We keep these 140 sentences (7* 20) as the test corpus in order to be speaker independent.

- For each sentence, we generate the corresponding lattice, L_i, $i = 1, \ldots, 140$.
- For each keyword, ϕ_k, $k = 1, \ldots, 7$:
  1. We compute its probability of occurrence in every sentences: $P(\phi_k | L_i)$.
  2. We sort the 140 sentences according to their probability. **Table 2** shows the position, in this classification, of the 7 sentences where the keyword is really pronounced ("true hits").
- We compute the average position of these true hits over every keyword. (see **Table3**).

muscular	1	2	3	4	5	7	12
grades	1	2	3	5	6	7	8
problems	1	2	3	4	5	6	13
ambulance	1	4	5	9	18	22	100
tradition	1	3	4	7	8	9	88
vocabulary	1	8	19	71	72	73	74
pizzerias	1	2	3	4	5	8	20
alligators	1	2	3	5	7	11	12
proceeding	1	2	3	4	5	6	7
overalls	1	2	3	5	6	14	53
informative	1	2	5	46	49	126	127
ankle	2	3	6	8	34	78	105
superb	1	2	5	28	30	45	121
silly	1	2	3	4	7	13	14
thursday	1	2	3	7	18	29	70
decorate	1	4	5	7	9	10	12
exposure	1	2	3	4	5	8	9
society	1	2	3	4	5	18	23
kidnappers	1	2	7	18	49	135	136
mirage	4	41	51	115	124	136	139

Table 2: Gaussian Models: True Hit Positions.

Reading 17th line: for "exposure", the first 5 position sentences sorted with this method actually contain this keyword, but the 6th and the 7th position sentences do not; only the 8th and 9th position sentences contain it.

Gaussian.	1.2	4.55	7.0	17.9	23.4	38.0	57.1
MLP.	1.1	2.57	7.05	12.3	23.3	29.0	47.2

Table 3: Average True Hit Positions, HMM models

Average positions are computed for the Gaussian based model and the MLP based model.

From the document analyst point of vue, we can say that "if we are only interested in finding three occurrences of a keyword, we usually only need to listen to 8-9 sentences out of 140 to find them".

In order to compare these results with those obtained with the frame labelling process shown in [1], we reproduce the latter in **Table 4**.

Gaussian.	2.1	4.4	8.75	14.4	24.3	40.6	60.1
Multi Gaussian.	2.1	4.3	8.5	14.1	24.0	39.9	61.0
MLP.	1.45	3.7	6.8	13.0	19.1	39.1	64.5

Table 4: Average True Hit Positions, Frame Labelling.

7. CONCLUSION.

The ambitious task of keyword spotting on speaker independent data without restrictions on the vocabulary has been tackled, [1][3]. A HMM approach has been used in order to compare it with older frame labeling technique. This comparison shows that a better result can be expected with this new approach. Due to the new lattice structure, an improvement in the search time has been noticed. Extensive tests on more word spotting oriented database will be achieved as well as tests on actual sound tracks of video where several speakers intervene. Ultimately the process will be coupled to a video pattern indexing tool and interactions between video and speech will be used to enhance the indexing results.

8. REFERENCES.

1. Gelin, Ph., Wellekens, C. J., *Keyword spotting for video indexing*, Proc. Int. Conf. Acoust. Speech end Signal Processing, 1996.
2. Bourlard, H., Kamp, Y., Ney, H., Wellekens, C. J., *Speaker-Dependent Connected Speech Recognition via Dynamic Programming and Statistical Methods*, Speech and Speaker Recognition, Karger, 1985.
3. James, D. A., Young, S.J., *A Fast Lattice-Based Approach to Vocabulary Independent Wordspotting*, Proc. Int. Conf. Acoust. Speech end Signal Processing, 1994.
4. Bourlard, H., Morgan, *N., Connectionist speech recognition: a hybrid approach*, Kluwer Academic Publishers,1994.
5. Rabiner, L., Juang, B-H., *Fundamentals of Speech Recognition*, PTR Prentice-Hall Inc, 1993.
6. Deller, J. R., Proakis, J. G., Hansen, J. H. L., *Discrete-Time Processing of Speech Signals*, Macmillan Publishing C^{o}, 1993.
7. Bourlard, H., Wellekens, *C. J., Links between Markov Models and Multilayer Perceptrons*, IEEE conference on Neural Information Processing Systems, 1988, Denver, CO, Ed. D. Touretzky, Morgan-Kaufmann Publishers, pp.502-510, 1989.

5. SEARCH OVER THE LATTICE

5.1. Blocking Effect

- Confusion Matrix

Since hypotheses are obtained through an N-Best like strategy, speaker variability is taken into account. For instance, if the speaker pronounces a "a" phoneme in the ["a"- "e"] area, the "a" hypothesis and the "e" hypothesis will be generated. Unfortunately, this mechanism is not efficient in case of total mispronunciation.
A confusion matrix that takes into account similarities between phonemes will give more freedom in the representation so as to mask the effect of phoneme mislabelling in the training and of phoneme misrecognition.
For each sentence of the training database, phonetic segmentation is determined according to the correct phoneme sequence. Acoustic vectors are then labeled according to this segmentation. Let us note X_L, the set of all acoustic vectors labeled by the correct phoneme φ_L. If we note φ_i the estimated phoneme, we can then compute the confusion probability:

$$P(\varphi_L|\varphi_i) = \frac{P(\varphi_i|\varphi_L)\,P(\varphi_L)}{P(\varphi_i)} = \frac{P(q_{3i} \vee q_{3i+1} \vee q_{3i+2}|\varphi_L)\,P(\varphi_L)}{P(q_{3i} \vee q_{3i+1} \vee q_{3i+2})} = \frac{[P(q_{3i}|\varphi_L) + P(q_{3i+1}|\varphi_L) + P(q_{3i+2}|\varphi_L)]}{P(q_{3i}) + P(q_{3i+1}) + P(q_{3i+2})} P(\varphi_L)$$

where q_{3i}, q_{3i+1} and q_{3i+2} are taken when using the training database with the language model, and φ_L when using the Viterbi segmentation on the known phoneme sequence.

- Transition

In [1], hypotheses were deduced from the local probabilities. Therefore the boundaries cannot be grouped as done in this paper. So, the end of an hypothesis does not necessarily coincide with the beginning of the next hypothesis. Between successive hypotheses temporal jumps are then required. Now, as boundaries hypotheses are grouped according to the Viterbi segmentation and its second iteration (see Backward part in section 4), jumps are no longer needed. This reduces searching time as hypotheses transitions boundaries are fixed.

5.2. Algorithm

Using the confusion matrix, let $\phi = \{\varphi_1, \ldots, \varphi_N\}$ be the phonetic transcription of the searched keyword.
For each *group of hypotheses* having the same boundaries, $h(\varphi_h, P_h, b, e) \in H_b^e$, where $P_h = P(\varphi_h|X_b^e)$,we can compute

$$P\left(\varphi_j|X_b^e\right) = \sum_{i=1}^{\#H_b^e} P_{h_i} P\left(\varphi_j|\varphi_{h_i}\right),\ \forall j = 1, \ldots, N$$

generating a new lattice specific to the keyword, denoted L.
Next, we search the best sequence of hypotheses denoted $\mathcal{H} = \{h_{l_1}, \ldots, h_{l_N}\}$ which maximizes the probability:

$$P(\mathcal{H}) = \prod_{i \in [l_1, \ldots, l_N]} P(h_i) \text{ , where } l_n \in [1, M]$$

and such that if $h_{l_j} \in H_{b_i}^{e_i}\ \forall i = 1, \ldots, N$,
then $e_i = b_{i+1}\ \forall i = 1, \ldots, N-1$.
The search of the optimal sequence of hypotheses is based on a recursive process.

- In step 1, the initialization process consists in searching over all the lattice L, each hypothesis $h_{l_1}(\varphi_1, P, s, t)$, $l_1 \in [1, \ldots, (M-N+1)]$ of occurrence of the first phoneme φ_1. For each of them we can initialize $\mathcal{H}^1 = \{h_{l_1}\}$ and $P(\mathcal{H}^1) = P(h_i)$, and execute the second step.
- In each step $k = 2, \ldots, N$, for the last hypothesis of $\mathcal{H}^{k-1}$, denoted $h_{l_{k-1}}(\varphi_{k-1}, P, s, t)$, we next search for hypotheses $h_{l_k}(\varphi_k, p', s', t')$ of occurrence of φ_k, such that $t = s'$.
- For each hypothesis h_{l_k} found:
  we build $\mathcal{H}^k = \{\mathcal{H}^{k-1}, h_{l_k}\}$,
  and calculate $P(\mathcal{H}^k) = P(\mathcal{H}^{k-1})P(h_i)$,
- if $k < N$, we go to step $k+1$,
- if $k = N$ and if $P(\mathcal{H}^N)$ is the maximum sequence probability encountered, we keep this sequence $\mathcal{H}^N$.
- Once each hypothesis h_{l_k} has been treated, we go back to step $k-1$.

At the end of this process which runs over the whole lattice, the sequence ϕ showing the maximum probability,

$$P(\phi|L) = \max_L \left[P(\mathcal{H}^N)\right] \text{ is found.}$$

6. RESULTS

The tests we have done are exactly the same as in [1], in order to compare them. The test aims to mimic the use of an indexing tool by a document analyst. They are based on the DARPA TIMIT corpus (1990). We randomly choose 20 different sentences out of the SX part of the test databases. We then select one keyword per sentence (see **Table 1**).

sx113 muscular	sx95 alligators	sx14 thursday
sx10 grades	sx100 proceeding	sx101 decorate
sx110 problems	sx20 overalls	sx199 exposure
sx103 ambulance	sx290 informative	sx99 society
sx137 tradition	sx109 ankle	sx102 kidnappers
sx53 vocabulary	sx373 superb	sx280 mirage
sx133 pizzerias	sx8 silly	

Table 1: List of Keywords.

Each SX sentence occurred 7 times in the test database as there are spoken by 7 different speakers.

3. LANGUAGE MODEL

The language model is based on a HMM structure composed by sub-models of phonemes, $\varphi_i \in \Phi$, connected in order to generate any possible phonetic sequence. The inter-phoneme connections are the time-shift invariant phoneme transition probabilities $P(\varphi_{i,t+1}|\varphi_{j,t})$. These probabilities are based on the number of phonetic transition, $N(\varphi_{j,t}, \varphi_{i,t+1})$, found in the training database:

$$P(\varphi_{i,t+1}|\varphi_{j,t}) = \frac{N(\varphi_{j,t}, \varphi_{i,t+1})}{\sum_v N(\varphi_{j,t}, \varphi_{v,t+1})}$$

3.1. Phoneme Models

Each phoneme, $\varphi_p \in \Phi$, is modeled by a standard 3 states HMM, where the states are denoted: $q_{3p}, q_{3p+1}, q_{3p+2}$. Each state may generate the local a priori probability $P(x_t|q)$, for an acoustic vector x_t, to be produced by a given state q.
Two different state probabilities are tested: Gaussian density based model giving $P_G(x_t|q)$ and a multi layer perceptron (MLP) based model giving an a posteriori probability, $P_{MLP}(q|x_t)$. In the latter case, Bayes rule is used to deduce an a priori probability. While the first parametric distribution has been used for classification with HMM for a long time, neural network generation of probabilities has drawn recently a lot of interest [4].

- Gaussian Distribution

Here, the parameters for each phoneme consist in a mean vector and a covariance matrix which is assumed to be diagonal for simplification purpose [6].

- Multi Layer Perceptron

It is now well accepted that the outputs of an MLP trained with a classification criterion (one active output only with all the others fired off) approximate the a posteriori probabilities. Using Bayes rule and the a priori probabilities of the classes, a posteriori probabilities can be converted into local probabilities within an irrelevant scaling factor in the Forward Backward or Viterbi algorithms, [7]. It is worth noticing that no distribution shape constrains the resulting distribution and that the a posteriori probabilities are trained according to a discriminant criterion.
It is out of the scope of this paper to describe in the details the training techniques for the determination of the distribution parameters. The learning algorithm of the MLP is a standard error backpropagation with a cross-validation test for iteration control to avoid overtraining, [2].

3.2. Models Estimation

The model we are dealing with, are estimated through a standard training iterative process, [5].

4. LATTICE GENERATION

Nodes and arcs make up the lattices.
Nodes are inter-phoneme connections that can be referred by frame number. Each arc represents a phoneme hypothesis.
Each hypothesis contains a beginning node b, an ending node e, a phoneme label φ, and the probability P that this phoneme is located between b and e. The hypothesis is denoted by $h(\varphi, P, b, e)$ where P stands for $P(X_b^e|\varphi)$ and $X_b^e = \{x_b, x_{b+1}, \ldots, x_e\}$.
The a priori probability that the vector sequence X_1^T can be associated with a specific path

$$\wp = \{q_\wp(1), q_\wp(2), \ldots, q_\wp(T)\}$$

through the HMM is given by:

$$P_\wp = \prod_{t=1}^{T} P(x_t|q_\wp(t)) P(q_\wp(t+1)|q_\wp(t)) .$$

Each sequence of states generated by a path through the HMM can also be viewed as a sequence of sub-paths through phoneme models, $\wp = \{\wp_{\varphi_1}, \wp_{\varphi_2}, \ldots, \wp_{\varphi_V}\}$, where V phonemes have been generated and $\wp_{\varphi_v} = \{q(\varphi_v, b_v), \ldots, q(\varphi_v, e_v)\}$, where $q(\varphi, t) = q_{3\varphi}$ or $q_{3\varphi+1}$ or $q_{3\varphi+2}$.
Then we can write:

$$P_\wp = \prod_{v=1}^{V} P\left(X_{b_v}^{e_v}|\varphi_v\right) P(\varphi_{v+1}|\varphi_v) ,$$

where

$$P\left(X_{b_v}^{e_v}|\varphi_v\right) = \prod_{t=b_v}^{e_v} P(x_t|q(\varphi_v, t)) P(q(\varphi_v, t+1)|q(\varphi_v, t)) .$$

At each time t, during the forward part of the Viterbi process, the probability associated with the best path finishing in each state is known.
As shown in [2], not all possible backtrack informations collected during the forward process should be saved.

- Forward

In a Viterbi approach, we only need to keep at each time t, the best finishing phoneme $\varphi(t)$ and its duration $\delta t(t)$. In a lattice making approach, as in the N-best approach, we need to keep more information. At each time t, we need to keep the N best finishing phonemes, $\varphi_1(t), \varphi_2(t), \ldots, \varphi_N(t)$, their respective durations, $\delta t_1(t), \delta t_2(t), \ldots, \delta t_N(t)$, and the probabilities associated with them $P_1(t), P_2(t), \ldots, P_N(t)$,

where $P_i(t)$ stands for $P(X_{t-\delta t_i(t)}^{t}|\varphi_i(t))$.

These associations can already be viewed as hypotheses.

- Backward

In order to be time efficient during the lattice search process, we cannot keep trace of all the exact N-best paths, as each N-best path has its own phoneme segmentation, and therefore would need to generate too much nodes. In order to limit the node generation, a two step process is used. First, the node generated by the best path (Viterbi) is kept. Second, for each selected node, we inspect the N finishing phonemes associated with, and select their beginning nodes. Third, we keep in the lattice hypotheses containing all the phonemes beginning and ending on selected nodes.
Doing so, we define *Groups of Hypotheses*, containing the same beginning and ending nodes.

KEYWORD SPOTTING ENHANCEMENT FOR VIDEO SOUNTRACK INDEXING

Philippe Gelin & Chris. J. Wellekens.

{gelin,welleken}@eurecom.fr; http://www.eurecom.fr
Institut Eurécom, Department of Multimedia Communications,
2229 route des Crêtes, BP 193, F-06904 Sophia Antipolis, France.

ABSTRACT

Multimedia databases contain an increasing amount of videos that are hardly semantically accessed. Among the useful indices that can be extracted from the sound track, the presence of a keyword at some place plays a prominent role.
This paper deals with the specificities of such a keyword spotter and the enhancement brought to our previous technique, [1] based on frame labeling. To be useful, such a keyword spotter has to be speaker independent. Moreover it has to be able to detect any word out of an open vocabulary. This directly implies the use of a phonemic representation of the word. These constraints usually lead to an excessively time consuming tool. The division of the indexing process into two parts, the first one off-line, the second one at the query time, allows a faster response.

1. INTRODUCTION

Multimedia databases contain an increasing amount of videos that are hardly semantically accessed. Content based indexing tools are thus of primary interest for easy access to the information. The search for semantically described events may rely on the video content itself (face recognition, scene understanding) but also on the sound track. Moreover containing audio and video information could enhance the indexing process. Among the useful indices that can be extracted from the sound track, the presence of a keyword at some place plays a prominent role. Other audio indices could be speaker identification, speech / non speech detection, language identification, male / female voice detection,...
This paper deals with the specificities of such a keyword spotter and the enhancement brought to our previous technique [1], based on frame labelling. To be useful, such a keyword spotter has to be speaker independent. Moreover it has to be able to detect any word out of an open vocabulary. This directly implies the use of a phonemic representation of the word. The severe constraints lead to an excessively time consuming tool. The division of the indexing process into two parts, the first one off-line, the second one at the query time, allows a faster response.
The off-line job consists in building a lattice of phoneme hypotheses. To obtain local phonetic probabilities, two models are compared: standard Gaussian distribution and neural network estimation. A Hidden Markov Model (HMM) trained on a standard corpus is used as a phonemic language. Then a Viterbi algorithm gives the best phoneme segmentation which can be viewed as an initial lattice of phoneme hypotheses. For each segment, the occurrence probability of every phoneme is computed by using the local phonetic probabilities. Using a threshold, the best hypotheses are sorted and added to the lattice. This lattice is supposed to contain all the required speech information for the search of a keyword that will take place in a on-line query. Therefore this is the only data saved for the on-line treatment.
At each query, this lattice is first enhanced according to a previously computed confusion matrix based on phonetic recognition rates of the HMM. For each hypothesis, the highest confusable phonemes are taken into account. This process is done at the query time in order to keep the size of the lattice as small as possible, but can be achieved off-line if response time is critical. Then, this enhanced lattice is parsed searching for the specific keyword. Introducing multiple phonetic transcriptions due to the various pronunciations of the searched keyword is easily done and would produce more precise detection. Experiments using TIMIT database will be reported.
The front-end analyzer is described in section 2, while section 3 describes the language model implied in the process. Two kinds of local probabilities are considered for the HMMs.
In Section 4, the two steps of hypothesis generation is analyzed. In the first one, a segmentation results from a Viterbi alignment. In a second one, the lattice is enhanced by a N-Best segmentation.
Section 5 presents our search algorithm over the lattice.
In Section 6, experiments are reported. Enhancement and results are commented.
In the conclusion, a glance on future perspectives reveals our development plans.

2. FRONT-END ANALYSIS

As in all recognition systems, speech is preprocessed over short time frames (32 msec here), shifted by 10 msec; each frame is described by a vector, x_t, in a so-called feature space $\Re^n$. This vector is composed with the 17 first Mel cepstrum coefficients, a voiceness estimator based on cepstrum coefficients, and all their delta coefficients, totaling 36 coefficients.
As it is well known [6], a voiced frame can be easily detected on a cepstrum analysis since the pitch-periodical nature of the spectrum corresponds to a peak in the cepstral domain. Knowing that fundamental frequency ranges from 40Hz to 250Hz, we can isolate the cepstrum part of this range to detect the highest energy frequency. The ratio between this energy and the average energy over this frequency domain gives us an measurement of voiceness.