Spoken Term Detection From Noisy Input

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

György Kovács and László Tóth
Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences
Szeged, Hungary
Email: {gykovacs, 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. The kind of techniques that are used usually are vocabulary-independent, using only the acoustic information available. In this scenario, however, we rely exclusively on the acoustic model, which is a drawback when it is unreliable; for example when the input is noisy. In this paper we investigate the possible accuracy of spoken term detection when the recordings were obtained using small-capacity, portable devices (wireless sensors) that have a quite low-quality microphone. The accuracy scores show, however, that despite the high amount of noise in the input recordings, our spoken term detection method can still produce an acceptable level of accuracy.**

I. Introduction

Wireless sensors and wireless sensor networks have become increasingly popular recently. Among their possible uses (like movement detection or measuring temperature and light), they are also capable of recording and transmitting speech audio data. The main weakness of wireless sensors, however, is their low capacity, which limits their possible uses. It is unlikely that speech recognition can be implemented solely on them; however, other speech technology applications which have less RAM and CPU requirements could be transferred to this platform. A straightforward choice is spoken term detection (STD), which is a relatively new area in speech technology. While speech recognition seeks to produce the correct transcript of speech utterances, spoken term detection attempts to locate those parts of the utterance where the user-entered *keyword* occurs. In the case of wireless sensors this means in practice that they are monitoring the audio data of their environment, and if the given keyword or keywords are uttered, they send a notice message to the *base station* (the central wireless sensor).

The potential of wireless sensors in this area is, however, limited due to their poor microphone, which can record only very noisy audio signals. In this study we seek to determine whether their audio quality reaches the level where spoken term detection could be performed efficiently; for this reason we perform STD experiments on noisy audio data recorded on the wireless sensors, and compare the achieved accuracy scores with the scores achieved on the same, although clean (noise-free) recordings. To make the experiments more relevant, we performed this testing using two techniques for phoneme identification.

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

The structure of this paper is as follows. First we will describe the spoken term detection task and its relation to speech recognition, and then introduce the STD algorithm we use and explain how it works. After, we will explain the phoneme recognition techniques we experimented with. Then we will describe the experimental setup, namely the evaluation methodology applied, the database and the wireless sensors used, and the process of testing. Lastly, we will present and analyze the test results, and draw some conclusions.

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 contains 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.

A. 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 [1]. (We also regard *keyword spotting* as just another term for open-vocabulary spoken term detection [2].) Now, following [3], we will present a mathematical formulation of the STD task.

As we perform exactly the same task for each given speech signal, we will present our formulation for only one speech signal A. We will denote the user-entered keyword by w, treated as a phoneme-sequence $w_1, \ldots, w_n$. In STD we are looking for each A' (which is some kind of continuous subset of A) that is likely to contain exactly w; that is,

$$P(A'|w) \geq P_{\min}. \tag{1}$$

The concept of a continuous subset is quite intuitive; in practice this simply means some parts of the input speech signal A. It is straightforward to come up with a general way of calculating $P(A|w)$, which then could be extended to work also on the A' parts of A. In the following we will deal with this case, which will be needed when we investigate the connection between spoken term detection and speech recognition.

978-1-4244-9107-0/11/$26.00 ©2011 IEEE

From the viewpoint of the run time requirements of a classic 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 part (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

$$P(A|w) = \sum_S P(A|S,w) \cdot P(S|w), \tag{2}$$

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,

$$\sum_S P(A|S,w) \cdot P(S|w) \approx \max_S P(A|S) \cdot P(S|w), \tag{3}$$

and therefore

$$\hat{S} = \arg\max_S P(A|S) \cdot P(S|w), \tag{4}$$

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.

At this point we should mention that we would like to have a vocabulary-independent solution; that is, when we calculate $\hat{S}$ in the preparation phase, we do not want to use w, but it is present in Eq. (4). In practice this means that we will use

$$\hat{S} = \arg\max_S P(A|S) \tag{5}$$

in the preparation phase. Furthermore, 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 Ls for which

$$P(L|w) \geq P_{\min}, \tag{6}$$

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

B. 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 [4]. 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 could use many techniques developed for and used in speech recognition, so next we will 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,

$$\hat{w} = \arg\max_{w \in W} P(w|A). \tag{7}$$

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

$$\hat{w} = \arg\max_{w \in W} \frac{P(A|w) \cdot P(w)}{P(A)}. \tag{8}$$

Now we should remark 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. [5]

$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \tag{9}$$

The more common generative approach of speech recognition makes use of Eq. (9), 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. Note that in practice these two values are indeed calculated independently.

So at this point we have found that what we would like to calculate in spoken term detection happens to be the acoustic model in the field of speech recognition.

The choice of the format of the intermediate representation for STD 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 might, however, be quite big, and performing a search in this data structure would take 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. Here we chose a simpler solution: for each speech signal just the most probable phoneme sequence is generated.

C. Using 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. Within 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 when 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 Ls for which

$$P(L|w) \geq P_{\min}, \tag{10}$$

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

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

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, it is common to allow phoneme insertions, deletions and substitutions. This means that we allow w_i and l_i to be empty (λ), although not at the same time; that is,

$$P(L|w) = \prod_{i=1}^{m} P(l_i|w_i) \geq P_{\min}, \tag{12}$$

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. $P(\lambda|w_i)$ denotes the probability of deleting the phoneme w_i (if $w_i \neq \lambda$), $P(l_i|\lambda)$ means the probability of inserting the phoneme of l_i (if $l_i \neq \lambda$), while $P(l_i|w_i)$ 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 λ. The optimal sequences can be determined by calculating the edit distance. The probability values may be computed from the errors of the phoneme recognizer, which were determined based on its *confusion matrix* [8]. It is also possible to assign different weights for each phoneme operation, but earlier we found [3] that it is usually not required, therefore we assigned the same weight for each operation. 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.

III. 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. The standard method for performing speech recognition is the application of hidden Markov models (HMMs) [5]; however, we used the more effective Artificial Neural Networks (ANNs) [9] instead. 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.

A typical way of enhancing the phoneme identification process is to decompose the phones into three pronunciation phases, or states. Hence, standard monophone systems dedicate a 3-state model to each phoneme. The recognition accuracy score can be further improved 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. As acoustic features we used the mel-frequency cepstral coeffcients (MFCCs) with their Δ and $\Delta\Delta$ values [5], 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.

It is also a fairly recent result that the performance of the HMM/ANN hybrid can be improved if a second neural net is trained on a longer context of outputs produced by the ANN of the hybrid [11]. 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 we tested two acoustic models. First a 3-state hybrid was created by training a multi-layer perceptron on the 156 phone label targets (i.e. three for each phonetic label). This neural net had 5000 hidden neurons and was trained using error backpropagation on 9 neighbouring MFCC frames as input. 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, serving as the second configuration tested. As we wanted to measure the effect of noise present in the input audio signals, the training and testing phases were done both for clean recordings and ones having noise in them; for the phone recognition results, see Table I in Section IV-E.

IV. 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 employed 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

$$Precision = \frac{N_C}{N_C + N_{FA}} \tag{13}$$

and

$$Recall = \frac{N_C}{N_{Total}}, \tag{14}$$

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 [12]. 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

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{15}$$

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 filter further 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 [13]. Unlike FOM, it uses all the hits supplied by the STD method in its primal form, and is defined as

$$ATWV = 1 - \frac{1}{T}\sum_{t=1}^{T}\big(P_{Miss}(t) + \beta P_{FA}(t)\big), \tag{16}$$

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

$$P_{Miss}(t) = 1 - \frac{N_C(t)}{N_{Total}(t)} \tag{17}$$

and

$$P_{FA}(t) = 1 - \frac{N_{FA}(t)}{T_{speech} - N_T(t)}, \tag{18}$$

where T_{speech} is the duration of the test speech in seconds. Usually the penalty factor for false alarms (β) 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). 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 70 broadcast news items were divided into three groups: the

Classification Method		Accuracy
Clear	HMM/ANN, 3 states	76.93%
	HMM/ANN 2-stage, triphone, 3 states	83.33%
Noisy	HMM/ANN, 3 states	66.32%
	HMM/ANN 2-stage, triphone, 3 states	73.42%

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

Classification Method		ATWV	MTWV	FOM
Clear	ANN, 3 states	0.3852	0.4101	78.63%
	ANN 2-stage, 3 states	0.6057	0.6323	90.42%
Noisy	ANN, 3 states	0.1858	0.1909	54.07%
	ANN 2-stage, 3 states	0.3985	0.3985	78.93%

TABLE II
THE PERFORMANCE OF THE PHONEMESEQUENCE-BASED STD METHOD USING THE DIFFERENT PHONEME RECOGNITION TECHNIQUES.

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. Recording Via Wireless Sensors

As in this study we wanted to test the effect of noise for spoken term detection, we performed each STD test twice: once for the original recordings, and once for recordings with significant noise. Although in these situations usually some kind of "artificial" noise (like white noise, pink noise or "babble" noise [14]) is introduced, we opted for another solution: we also recorded the broadcast news through a low-quality microphone of wireless sensors. This solution, besides having the same broadcast news recordings in two, parallel forms, also had the advantage that our noisy recordings were not created artificially, but were created in everyday circumstances, mirroring conditions of a typical application.

In this study Crossbow Iris sensor nodes (*motes*) were used that have a 8 MHz processor with a RAM of $8K$ bytes and a programmable flash memory of $128K$ bytes. The microphone and other input peripherals are located on a piece of hardware that can be attached to it, on the so-called *sensor board*. Crossbow MTS300 sensor boards were used, which, besides the microphone, also contain light and temperature sensors.

D. The Testing Process

To evaluate the effect of noise on STD accuracy scores we tested two phoneme identification techniques for both noisy and clean data, resulting in a total of four variations. For each variation we had to set the minimal probability threshold $P_{\min}$, for which we used a quite straightforward method: we performed spoken term detection on the development set, 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 value determined for the threshold. The performance of the method using different ways of phoneme classification was measured with this list in terms of ATWV, MTWV and FOM.

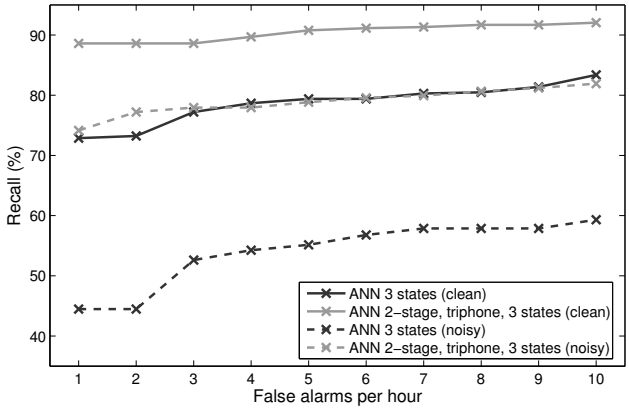

Fig. 1. Recall scores achieved when allowing 1, ... 10 false alarms per hour.

E. Results

Table I shows the phoneme-level accuracy scores of the different phoneme recognition techniques, while Table II gives the accuracy scores of the STD method using them. Figure 1 shows the FOM curve (the recall values when allowing only 1, ..., 10 false alarms per keyword per hour) of each configuration. From the phoneme-level accuracy scores it is clear that noise indeed lowers these values: the scores of the noisy recordings are about 10% lower than the corresponding ones of the clear recordings. Further, it can be seen that 2-stage training is beneficial for both the clean and the noisy cases.

Looking at the STD accuracy scores, the first thing we notice is that the MTWV and ATWV scores fall quite close to each other. As they differ only in the applied probability threshold $P_{\min}$ (MTWV uses the optimal one for the test set, whereas ATWV uses the optimal one got on the development set and used in the test), this suggests the stability of the thresholds found. Another thing to notice is the superiority of the 2-stage ANN training mechanism over the classic 3-state one in both clean and noisy environments: the accuracy scores obtained are far better in both cases.

As for the effect of noisy input, it is clear that it reduces all the accuracy scores. In the case of the standard 3-state ANN training technique without two passes, it even turned out that this configuration is practically unusable in such a noisy environment: the ATWV score of 0.1858 is very poor, and the FOM score of 54.07% is also quite low. By using the 2-stage ANN training technique, however, these scores rose to 0.3985 and 78.93%, which are practically identical to those of the one-stage configuration with a clean input, and it indicates

that it is suitable for practical applications.

V. Conclusions

In the spoken term detection task we seek to find all occurrences of a given, user-entered keyword. A straightforward extension of this problem is to monitor the audio data in some environment and send an alert when the given keyword or keywords were uttered. For this task wireless sensors are ideal devices, but due to the poor microphones they can produce only noisy audio recordings. In this study we examined the possible spoken term detection accuracy scores, and found that they can still be applied: the ATWV score of about 40% and the FOM score of almost 80% mean an accuracy level which is acceptable in practice.

References

[1] 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.

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

[3] G. Gosztolya and L. Tóth, "Spoken term detection based on the most probable phoneme sequence," in *Proceedings of the 2011 International Symposium on Applied Machine Intelligence and Informatics (SAMI) (IEEE)*, Smolenice, Slovakia, Jan 2011, pp. 101–106.

[4] 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.

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

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

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

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

[11] 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.

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

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

[14] G. Kovács and L. Tóth, "Localized spectro-temporal features for noise-robust speech recognition," in *Proceedings of the 2010 International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI) (IEEE)*, Timisoara, Romania, May 2010, pp. 481–485.