# Spoken Term Detection Based on the Most Probable Phoneme Sequence

Gábor Gosztolya
University of Szeged, Hungary
Department of Informatics
Email: ggabor@inf.u-szeged.hu

László Tóth
Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences
Szeged, Hungary
Email: tothl@inf.u-szeged.hu

***Abstract*—The aim of the spoken term detection task is to find the occurrence of user-entered keywords in an archive of audio recordings. In this area, besides the accuracy of hits returned, the speed of search is also very important, for which an intermediate representation of recordings is normally used. In this paper we evaluate a spoken term detection method which represents the speech signals by their most probable phoneme sequence, on which a dynamic search is then performed. As the accuracy of the phoneme recognizer used is vital, we shall test this method by using several approaches of phoneme identification. We found that our method already achieves satisfactory accuracy, although its run time is still rather high. We also found that this approach is heavily dependent on the performance of the phoneme recognizer.**

## I. Introduction

Spoken term detection (STD) is a relatively new area, which is closely related to speech recognition. Both seek to precisely match the relation between audio speech recordings and their transcripts; but while speech recognition seeks to produce the correct transcript of speech utterances, spoken term detection seeks to locate those parts of the utterance where the user-entered keyword occurs.

In the STD task a great emphasis is placed on the speed of the search process as in practice a user expects a quick response, whereas typically several dozen hours of audio recordings have to be examined. For this reason usually some kind of intermediate representation is used, which could be slow to generate, but in which it is easy to search. Among the numerous possibilities we chose the most probable phoneme sequence as it is quite compact, while still containing enough information to make high accuracy possible. But because this approach works directly on this most probable phoneme sequence, it is vulnerable to errors; for this reason we tested a number of phoneme recognition approaches.

The structure of this paper is as follows. First we will describe the spoken term detection task and its relation to speech recognition. Alongside this, we will also introduce a formulation of the STD problem. Then we will introduce our STD algorithm and explain it in detail. After, we will explain the various phoneme recognition techniques we experimented with. Then we will describe the experimental setup, namely the evaluation methodology applied, the database used, and our process of testing. Lastly, we will present and analyze the test results, and draw some conclusions.

This research was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency.

## II. The Spoken Term Detection Task

In the spoken term detection task we would like to find the user-entered expressions (which we will call *terms* or *keywords*) in an audio database, which will be referred to as the set of *recordings*. An STD method returns a list of *hits*, each of which contain the point of occurrence (i.e. a speech signal index, starting and ending times), the term found, and a probability value that can be used to rank the hits. In contrast to other similar tasks, in STD the order of the hits does not matter; instead, the probability value is mainly used to filter the hit list further, keeping just the more probable elements.

Next we will examine the relationship between the spoken term detection task and the speech recognition one, and then present a mathematical formulation of the STD problem.

### A. Relation to Speech Recognition

The STD task is quite closely related to speech recognition; clearly, in theory, one possible solution is to perform speech recognition on the recording set, and then search for the given keywords only in the text output. This approach, however, relies heavily on the vocabulary used during speech recognition, which is a definite weak point for any search application: the list of frequent search terms changes quite rapidly. It is so mainly because most search terms are nouns, and a big proportion of them are proper nouns, whose use is liable to fluctuate quite markedly. Yahoo said that $70\%$ of the keywords entered into their search engine are nouns, and more than half of them ($40\%$) are proper nouns [1]. Due to this, nowadays in spoken term detection a great emphasis is laid on vocabulary-independence, which excludes simply searching in the text output of speech recognition performed on the recognition set. Still, spoken term detection uses many techniques developed for and used in speech recognition, so we should first examine the speech recognition problem.

In the speech recognition task we have an input speech signal $A$ and a dictionary (i.e. a list of possible words or word sequences) $W$, and we look for the most probable word $\hat{w}$ for this signal; that is,

$$\hat{w} = \arg\max_{w \in W} P(w|A). \quad (1)$$

978-1-4244-7428-8/11/$26.00 ©2011 IEEE

The *discriminative* approach of speech recognition makes use of this formula. Usually, however, Bayes' theorem is first applied to this equation, then

$$\hat{w} = \arg\max_{w \in W} \frac{P(A|w) \cdot P(w)}{P(A)}. \quad (2)$$

Now we should note that $P(A)$ has the same positive value for all $w \in W$. So if we omit it from our equations, the most probable word sequence $\hat{w}$ will still be the same, i.e. [2]

$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \quad (3)$$

The more common generative approach of speech recognition makes use of Eq. (3), and we will also use it here. This formula has two distinct components; the former one, $P(A|w)$, describes the relationship between the speech signal $A$ and the actual word $w$, and it is called the *acoustic model*. The latter one, $P(w)$, which is the *language model*, is independent of the signal $A$, and it estimates the probability of the word $w$.

In contrast to speech recognition, an important aspect of spoken term detection is its vocabulary-independence. Due to this, from the former two factors we do not want to rely on $P(w)$, but concentrate just on the acoustic model $P(A|w)$. Note, however, that omitting $P(w)$ affects the potential accuracy of spoken term detection: it is known that even human listeners rely on the meaning of sentences spoken, which is, of course, described by the language model. Moreover, some words may contain entire other words (or parts which are almost identical), and this could lead to false detections if we just rely on the acoustic model.

### B. Formulating the Spoken Term Detection Task

Spoken term detection is a relatively new task; it has, however, been intensively studied for more than a decade [3], [4]. (We also regard *keyword spotting* as just another term for open-vocabulary spoken term detection [5].) Now we will present a mathematical formulation of the STD task.

From the viewpoint of the run time requirements of a spoken term detection method, the whole process can be divided into two distinct phases: *before* the user types in the search terms, and *after* it. The run times of the former part are not too important; of course it has to finish within a reasonable time, but even ten times that of real time is still acceptable. The latter part, however, is crucial: if a user would like to find a keyword in several hours of recordings, even $1/100$th of real time might be regarded as far too slow.

To meet this requirement, it is straightforward to divide the processing of the speech signals into two parts so that after the first one (which we will call the *preparation phase*) we get an intermediate representation which is compact, but contains as much relevant information as possible. This way a search in the *search phase* could be done quite quickly, and still produce good accuracy scores. Formulating this approach, we can write

$$P(A|w) = \sum_S P(A|S, w) \cdot P(S|w), \quad (4)$$

where $S$ is some intermediate representation. Doing this, however, means that the set of all possible $S$-s has to be evaluated, which in practice makes the use of this formula quite difficult. In practice it is convenient to approximate it; that is,

$$\sum_S P(A|S, w) \cdot P(S|w) \approx \max_S P(A|S) \cdot P(S|w), \quad (5)$$

and therefore

$$\hat{S} = \arg\max_S P(A|S) \cdot P(S|w), \quad (6)$$

where $\hat{S}$ is a (simplified) intermediate representation. This equation can now be divided into two distinct parts: $P(A|S)$ describes the connection between the original speech signal and the intermediate representation, which is just what the preparation phase calculates; while $P(S|w)$ represents the relationship between the intermediate representation and the word $w$, for which the search phase is responsible.

Up until now we have analyzed the problem from a speech recognition point of view, where we first looked for the most probable word for the given speech signal, then we looked for the "ideal" intermediate representation for it. Now we will turn to the spoken term detection viewpoint, and for this reason we will further decompose Eq. (6). Due to the vocabulary independence criterion, when we look for the optimal representation $\hat{S}$, we do not want to use $w$, which leaves us with

$$\hat{S} = \arg\max_S P(A|S). \quad (7)$$

In the search phase we would like to identify those parts of speech where the occurrence of a search term $w$ is probable, which means that we are looking for all non-overlapping $L$s for which

$$P(L|w) \geq P_{min}, \quad (8)$$

where $L$ is a "continuous" part of $\hat{S}$, and $P_{min}$ is a minimum probability threshold. (We will assume this time that from a part of the representation $\hat{S}$ we can obtain the corresponding part of the speech signal.)

The choice of the format of the intermediate representation is not trivial; several choices are available in the literature. A straightforward one is a table of all phoneme probabilities for each frame (usually every 1/100th of a second) [6]; it, however, could be quite big, and performing a search in this data structure takes quite a lot of time. Another possibility is to run a speech recognizer without a language model to obtain a graph of the most probable phonemes for each utterance; it is much more compact, although searching in it could also be rather slow. We chose a more simplified form: for each speech signal just the most probable phoneme sequence is generated by the speech recognizer. Now we will explain this approach and our algorithm in more detail.

## III. Spoken Term Detection Based on the Most Probable Phoneme Sequence

When choosing an intermediate representation in the STD problem we have to satisfy two requirements. First, this representation should contain as much relevant information as possible, to achieve a good accuracy score; and second, it should be simple and very compact, to allow a quick search. Clearly, it is hard to fulfil these two requirements at the same time; thus all intermediate representations represent a trade-off between these two aspects.

In the literature it is common to keep an $N$-best list of the most probable phoneme sequences, which then produces a special type of graph called a *lattice* [7]. The nodes of this lattice are assigned to frames (i.e. time indices), while the edges between these nodes are the possible transitions belonging to one phoneme each, and they also have a probability value. In this kind of data structure a dynamic search can be performed. In practice, however, this could still prove to be too slow; for this reason we decided to just keep the most probable phoneme sequence for each recording.

This approach has two clear advantages over the lattice-based one: firstly, the number of phonemes (which form the edges of the graph) is dramatically reduced. The reason for this is that when constructing a lattice, the $N$ best phoneme edges are kept for each possible node, i.e. for each frame; from experience it means about 50 times more edges when using $N = 3$ than our approach of keeping only the most probable phoneme sequence. The second advantage of this approach is that a string itself is a much simpler data structure than a lattice, and this makes performing a search more straightforward.

This approach can be formulated in the following way. $\hat{S}$ will be represented as a phoneme sequence, $\hat{S} = s_1 \dots s_N$, and for each $s_i$ we also note its starting and ending time, its probability value and the phoneme label. Then, for a term $w = w_1 w_2 \dots w_n$ we look for all the non-overlapping subsequence $L$s for which

$$P(w|L) \geq P_{min}, \tag{9}$$

where $P_{min}$ is a threshold set previously. Making the assumption that the phonemes of a word are independent, we get

$$P(w|L) = \prod_{i=1}^{n} P(w_i|l_i) \geq P_{min}, \tag{10}$$

where $l_1, \dots, l_n$ are the phonemes of the subsequence $L$. Naturally this approach is very vulnerable to the errors of phoneme classification: during a search we can no longer use acoustic information to correct the errors of phoneme classification. (Of course this is also the drawback of the lattice-based approach, but not to this extent as we can correct these classification errors by using the alternative edges of the graph.) To compensate for this, we allowed phoneme insertions, deletions and substitutions. This means that we allow $w_i$ and $l_i$ to be empty ($\lambda$), although not at the same time; that is,

$$P(w|L) = \prod_{i=1}^{m} P(w_i|l_i) \geq P_{min}, \tag{11}$$

where by omitting the $w_i = \lambda$ values from the sequence $w_1, \dots, w_m$ we get the word $w$, and without the $l_i = \lambda$ values the sequence $l_1, \dots, l_m$ forms $L$. $P(w_i|\lambda)$ refers to the probability of deleting the phoneme $w_i$ (if $w_i \neq \lambda$), $P(\lambda|l_i)$ means the probability of inserting the phoneme of $l_i$ (if $l_i \neq \lambda$), while $P(w_i|l_i)$ is the probability of substituting phoneme $w_i$ for the phoneme of $l_i$ in the case where neither $w_i$ nor $l_i$ is $\lambda$. The optimal sequences can be determined by calculating the edit distance [8]. The probability values may be computed from the errors of the phoneme recognizer, which were determined based on its *confusion matrix* [9].

In practice when combining probability values obtained from different sources it is common to use some kind of weighting procedure, assigning different degrees of importance to the individual values. In our case we assigned different weights for the three phoneme operations, relative to the probability value of $l_i$. A further task is to set the value of the threshold $P_{min}$, where we should choose a score which works well for search terms having quite different lengths. For this reason we used an adaptive threshold, which was normalized based on the number of phonemes in the search term.

Using the most probable phoneme sequence as an intermediate representation has another potential advantage over the lattice-based one. It is common to use some kind of *indexing*; that is, use some kind of data structure for storing the inner representation which permits very quick search operations at the cost of a slower preparation phase and using (much) more storage space. It can readily be seen that indexing a phoneme array is much easier than indexing a much more complex graph; although even more sophisticated indexing methods can be used if we no longer allow phoneme insertions and deletions. In this study we have not yet applied any kind of indexing method, but we plan to do so in the near future as it is a very effective way of speeding up the search phase.

## IV. High-Precision Phoneme Identification

The STD approach described above relies on the output of the phoneme recognizer, so it is vital to use one which works with a high precision. An interesting issue is to what extent the accuracy of the phoneme recognizer affects the behaviour of a STD system; and since this cannot be determined in any other way, we decided to make experiments with a number of such recognizers.

The standard method for performing speech recognition is the application of hidden Markov models (HMMs) [2]. To get the most probable phonetic transcripts of the utterances we applied the publicly available HTK package [10]. As we intended to build an open-vocabulary system, at the language modelling level only a phoneme bigram was used, so the usual word-level model was not included in our system. In this case the accuracy of recognition just depends on the quality of the acoustic models, hence we applied various refinements to the acoustic modelling part.

Conventional HMMs apply mixtures of Gaussian curves for the estimation of likelihoods. In the simplest case one such mixture is mapped to each of the phonemes of the language. However, better results are obtained if the phones are decomposed into three pronuncation phases, or states. Hence, standard monophone systems dedicate a 3-state model to each phoneme. The recognition accuracy score can be raised further if we use different models for the same phone pronounced in a different context. These context-dependent models are called triphones, and their application is standard practice nowadays. Because we sought to test the behaviour of our STD method when using different phoneme recognition approaches, we trained the HTK system both with monophone and triphone 3-state models. As acoustic features we used the mel-frequency cepstral coeffcients (MFCCs) with their $\Delta$ and $\Delta\Delta$ values [2], which is the most conventional feature set. The monophone phone set consisted of 52 labels, while the triphone system was composed of 1073 physical states.

The acoustic models may be further refined by using more sophisticated machine learning techniques. One possibility is to apply Artificial Neural Nets (ANNs) [11] to estimate the local probability values instead of Gaussian curves. The resulting construct is called the HMM/ANN hybrid [12]. Thanks to the advantages of ANNs, a 1-state monophone hybrid model can produce just as good a accuracy score as a standard 3-state triphone HMM. However, like the conventional HMM, the hybrid can also be further refined by applying 3 states per phone instead of just one. The triphone modelling scheme may also be adapted to ANN-based modelling, but it requires a very long training time due to the increased number of models. Hence researchers have started to apply triphone models in HMM/ANN hybrids only relatively recently [13], [14].

It is also a fairly recent result that the performance of the hybrid can be improved if a second neural net is trained on a longer context of outputs produced by the ANN of the hybrid [15], [16]. The reason is that, thanks to the context, this second net is able to correct the errors of the first net to a certain extent. We shall call this construct the 2-stage hybrid model.

In our experiments with the hybrid case the following acoustic models were constructed. The standard 1-state hybrid was created by training a multi-layer perceptron on the 52 phone label targets. This neural net had 5000 hidden neurons and was trained using error backpropagation on 9 neighbouring MFCC frames as input. Then a 3-state system was created by training the neural net with three states per phone, which required modifying it so as to have 156 output neurons. The output of this ANN was then used as input for training the second net of the 2-stage models. Again, 9 neighbouring blocks of outputs were used as input features to allow context modelling. Both the 1-state and 3-state modelling tasks were repeated with the 2-stage construct. Due to its increased training time, triphone hybrid models were created only for the 2-stage scheme; the monophone results already convincingly showed the superiority of the 2-stage system over the 1-stage model (for the phone recognition results, see Table I in Section V-D).

## V. Experiments and Results

Having defined the spoken term detection method and the phoneme recognition techniques used, we now turn to the testing process. First we define the evaluation methodology, then describe the database used and the way of testing, and finally we present and analyze the results.

### A. Methods of Evaluation

A Spoken Term Detection system returns a list of hits for a query. Given the correct list of hits (the references), we should rate the performance of the system to be able to compare different systems and configurations. Since it is a standard information retrieval task, it is straightforward to apply standard IR metrics: precision and recall, defined as

$$Precision = \frac{N_C}{N_C + N_{FA}} \tag{12}$$

and

$$Recall = \frac{N_C}{N_{Total}}, \tag{13}$$

where $N_C$ is the number of correct hits returned, $N_{FA}$ is the number of false alarms, and $N_{Total}$ is the total number of reference occurrences [17]. Thus, a perfect system has both a precision and a recall score of 1 (or $100\%$). The problem with these metrics is that in practice there is a trade-off between these two values: high precision can easily lead to a low recall score, while it is easy to achieve high recall rates while getting poor precision scores. Hence it would be better to summarize the performance of a system using just one score. In IR tasks usually the F-measure is used for this, which is the harmonic mean of precision and recall, defined as

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{14}$$

This formula, however, has the drawback that it weights precision and recall equally, which might be different from our preferences. We could also use different weights for the two measures, but their relative importance is also not really clear. This is why in the field of Spoken Term Detection usually some other – although similar – measures are used, which usually further filter the list of hits returned, keeping just the more probable candidates. The one which was more commonly applied previously is the Figure-of-Merit (FOM), which the average of the recall scores when we allow only $1, 2, \ldots 10$ false alarms per hour per keyword. Another, more strict measure was defined by the National Institute of Standards and Technology (NIST) in its 2006 evaluation of Spoken Term Detection [18]. Unlike FOM, it uses all the hits supplied by the STD method in its primal form, and is defined as

$$ATWV = 1 - \frac{1}{T} \sum_{t=1}^{T} \big( P_{Miss}(t) + \beta P_{FA}(t) \big), \tag{15}$$

where $T$ is the number of terms, $P_{Miss}(t)$ is the probability value of missing the term $t$ and $P_{FA}(t)$ is the probability value of a false alarm. These probability values are defined as

$$P_{Miss}(t) = 1 - \frac{N_C(t)}{N_{Total}(t)} \tag{16}$$

| Classification Method | | Accuracy |
|---|---|---|
| Monophone | HMM, 3 states | 68.35% |
| | HMM/ANN, 1 state | 75.56% |
| | HMM/ANN, 3 states | 76.93% |
| | HMM/ANN 2-stage, 1 state | 77.46% |
| | HMM/ANN 2-stage, 3 states | 79.18% |
| Triphone | HMM, 3 states | 75.38% |
| | HMM/ANN 2-stage, 3 states | 83.33% |

TABLE I
PHONEME ACCURACY SCORES OF THE DIFFERENT PHONEME RECOGNITION TECHNIQUES.

| Classification Method | | ATWV | MTWV | FOM |
|---|---|---|---|---|
| Monophone | HMM, 3 states | 0.2625 | 0.2720 | 69.26% |
| | HMM/ANN, 1 state | 0.4125 | 0.4228 | 76.09% |
| | HMM/ANN, 3 states | 0.4880 | 0.5038 | 79.64% |
| | HMM/ANN 2-stage, 1 state | 0.5070 | 0.5262 | 86.35% |
| | HMM/ANN 2-stage, 3 states | 0.5556 | 0.5956 | 83.20% |
| Triphone | HMM, 3 states | 0.4725 | 0.4974 | 80.69% |
| | HMM/ANN 2-stage, 3 states | 0.5666 | 0.6208 | 90.90% |

TABLE II
THE PERFORMANCE OF OUR STD METHOD USING THE DIFFERENT PHONEME RECOGNITION TECHNIQUES.

and

$$P_{FA}(t) = 1 - \frac{N_{FA}(t)}{T_{speech} - N_T(t)}, \tag{17}$$

where $T_{speech}$ is the duration of the test speech in seconds. Usually the penalty factor for false alarms ($\beta$) is set to 1000. A system achieving perfect detection (i.e. having a precision and a recall of 1.0) has an ATWV score of 1.0; a system returning no hits has a score of 0.0; while a system which finds all occurrences, but produces 3.6 false alarms for each term and speech hour also has a score of 0.0 (assuming that $T_{speech}$ is significantly larger than $N_T$) [19]. Further, *max-ATWV* (or MTWV) is a (theoretical) upper bound of ATWV, where we calculate the ATWV score for every $N$-best list of the hit list returned, and take their maximum. It summarizes the performance of a given algorithm if the minimal probability threshold $P_{min}$ has been optimally chosen.

Recently ATWV has become the more frequently used evaluation metric, and we will mainly use this. Out of curiosity, however, we will also calculate the FOM and MTWV scores.

### B. The Testing Database

We used recordings of Hungarian broadcast news for testing, which were recorded from 8 different TV channels. The recordings were divided into clear, noisy and spontaneous subsets, from which we used only the clear ones. The 70 broadcast news were divided into three groups: the first, largest one (about 5 hours long) was used for training purposes. The second part (about 1 hour long) was the *development set*: these recordings were used while developing the search method and fine-tuning its parameters. The third part was the *test set* (about 2 hours long), and it was used to evaluate the overall performance of each method. We chose 25 words and expressions as search terms, which came up in the news recordings quite frequently. They varied between 6-16 phonemes in length (2-6 syllables), and they were all nouns, half of them (12) being proper nouns.

### C. The Testing Process

Our STD method has several parameters which should be fine-tuned to achieve optimal performance: we have to choose the weights for the actions of phoneme insertion, deletion and substitution, and also set the minimal probability $P_{min}$. For the former task we used the development set of recordings, and applied the MTWV metric. We tuned the three weights in two phases: first we used the same value for all three, and chose the value where the MTWV score was the highest. Then in the second step all three weights were fine-tuned independently, one after another, using step sizes of 5% of the common weight found in the first phase.

Next, having determined the weights, we had to pick a value for $P_{min}$, for which we used a quite straightforward method: we performed spoken term detection on the development set again, calculated the MTWV metric, and chose the threshold belonging to the peak value. Then we performed STD on the test set of recordings using the values determined for the weights and for the threshold. The performance of the method using different ways of phoneme classification was measured on this list in terms of ATWV, MTWV and FOM.

### D. Results

Table I shows the phoneme-level accuracy scores of the different phoneme recognition techniques, while Table II gives the accuracy scores of our STD method using them. It is quite interesting that the basic phoneme recognition method (using HMMs with Gaussians, with 3 states) with its phoneme accuracy of 68.35% can be used for speech recognition in practice; the results show, however, that it is not so for spoken term detection: the ATWV score of 0.2625 is quite poor. The reason for this is probably that the language model assists this mediocre acoustic model, but in spoken term detection we have no such correction possibility.

In general, the way the FOM metric behaves is quite similar to that of the phoneme accuracy metric: the higher the latter is, the higher the former becomes, and by about the same amount. Only the higher values are exceptions, where the FOM score increases more than the phoneme accuracy. This increase is probably due to the fact that FOM is a quite lax evaluation metric: in practice 10 false alarms per hour per keyword is barely acceptable. The ATWV score, however, is a much more strict one: an increase in phoneme-level accuracy from 68.35% to 83.33% led to an increase from 0.2625 to 0.5666 in terms of ATWV, and it rose from 0.2720 to 0.6208 when using MTWV. Interestingly, though, the error scores were cut by about the same amount (in this case by 47%, 41% and 47%). The jump between the ATWV and MTWV scores in the case of the HMM/ANN 2-stage hybrids using 3 states shows that the value found for $P_{min}$ was not stable, so probably a different search term length normalization method is called for.

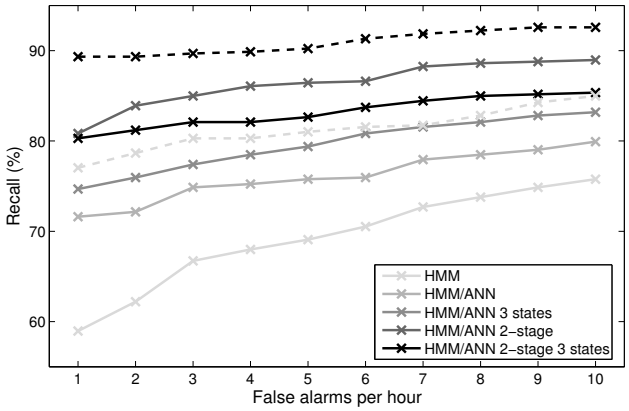


Fig. 1. The recall scores achieved when allowing 1, 2, ... 10 false alarms per hour. Continuous and dashed lines represent monophone and triphone phoneme models, respectively.

The FOM curve (which shows the recall scores as a function of the number of false alarms per hour) of the STD method using the different phoneme recognizers can be seen in Fig. 1; the FOM values shown in Table II come from calculating the mean of the values of these curves. The continuous lines show the phoneme recognizers using the monophone phoneme model, while the dashed lines show configurations with triphone phoneme models. We should add that the curves belonging to the HMM/ANN 2-stage hybrids using 3 states are quite flat, which means that most hits are found even when there are only a few false alarms. This is also reflected in the high ATWV and MTWV scores obtained for these configurations as these metrics allow quite few false alarms.

We made another important observation when examining the weights of the phoneme insertion, deletion and substitution operations. The optimal weights were so high that compared to them the probability of the individual phonemes became negligible: we got the same accuracy results when we ignored the phoneme probability values. This means that in practice only the most probable phoneme sequence is important, but the probabilities of the phonemes are not, which has several advantages. First, we have to set only two weights in this case, and this makes their tuning much easier. Second, a phoneme sequence can be stored in a simpler data structure without the probability values of each individual phoneme, which can be exploited when developing an indexing method to speed up the search phase of spoken term detection.

While the accuracy scores achieved already support practical use, the search speed unfortunately still needs to be improved: for our current implementation it takes an average of 10.75 seconds per content hour per keyword to supply a list of hits. This is partly due to technical issues: the current implementation was done in Matlab, and we concentrated primarily on accuracy and experimenting, not on classic software design. It would be a good idea to incorporate some kind of indexing mechanism, which we intend to do in the near future.

## VI. Conclusions

In the spoken term detection task we seek to find all occurrences of an user-entered expression in a list of audio recordings. As a user expects a quick response, usually the kind of intermediate representation of the recordings is used which permits a quick search. For this reason we chose the most probable phoneme sequence; but as this approach strongly depends on the phoneme recognition technique applied, we conducted experiments with a number of such methods. We found that, provided a high-accuracy phoneme recognizer is applied, our system already achieves a high accuracy. However, its running time is still too high, so we will concentrate on this issue in the future.

## References

[1] C. Barr, R. Jones, and M. Regelson, "The linguistic structure of English web-search queries," in *Proceedings of EMNLP 2008*, Waikiki, Honolulu, Hawaii, 2008, pp. 1021–1030.

[2] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*. Prentice Hall, 2001.

[3] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proceedings of SIGIR*, Amsterdam, The Netherlands, 2007.

[4] J. Junkawitsch, L. Neubauer, H. Höge, and G. Ruske, "A new keyword spotting algorithm with pre-calculated optimal thresholds," in *Proceedings of ICSLP*, vol. 4, Philadelphia, PA, USA, 1996, pp. 2067–2070.

[5] D. Wang, "Out-of-vocabulary spoken term detection," Ph.D. dissertation, University of Edinburgh, 2010.

[6] R. Wallace, R. Vogt, B. Baker, and S. Sridharan, "Optimising Figure of Merit for phonetic spoken term detection," in *Proceedings of ICASSP 2010*, Dallas, Texas, USA, 2010.

[7] I. Szöke, P. Schwarz, P. Matějka, and M. Karafiát, "Comparison of keyword spotting approaches for informal continuous speech," in *Proceedings of Eurospeech 2005*, Lisbon, Portugal, 2005, pp. 633–636.

[8] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons Inc., 2001.

[10] S. Young, *The HMM Toolkit (HTK) (software and manual)*, http://htk.eng.cam.ac.uk/, 1995.

[11] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[12] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell: Kluwer Academic, 1993.

[13] T. Pavelka and P. Král, "Neural network acoustic model with decision tree clustered triphones," in *Proceedings of MSLP*, Cancún, Mexico, 2008, pp. 216–220.

[14] A. Abad, T. Pellegrini, I. Trancoso, and J. Neto, "Context dependent modelling approaches for hybrid speech recognizers," in *Proceedings of Interspeech*, Makuhari, Japan, 2010.

[15] J. P. Pinto, G. S. V. S. Sivaram, M. Magimai.-Doss, and H. Hermansky, "Analysis of MLP based hierarchical phoneme posterior probability estimator," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 2, pp. 225–241, 2011.

[16] H. Ketabdar and H. Hermansky, "Enhanced phone posteriors for improving speech recognition systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1094–1106, 2010.

[17] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press, New York, 1999.

[18] *NIST Spoken Term Detection 2006 Evaluation Plan*, http://www.nist.gov/speech/tests/std/docs/std06-evalplan-v10.pdf, 2006.

[19] J. Pinto, H. Hermansky, I. Szöke, and S. Prasanna, "Fast approximate spoken term detection from sequence of phonemes," in *Proceedings of SIGIR*, Singapore, 2008.