FAST KEYWORD DETECTION WITH SPARSE TIME-FREQUENCY MODELS

Effrosyni Kokiopoulou[*], *Pascal Frossard*

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - ITS
CH- 1015 Lausanne, Switzerland
{effrosyni.kokiopoulou,pascal.frossard}@epfl.ch

Olivier Verscheure

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA.
ov1@us.ibm.com

ABSTRACT

We address the problem of keyword spotting in continuous speech streams when training and testing conditions can be different. We propose a keyword spotting algorithm based on sparse representation of speech signals in a time-frequency feature space. The training speech elements are jointly represented in a common subspace built on simple basis functions. The subspace is trained in order to capture the common time-frequency structures from different occurrences of the keywords to be spotted. The keyword spotting algorithm then employs a sliding window mechanism on speech streams. It computes the contribution of successive speech segments in the subspace of interest and evaluates the similarity with the training data. Experimental results on the TIMIT database show the effectiveness and the noise resilience of the low complexity spotting algorithm.

Index Terms— Keyword spotting, sparse representations

1. INTRODUCTION AND MOTIVATION

Speech pattern matching consists in matching characteristic parameters extracted from an incoming test speech signal with those of a collection of pre-recorded reference speech templates. Keyword spotting [1], speech recognition and speaker detection are typical tasks that employ speech pattern matching techniques for recognition or detection purposes. In keyword spotting and speech recognition tasks, the test speech sample and reference speech templates are uttered words, while speaker detection uses several seconds of individuals' voices.

Conventional methods are known to degrade dramatically when a mismatch occurs between training and testing conditions. For example, an acoustic model trained using clean speech data or data from a particular environment may offer poor recognition/detection performance for noisy test data or data from a different acoustic environment. A mismatch between training and testing data can be caused by a number of factors, with background noise being one of the most prominent. Traditional approaches for removing the mismatch thereby reducing the effect of noise on recognition include: i) removing the noise from the test signal (known as noise filtering or speech enhancement [2]), and ii) constructing a new acoustic model to match the appropriate test environment (known as noise or environment compensation [3]).

More recent studies are focused on the methods requiring small knowledge about the noise or environment, since this knowledge is difficult to obtain in real-world applications involving mobile environments subject to unpredictable non-stationary noise. For example, recent studies on the missing-feature method suggest that, when knowledge of the noise is insufficient for cleaning up the speech data, one may alternatively ignore the severely corrupted speech data and focus the recognition only on the data with least contamination [4]. This can effectively reduce the influence of noise while requiring less knowledge than usually needed for noise filtering or compensation. However, the missing-feature method is only effective for partial noise corruption; that is, when the noise only affects part of the speech representation.

In this work, we propose a flexible, low complexity, pattern matching method for continuous speech streams that performs well in adverse testing environments. The method makes no assumption about the noise properties and rather relies on sparse approximation of speech elements in a time-frequency feature domain in order to identify the most meaningful features in keywords of interest. We propose to extract the meaningful signal features by a simultaneous pursuit algorithm, which identifies a common subspace model from the occurrences of a certain keyword. Speech patterns are projected on the subspace and a simple similarity distance permits to detect keywords by comparisons with the training data. An alternative solution based on sparse approximation with Stochastic Matching Pursuit has been proposed in [5] to compute linear expansions of speech signals with time-frequency atoms. However, we are rather focusing on a low complexity detection scheme that can be applied to continuous speech streams. Experimental results show that the proposed method is superior to the template matching method for keyword spotting, while being noise resilient at the same time.

[*]This work has been partly supported by NCCR IM2.

2. SPARSE REPRESENTATIONS

Sparse signal representations have attracted a lot of interest recently due to their contribution in compression and analysis problems. They exhibit interesting properties such as compact signal representation and noise resilience. In practice, one can use a greedy algorithm to build a sparse representation of a signal. In this work we focus on simultaneous sparse approximation which extracts the common sparse representation that is present in a given set of signals. For this purpose we use Simultaneous Orthogonal Matching Pursuit (SOMP) [6]. Note that SOMP has been already successfully applied to image recognition problems in our previous work [7].

Assume initially the existence of a redundant dictionary $\mathcal{D} = \{\phi_\gamma\}, \gamma \in \Gamma$, which spans the Hilbert space $\mathcal{H}$ of the signals of interest. Consider a signal as an element of $R^m \supseteq \mathcal{H}$. Assume that we have n training signals of similar structure and we want to extract their most common structure. When the training signals are stacked together they form a signal matrix, $S = [s_1, s_2, \ldots, s_n] \in \mathbb{R}^{m \times n}$. The SOMP algorithm extracts a subset Φ of the dictionary $\mathcal{D}$ which spans the common subspace where all the training signals belong to. In other words, SOMP computes a Φ such that all the columns of S are simultaneously approximated, $\|s_i - \Phi c_i\|_2 \leq \delta$, where $\|\cdot\|_2$ is the L2 norm. Note that each signal s_i is represented by a coefficient vector c_i in the common subspace spanned by Φ.

Initially, SOMP sets the residual $R_0 = S$ and then proceeds iteratively by selecting in the j-th step the atom ϕ_{γ_j} that best matches the residual R_{j-1}; that is,

$$\gamma_j = \arg\max_{\gamma \in \Gamma} \|R_{j-1}^\top \phi_\gamma\|_1.$$

At each step, it updates the residual by orthogonal projection on the span of the selected atoms (i.e., $R_j = (I - P)R_{j-1}$, where P is the orthogonal projector on the span$\{\phi_{\gamma_1}, \ldots, \phi_{\gamma_j}\}$). After K steps of SOMP, the signals in S are approximated by a linear combination of a few common atoms i.e.,

$$S = \Phi C + R_K \tag{1}$$

where R_K is the residual of the approximation. The computational complexity (of K steps) of SOMP is roughly $O((Kd + K^2)m)$, where d is the dictionary size. The first term corresponds to the atom selection step and the second to the orthogonalization cost.

3. KEYWORD SPOTTING ALGORITHM

We are ready now to describe the main steps of our keyword spotting algorithm. We have observed that different occurrences of the same keyword have similar time-frequency structures in the short-time spectral or cepstral domain (see Fig. 1). Our intuition is that SOMP is able to extract the common subspace of their similar time-frequency structures and hence to provide an effective sparse representation for further use in keyword spotting. The proposed algorithm consists of the following main steps.

Algorithm 1 Keyword spotting algorithm

```
1: **Training**
2: Extract the time-frequency features of all speech elements $s_i$ in the training set.
3: Align the training speech elements using DTW.
4: Extract the subspace model $\Phi$ using SOMP.
5: Compute the coefficient vector $c_i$ of each training speech segment $s_i$ and $\Phi^\dagger$ as well.
6: **Online Testing**
7: **for** $p = 0, \ldots$ **do**
8:     Extract the feature image $t$ of the speech segment indicated by the current window location.
9:     Compute the coefficient vector $c_t = \Phi^\dagger t$.
10:    Compute the distance among the coefficient vectors i.e., $d_{\min}(p) = \min_i d(c_t, c_i)$.
11:    **if** $d_{\min}(p) < \theta$ **then**
12:        Output that the keyword is present at position $p$.
13:    **end if**
14:    $p = p + 1$.
15: **end for**
```

1. *Feature extraction.* This step extracts the speech features of all occurrences of a certain keyword from the set of training speech signals. One approach is to use the PLP modified power spectrum [8] as feature domain. However, this does not exclude the fact that other features may be used as well. In the PLP domain, each speech signal is represented as a two dimensional signal in the time-frequency plane. Note that the PLP feature space exhibits a smooth structure which is amenable to sparse representations.

2. *Alignment.* This step is required if the reference speech elements are not of equal time span (duration). The n extracted feature sets are time aligned given a reference timespan. Various state-of-the-art techniques can be used [9]. We use here Dynamic Time Warping (DTW) in order to align training speech segments of different duration.

3. *Sparse representation model.* Next, we build a sparse representation model of the aligned speech feature time - frequency sets. In particular, we build a common subspace $\Phi \in \mathbb{R}^{d \times m}$ using SOMP. This step is generally performed off-line.

4. *Approximation in the common subspace.* In this step, each reference speech element s_i is approximated in the subspace Φ and we obtain a set of n coefficient vectors $C = [c_1, c_2, \ldots, c_n]$, which holds the weights of the linear combination in the approximation (1). C is obtained by solving n small least-squares problems of the following form $c_i = \arg\min_c \|\Phi c - s_i\|_2$, $i = 1, \ldots, n$. This can be viewed as a dimensionality reduction step since at the end of this step, each speech element s_i is represented by a small coefficient vector c_i.

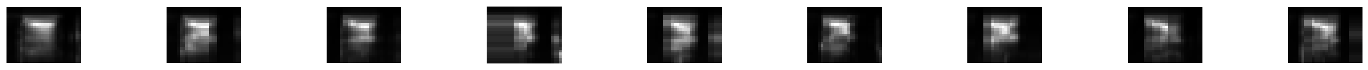


Fig. 1. The PLP features of different occurrences of “dark”.

5. *Keyword detection (online testing phase)*. Consider a test speech stream of duration at least equal to the duration of the aligned reference speech elements (possibly infinite for continuously streamed speech data). We use a sliding window mechanism where the time span of the window is equal to the duration of the reference (aligned) speech segments. Our matching method consists of the following steps. (i) The time-frequency feature image t is extracted from the segment of the test speech stream that is indicated by the current window position. The feature image is approximated in the subspace corresponding to the given keyword, resulting in a coefficient vector c_t, as $c_t = \arg\min_c \|\Phi c - t\|_2$. (ii) A distance measure $d(\cdot,\cdot) : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}^+$ is computed among c_t and each of the n coefficient vectors resulting from the training phase. An example distance metric d is the L1 or the L2 distance. The minimum ($d_{\min}$) among the n vectors is computed and compared against a predefined threshold θ. Whenever $d_{\min} < \theta$, it indicates the presence of the keyword at the current location of the time window. The sliding window then moves forward the process repeats from Step 1.

The main steps of the proposed keyword spotting algorithm are summarized in Algorithm 1. Observe that in each step the algorithm requires: (i) the solution of a small least-squares system and (ii) the computation of n distances among (low dimensional) coefficient vectors. Note in passing that the cost of step (i) can be reduced if $\Phi^\dagger$ is pre-computed, so it simplifies to a simple matrix vector product. Alternatively, the coefficients can be replaced by the values of the inner products between the test signals and the vectors that form the subspace of interest, without significant loss in performance.

4. EXPERIMENTAL RESULTS

We use the TIMIT database [10] and try to detect the keywords “dark” and “water” (independently) in the sentence “she had your dark suit in greasy wash water all year” (SA1 part) spoken by several different speakers from various dialect regions. We compare our method, called SPARSE, and the template matching method, called TM, in terms of detection performance, and noise resilience.

We build a redundant dictionary $\mathcal{D}$ of Gaussian generating functions with 5 scales, logarithmically equi-distributed in $[1, N/4]$ (where N is the signal size) and 5 orientations angles in $[0, \pi]$. We employ SOMP in order to build the subspace Φ. For time - frequency features, we use the PLP features of 12th model order, with 25msec window span and 10msec step time.

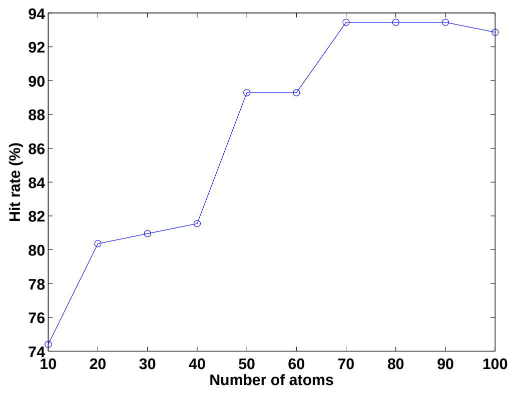


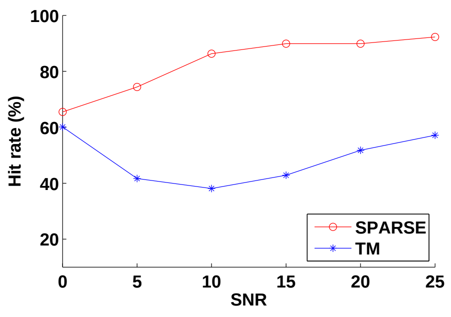
Fig. 2. Hit rate variation versus the number of atoms in SPARSE, for the keyword “dark”.

	SPARSE	TM
“dark”	93.4	89.3
“water”	91.1	76.2

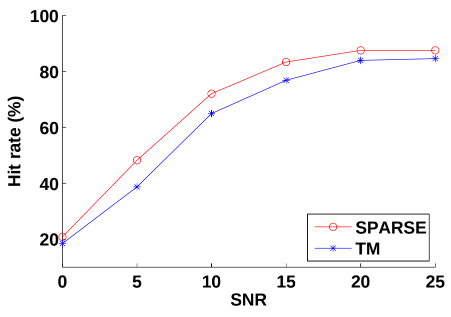
Table 1. Hit rates (%) of both methods, measured on clean data.

In the TM method [11] we use as a template the mean of the training signals; that is, the mean of the aligned different occurrences of the keyword. Since we are interested in streaming scenarios, the TM method has been implemented by applying a sliding window over the test speech signal. In each step, the features of the window speech segment are extracted and compared to the template using DTW. The total cost of the minimum path computed by DTW is used as a confidence measure of the keyword presence. Note that DTW scales as O(ℓ^2), where ℓ is the length (duration) of signals that are aligned.

We first evaluate the detection performance of our method, compared with TM. The training set consists of 462 signals and the test set of 168 signals that contain the keyword. We measure the detection performance by the *hit rate*, where hit is considered as the case when the keyword position is estimated correctly. The position information can be verified using the time alignment information from the TIMIT database, as ground truth. Fig. 2 shows the hit rate of SPARSE versus the number of atoms in Φ, for the keyword “dark”. Notice that a few atoms are sufficient to capture the time-frequency



(a)



(b)

Fig. 3. Noise resilience experiments for both methods. (a) “dark” and (b) “water”.

structure of the keyword and reach a satisfactory performance (for example, above 90%). Table 1 shows the best hit rates achieved by SPARSE along with the hit rate achieved by TM, for both tested keywords.

Finally, we test the behavior of the methods under additive white Gaussian noise. In SPARSE, we use 70 atoms for both keywods. Fig. 3 shows the variation of the hit rate when SNR varies from 0 to 25 with Step 5. Notice that in both cases SPARSE is more noise resilient than TM. Observe that “water” is harder to detect under heavy noise than “dark”, possibly due to the fact that it is longer. Notice that in both cases, SPARSE is superior to TM.

5. CONCLUSIONS

We presented a low complexity keyword spotting algorithm based on sparse representations of speech signals in the time-frequency feature space. The training signals are jointly represented in a common subspace, built by simple generating functions. The algorithm is appropriate for streaming systems and uses a sliding window mechanism over the feature stream. The keyword similarity is computed in the low dimensional space, where all the speech elements are projected. The experimental results indicated the effectiveness of the method and its noise resilience.

6. REFERENCES

[1] K. M. Knill and S. J. Young. Speaker dependent keyword spotting for accessing stored speech. *Technical report CUED/F-INFENG/TR 193, Cambridge University*, October 1994.

[2] S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. Acoust., Speech, Signal Processing*, 27(2):113–120, 1979.

[3] J. Ming. Noise compensation for speech recognition with arbitrary additive noise. *IEEE Trans. Audio, Speech and Language Processing*, 14:833–844, May 2006.

[4] J. Ming and F. Jack Smith. Speech recognition with unknown partial feature corruption-A review of the union model. *Comput. Speech Lang.*, 17(2-3):287–305, 2003.

[5] K. Wang and DM Goblirsch. Extracting dynamic features using the stochastic matching pursuitalgorithm for speech event detection. *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 132–139, 1997.

[6] J. Tropp, A. Gilbert, and M. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing, special issue “Sparse approximations in signal and image processing”*, 86:572–588, April 2006.

[7] E. Kokiopoulou and P. Frossard. Semantic coding by supervised dimensionality reduction. *IEEE Transactions on Multimedia*, 2008. to appear.

[8] H. Hermansky. Perceptual linear predictive (plp) analysis of speech. *J. Acoust. Soc. Am.*, 87(4):1738–1752, April 1990.

[9] E. A. Yfantis, T. Lazarakis, A. Angelopoulos, J. D. Elison, and Y. Zhang. On time alignment and metric algorithms for speech recognition. *Int. Conf. on Information Intelligence and Systems*, pages 423–428, 1999.

[10] L. et al. Lamel. Speech database development: Design and analysis of the acoustic phonetic corpus. *DARPA Speech Recognition Workshop, Report No. SAIC-86/1546*, pages 100–109.

[11] C. Myers, L. Rabiner, and A. Rosenberg. An investigation of the use of dynamic time warping for word spotting and connected speech recognition. *IEEE ICASSP*, pages 173–177, April 1980.