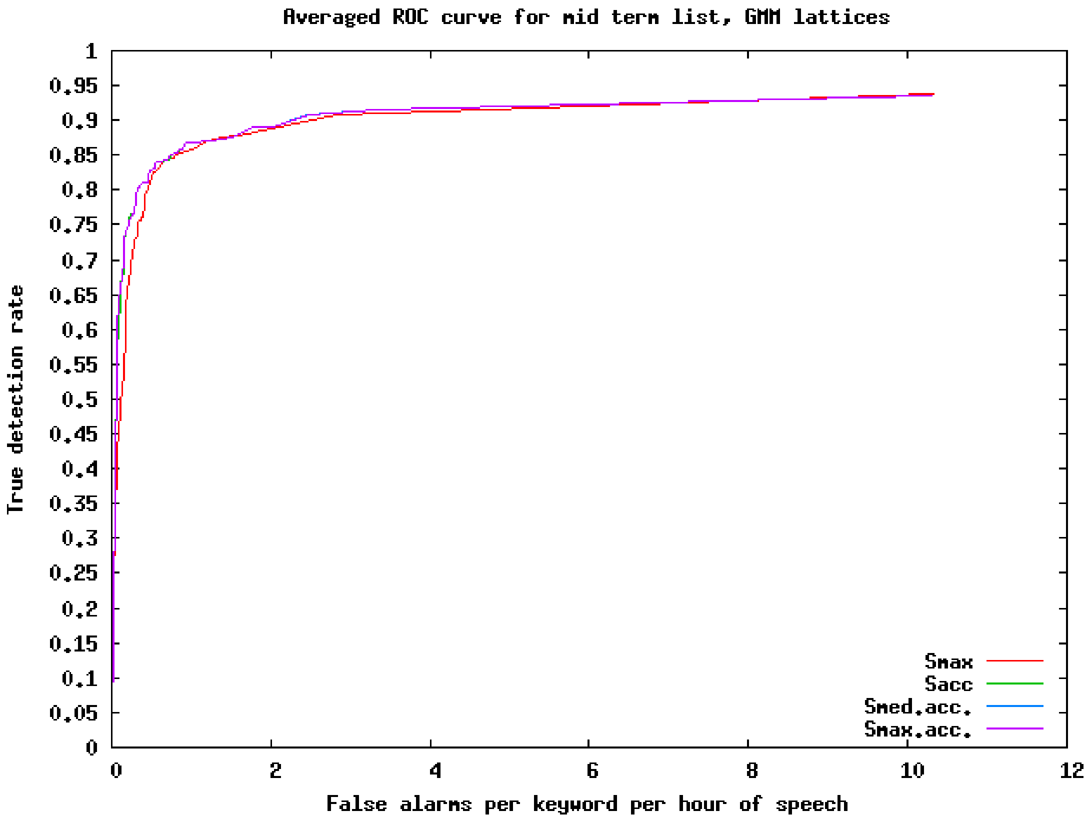

Figure A.5: ROC curve for medium keyword list, using GMM lattices, for the four posterior rescoring methods.

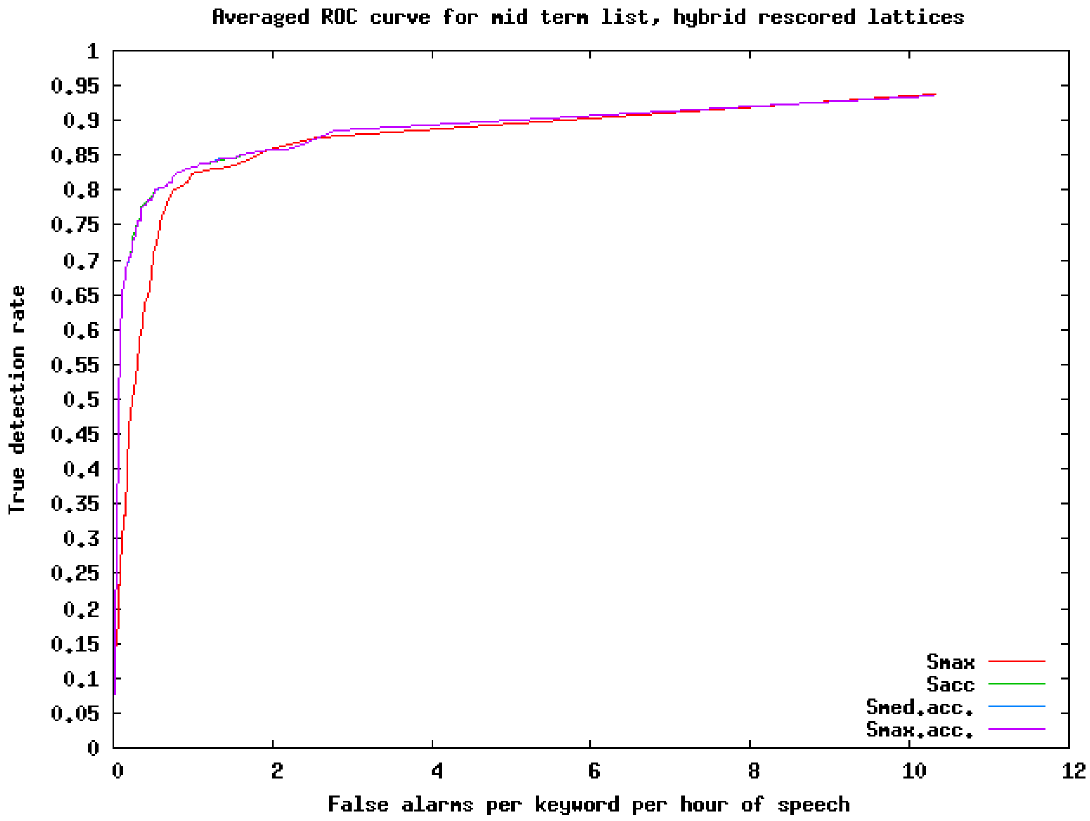

Figure A.6: ROC curve for medium keyword list, using hybrid rescored lattices, for the four posterior rescoring methods.

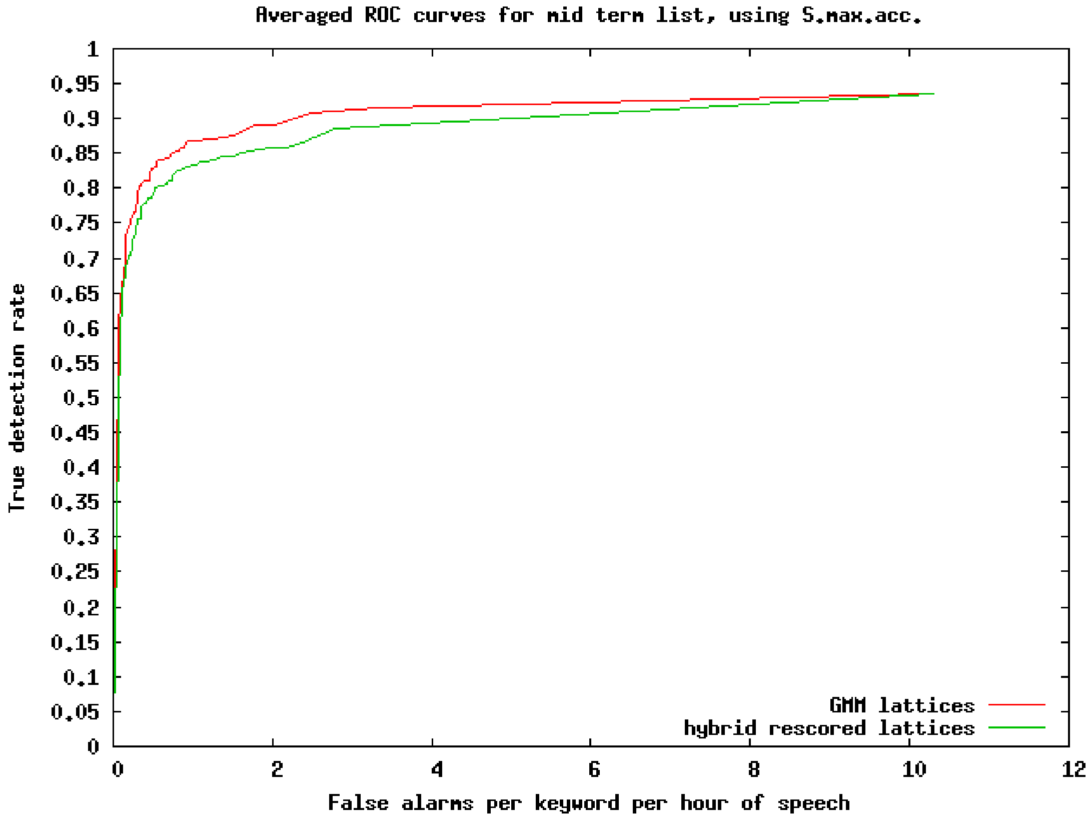

Figure A.7: ROC curve for medium keyword list, comparing GMM lattices coming from LVCSR system and hybrid rescored lattices using an MLP, for the $S_{max.acc.}$ criterion.

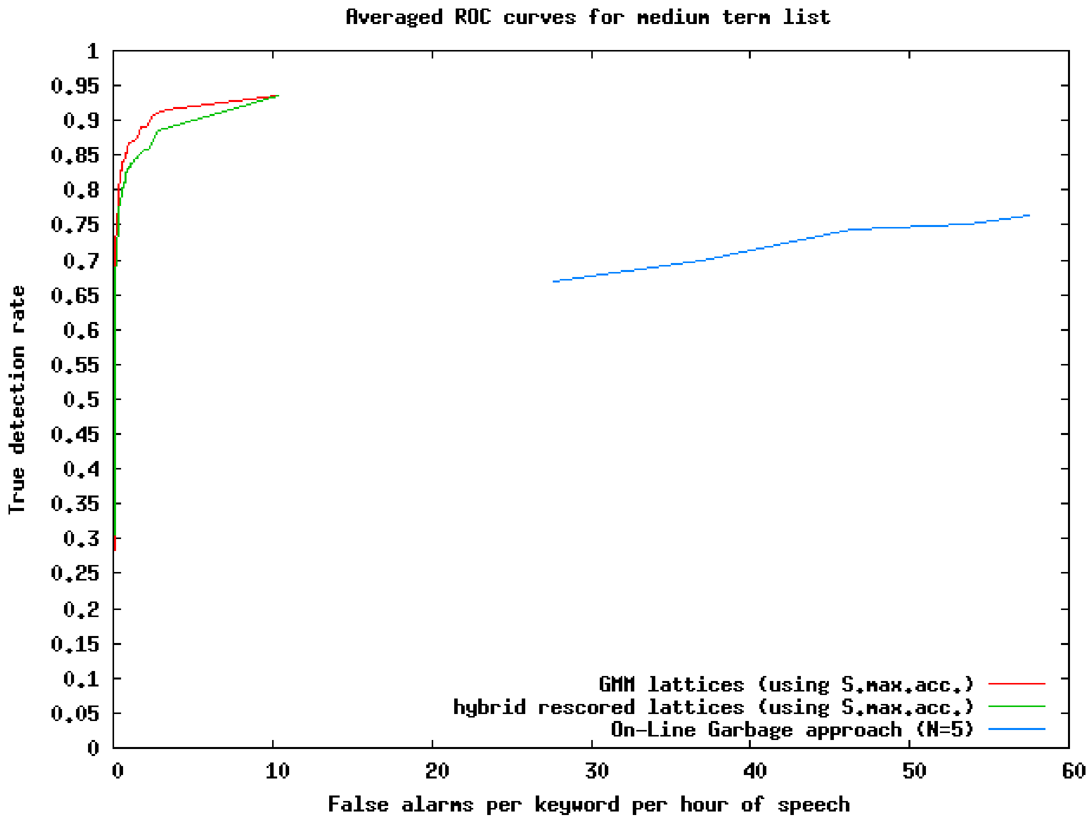

Figure A.8: ROC curve for medium keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

A.3 Long keywords

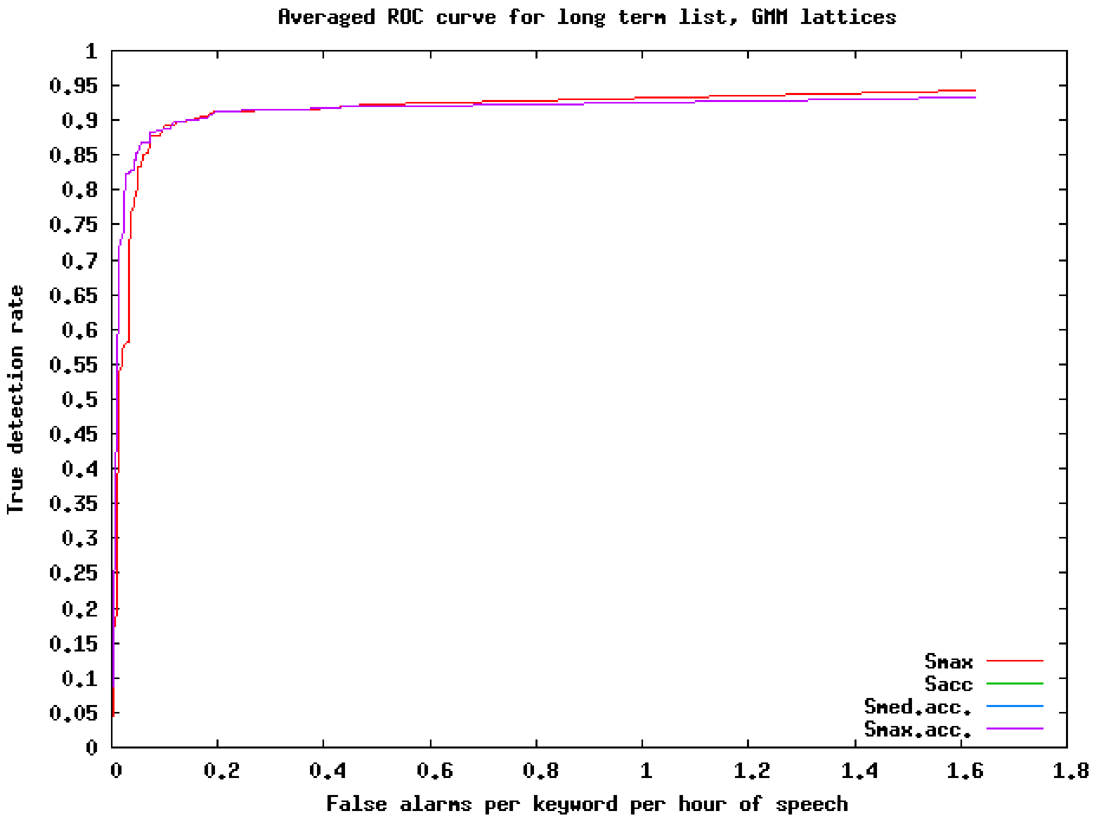

Figure A.9: ROC curve for long keyword list, using GMM lattices, for the four posterior rescoring methods.

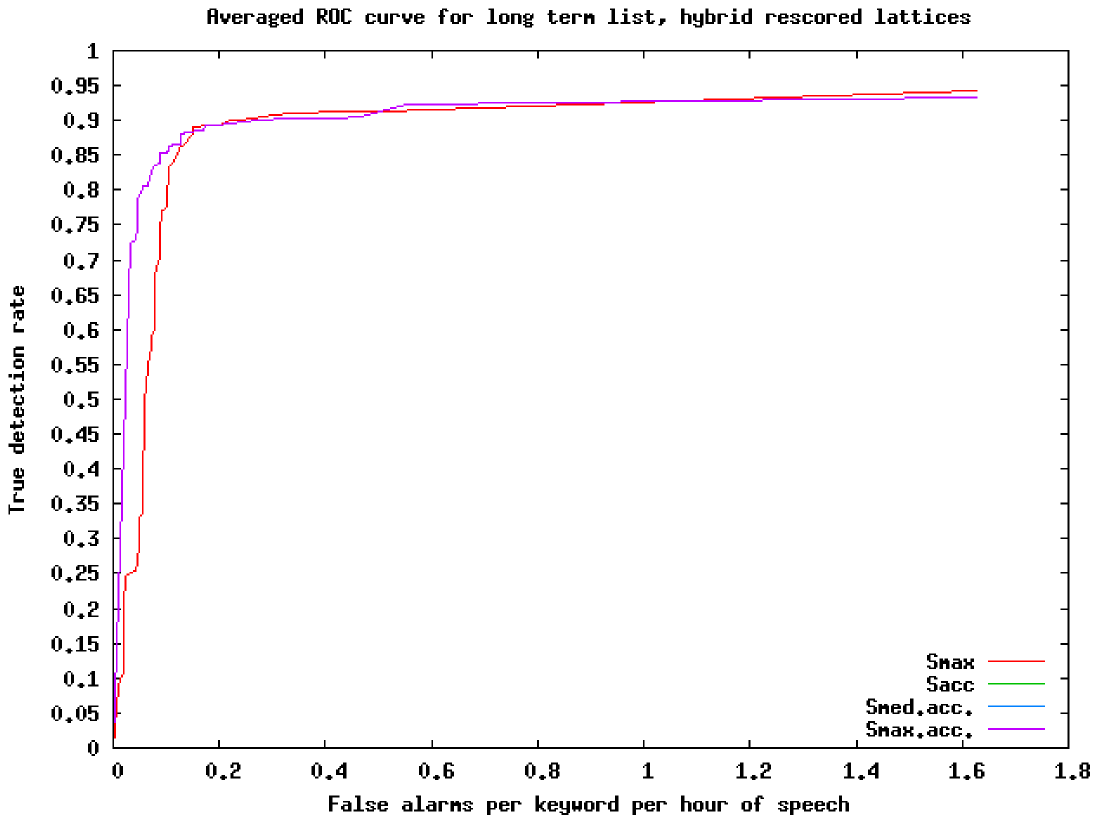

Figure A.10: ROC curve for long keyword list, using hybrid rescored lattices, for the four posterior rescoring methods.

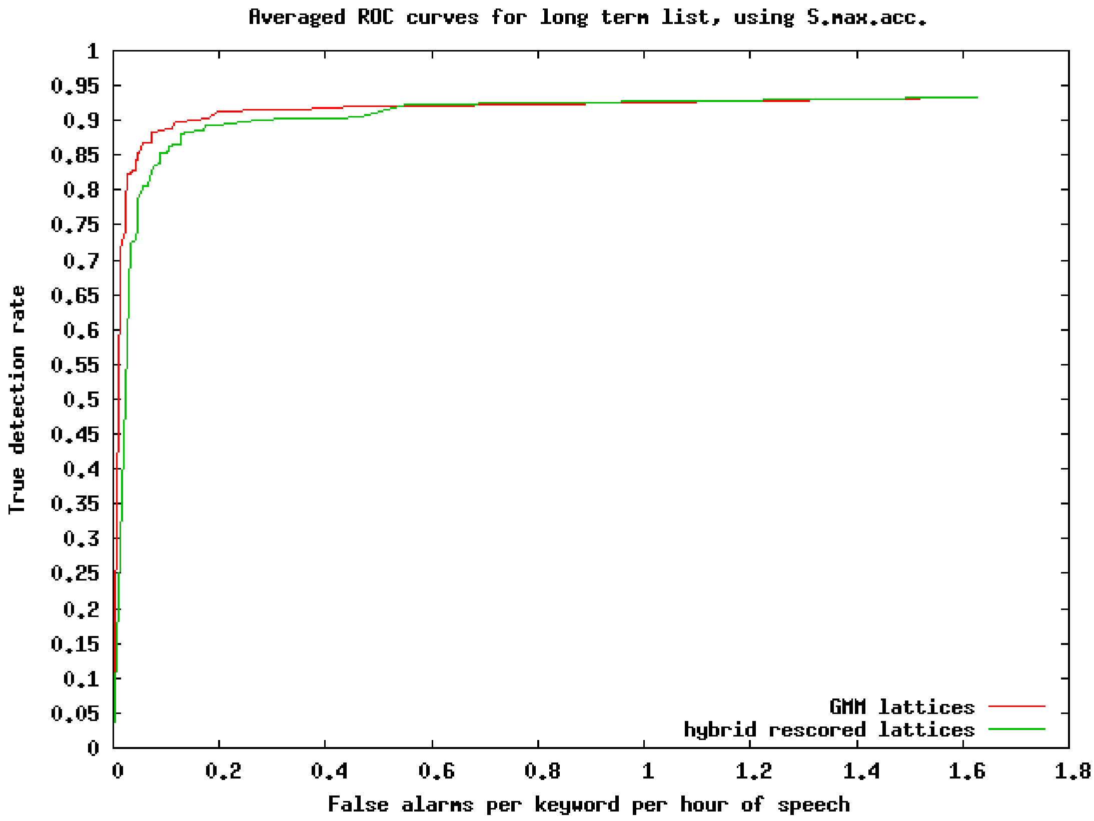

Figure A.11: ROC curve for long keyword list, comparing GMM lattices coming from LVCSR system and hybrid rescored lattices using an MLP, for the $\mathcal{S}_{max.acc.}$ criterion.

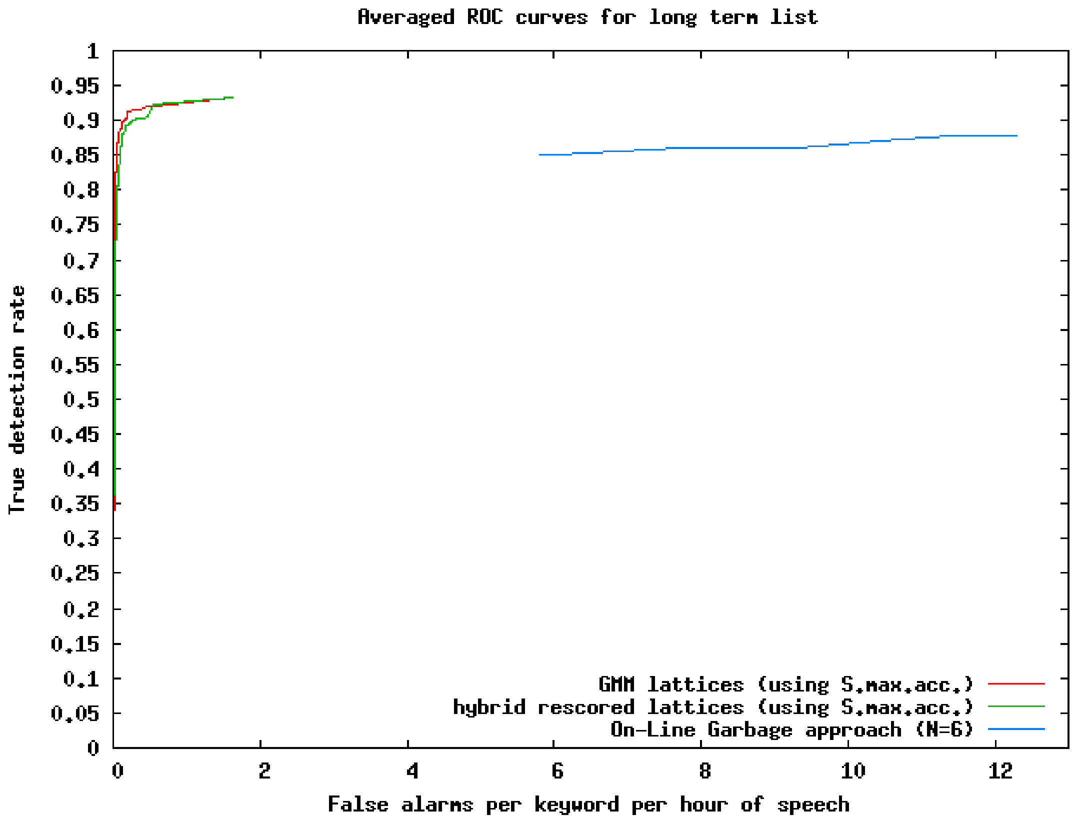

Figure A.12: ROC curve for long keyword list, comparing GMM lattices coming from LVCSR system, hybrid rescored lattices using an MLP (for the $\mathcal{S}_{max.acc.}$ criterion), and On-Line Garbage Modeling system.

Bibliography

[1] Asadi A., Schwartz R., Makhoul J. : Automatic Detection of New Words in a Large Vocabulary Continuous Speech Recognition System. In *ICASSP*, 1990, pp 125-128

[2] Asadi A., Schwartz R., Makhoul J. : Automatic Modeling for Adding New Words in a Large Vocabulary Continuous Speech Recognition System. In *ICASSP*, 1991, pp 305-308

[3] Bahl L.R., P.F. Brown, De Souza P.V., Mercer R.L. : Maximum Mutual Information Estimation of Hidden Markov Model parameters. In *ICASSP* (Tokyo), pp. 49-52, 1986

[4] Boite J-M, Bourlard H., D'Hoore B., Haesen M. : A New Approach Towards Keyword Spotting. Lernout & Hauspie Speech Products (Ieper, Belgium) ; *Eurospeech'93* (Berlin Germany September 19-23, pp. 1273-1276, 1993)

[5] Boite J-M, Bourlard H., D'Hoore B. : Optimizing Recognition and Rejection Performance in Word-Spotting Systems. In *ICASSP*, 1994, pp. I.373-I.376.

[6] Boite R., Bourlard H., Dutoit T., Hancq J., Leich H. : Traitement de la Parole. Presses polytechniques universitaires romandes (2000)

[7] Bourlard H. : Reconnaissance de la Parole et du Locuteur. Ecole Polytechnique Fédérale de Lausanne

[8] Bourlard H., Kamp Y., Ney N., C.J. Wellekens : Speaker-Dependant Connected Speech Recognition via Dynamic Programming and Statistical Methods. Speech and Speaker Recognition, *Biblthca phonet., No.12, pp. 115-148* (Karger, Basel 1985)

[9] Bourlard H., Wellekens C.J. : Links between Markov models and multilayer perceptrons. D.S. Toureski editor, Advances in Neural Information Processing Systems, Volume 1, pp. 502-510, San Mateo, C.A., 1989, IEEE, Morgan Kaufmann

[10] Bourlard H., Morgan N. : Connectionist Speech Recognition-A Hybrid Approach. Kluwer Academic Publishers, 1994

[11] Bourlard H., Silaghi M.-C. : Iterative Posterior-Based Keyword Spotting Without Filler Models. Proceedings of the *IEEE* Automatic Speech recognition and Understanding (ASRU'99)

[12] De Greve Z. : Application in Automatic Speech Recognition : Keyword Spotting Based On On-line Garbage Modeling. Internship Report (IDIAP Research Institute, 2006). Faculté Polytechnique de Mons

[13] Dutoit T. : Traitement de la parole, cours de Traitement de l'Information. Support vidéo (2000). Faculté Polytechnique de Mons.

[14] Dutoit T. : Traitement du Signal. Course notes 2005. Faculté Polytechnique de Mons.

[15] Evermann G., Woodland P.C. : Large Vocabulary Decoding and Confidence Estimation Using Word Posterior Probabilities. In *ICASSP* 2000, vol. 3, pp 1655-1658

[16] Fousek P., Hermansky H. : Multi-resolution RASTA filtering for TANDEM-based ASR. *IDIAP Research Report* (April 2005)

[17] Fousek P., Hermansky H. : Towards ASR Based on Hierarchical Posterior-based Keyword Recognition. *IDIAP Research Report* (October 2005)

[18] Goel V., Byrne W.J. : Minimum Bayes-Risk Automatic Speech Recognition. Pattern Recognition in Speech and Language Processing, W. Chou, and B.- H. Juang, (editors), CRC Press, 2003

[19] Gold B., Morgan N. : Speech and Audio Signal Processing, Processing and Perception of Speech and Music. John Wiley & Sons, 2000

[20] Gosselin B. : Classification et Reconaissance Statistique de Formes, cours de Traitement de l'Information. Course notes 2000. Faculté Polytechnique de Mons.

[21] Hain T. *et.al.* : The AMI Meeting Transcription System: Progress and Preformance, NIST RT06 evaluations, 2006

[22] Hain T. *et.al.* : The 2005 AMI System for Transcription of Speech in Meetings. In Proc. *NIST RT'05* Workshop, Edinburgh, 2005

[23] Hermansky H., Mahadeva Prasanna S.R. : Multi-RASTA and PLP in Automatic Speech Recognition. *IDIAP Research Report* (July 2006)

[24] Hermansky H. : Perceptual Linear Predictive (PLP) Analysis of Speech. The Journal of the Acoustical Society of America 87, pp. 1738-1752, 1990

[25] Higgins A.L., Wohlford R.E. : Keyword Recogntion Using Template Concatenation. In *ICASSP*, 1985, pp 1233-1236

[26] Ikbal S. : Nonlinear Feature Transformations for Noise Robust Speech Recognition. Thesis N^o3125, Ecole Polytechnique FÃl'dÃl'rale de Lausanne, Institut de Traitement des Signaux, 2004

[27] James D.A., Young S.J. : A Fast Lattice-Based Approach to Vocabulary Independent Word-Spotting. Cambridge University Engineering Department. In *ICASSP*, 1994.

[28] Junqua J.-C., Haton J.-P. : Robustness in Automatic Speech Recognition : Fundamentals and Applications. Kluwer Boston, Inc.; First edition (1996)

[29] Jurafsky D., James H.M. : Speech and Language Processing, An Introduction to Natural Language Processing, Computationnal Linguistics, ans Speech Recognition. Prentice-Hall *Inc.*, 2000

[30] Ketabdar H., Vepa J., Bengio S., Bourlard H. : Posteriors Based Keyword Spotting with A Priori Thresholds.

[31] Kopka H., Daly P.W. : A guide to $\LaTeX\,2_\varepsilon$, Document Preparation for Beginners and Advanced Users. Second Edition (1995). Addison-Wesley Publishing company

[32] Mangu L., Brill E., Stolcke A. : Finding Consensus in Speech Recognition : Word Error Minimization and Other Applications of Confusion Networks. In Computer, Speech and Language, 14(4), p.373-400, 2000

[33] Marcus J.N.. : . A Novel Algorithm for HMM Word Spotting, Performance Evaluation and Error Analysis. In *ICASSP*, 1992, pp II.89-II.92

[34] Odell J. : The Use of Context in Large Vocabulary Speech Recognition. Ph.D. thesis, Cambridge University Engineering Department, Cambridge, U.K., 1995.

[35] Pierce J.R. : Whither Speech Recognition ? The Journal of the Acoustical Society of America, october 1969, vol. 46, issue 4B, pp 1049-1051

[36] Rabiner L.R. : A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol 77, n°2, pp. 257-285, 1989.

[37] Rabiner L.R., Schafer R.W. : Digital Processing of Speech Signals. PTR Prentice-Hall *Inc.*, Englewodd Cliffs, NJ, USA, 1978

[38] Rabiner L.R., Juang B.-H. : Fundamentals of speech recognition. PTR Prentice-Hall *Inc.*Englewodd Cliffs, NJ, 1993

[39] Rohlicek J.R., Russel W., Roukos S., Gish H. : Continuous Hidden Markov Modeling for Speaker-independent Word Spotting. In *ICASSP*, 1989, pp. 627-630

[40] Rohlicek J., Jeanrenaud P., Ng K., Gish H., Musicus B., Siu M. : Phonetic Training and Language Modeling for Word Spotting. In *ICASSP*, 1993, pp 459-462

[41] Rolland C. : LaTeXpar la pratique. *O'Reilly* edition (Paris, France 1999)

[42] Rose R., Paul D. : A Hidden Markov Model Based Keyword Recognition System. In *ICASSP*, 1990, pp. 129-132

[43] Schwartz R., Nguyen L., Makhoul J. : Multiple-Pass Search Strategies, on *Automatic Speech and Speaker Recognition, Advanced Topics*, Lee C.-H., Soong F.K., Paliwal K.K. . Kluwer Academic Publishers, pp. 429-456, 1996

[44] Stolcke A. : SRILM – an extensible language modeling toolkit. In Proc. *ICSLP*, Denver, Colorado, pp. 901-904, 2002. http://www.speech.sri.com/projects/srilm/

[45] Szöke I., Schwarz P., Matejka P., Burget L., Karafiat M., Cernoky J. : Phoneme Based Acoustic Keyword Spotting in Informal Continuous Speech. In *Interspeech* 2005, pp 633-636

[46] Szöke I. *et.al.* : BUT System for NIST STD 2006. In Proc. NIST Spoken Term Detection Evaluation workshop (STD 2006), Washington D.C., US

[47] Szöke I., Schwarz P., Matejka P., Burget L., Karafiat M., Fapso M., Cernoky J. : Comparison of Keyword Spotting Approches for Informal Continuous Speech.

[48] Vertanen K. : An Overview of Discriminative Training for Speech Recognition.

[49] Weintraub M. : LVCSR Log-Likelihood Ratio Scoring for Keyword Spotting. In *ICASSP*, 1995, volume 1., pp. 297-300

[50] Wessel F., Schlüter R., Macherey K., Ney H. : Confidence Measures in Large Vocabulary Speech Recognition. IEEE Transactions on Speech and Audio Processing, vol.9, no.3, March 2001

[51] Wilpon J., Rabiner L., Lee C.-H., Goldman E. : Automatic Recognition of Keywords in Unconstrained Speech using Hidden Markov Models. *IEEE* transactions, 1990, *ASSP*. ASSP-38(11) : 1870-1878

[52] Young S.J., Russel N.H., Thornton J.H.S : Token Passing ; a Simple Conceptual Model for Connected Speech Recognition Systems. CUED Technical Report F INFENG/TR38, Cambridge University, 1989

[53] Young S. *et.al.* : The HTK book (for HTK version 3.4). Cambridge University Engineering Department, Dec. 2006.

[54] Zeppenfeld T., Houghton R., Waibel A. : Improving the MS-TDNN for Word Spotting. In *ICASSP*, 1993, pp II.475-II.478