Use of GPU and Feature Reduction for Fast Query-by-Example Spoken Term Detection

Gautam Mantena, Kishore Prahallad
International Institute of Information Technology - Hyderabad, India
gautam.mantena@research.iiit.ac.in, kishore@iiit.ac.in

Abstract

For query-by-example spoken term detection (QbE-STD) on low resource languages, variants of dynamic time warping techniques (DTW) are used. However, DTW-based techniques are slow and thus a limitation to search in large spoken audio databases. In order to enable fast search in large databases, we exploit the use of intensive parallel computations of the graphical processing units (GPUs). In this paper, we use a GPU to improve the search speed of a DTW variant by parallelizing the distance computation between the Gaussian posteriorgrams of spoken query and spoken audio. We also use a faster method of searching by averaging the successive Gaussian posteriorgrams to reduce the length of the spoken audio and the spoken query. The results indicate an improvement of about 100x with a marginal drop in search performance.

1 Introduction

Query by example spoken term detection (QbE-STD) task is to detect a spoken query within a spoken audio database. In a conventional approach, an automatic speech recognition (ASR) system is used to convert the speech signal to a sequence of symbols and then text-based search techniques are exploited for search (Szöke et al., 2008), (Saraclar and Sproat, 2004), (Miller et al., 2007). However, ASR-based techniques assume the availability of labelled data for training the acoustic and language models and thus a limitation for low resource languages. To overcome this issue, dynamic time warping (DTW) based techniques are proposed for QbE-STD search (Zhang and Glass, 2009), (Anguera and Ferrarons, 2013), (Mantena et al., 2014), (Gupta et al., 2011), (Hazen et al., 2009).

Parameters extracted from the speech signal such as Mel-frequency cepstral coefficients and frequency domain linear prediction cepstral coefficients (Thomas et al., 2008), (Ganapathy et al., 2010) cannot be used directly as they also capture speaker information. To overcome this issue, Gaussian (Zhang and Glass, 2009), (Anguera and Ferrarons, 2013), (Mantena et al., 2014) and phone (Gupta et al., 2011), (Hazen et al., 2009) posteriorgrams are used as feature representations for DTW-based search. Gaussian posteriorgrams are a popular feature representation in low resource scenarios as they do not require any prior labelled data to compute them. In (Zhang and Glass, 2009), (Mantena et al., 2014), (Anguera, 2012), Gaussian posteriorgrams are shown to be a good feature representation to suppress speaker characteristics and to perform search across multilingual data.

Segmental DTW (S-DTW) is a popular technique for searching a spoken query within a spoken audio data (Zhang and Glass, 2009). In (Zhang and Glass, 2011), it is shown that the computational upper bound of S-DTW is of the order $O(mn^2)$, where m, n are the lengths of the spoken audio and spoken query respectively. To improve the search time, variants of DTW-based techniques with a computational upper bound of $O(mn)$ such as sub-sequence DTW (Anguera and Ferrarons, 2013) and non-segmental DTW (NS-DTW) (Mantena et al., 2014) are used for QbE-STD. However, DTW-based search techniques are still slow as compared to other model based approaches (Szöke et al., 2008), (Saraclar and

Sproat, 2004), (Miller et al., 2007) and thus a limitation for searching large databases.

An approach to improve the search speed is to use hardware solutions such as graphical processing units (GPUs). GPU is a computing device that are designed for intensive, highly parallel computations that are often needed in real time speech applications. In ASR, GPUs have been used to compute the acoustic likelihoods for large mixture models (Shi et al., 2008), (Cardinal et al., 2008), and in building computation intensive machine learning algorithms such as deep neural networks (Povey et al., 2011), (Bergstra et al., 2010). In (Zhang et al., 2012), GPUs were used to perform constraint based search to prune out the spoken audio references to perform the QbE-STD search. The pruning process was implemented by computing a lower bound estimate for DTW.

In this paper, we use a GPU to improve the search speed of NS-DTW using Gaussian posteriorgrams as feature representation of speech. The contributions of this paper are as follows: (a) Experimental results to show the effect of the Gaussian posteriorgram dimension on the QbE-STD search using NS-DTW algorithm, (b) GPU implementation of NS-DTW. The results indicate that by using a GPU, NS-DTW search speed can be made independent (an approximation) of the dimension of the Gaussian posteriorgram, and (c) We also use a faster method of searching by averaging the successive Gaussian posteriorgrams to reduce the length of the spoken audio and the spoken query. The results indicate an improvement of about 100x with a marginal drop in search performance.

The organization of the paper is as follows: Section 2 describes the database used in this work. In Section 3, we describe the DTW-based algorithm used to perform the search. Section 4 and Section 4.1 describes the computation of Gaussian posteriorgrams and its effect on the search speed. Section 5 describes the GPU implementation of NS-DTW and followed by conclusions in Section 6.

2 Database and Evaluation

In this work, we use MediaEval 2012 data for evaluation which consists of audio recorded via telephone in 4 South African languages (Barnard et al., 2009). We consider two data sets, development (dev) and evaluation (eval) which contain spoken audio (reference) and spoken query data. The statistics of the audio data is shown in Table 1.

Table 1: Statistics of MediaEval 2012 data.

Data	Utts	Total(mins)	Average(sec)
dev reference	1580	221.863	8.42
dev query	100	2.372	1.42
eval reference	1660	232.541	8.40
eval query	100	2.537	1.52

All the evaluations are performed using 2006 NIST evaluation criteria (Fiscus et al., 2007), (Metze et al., 2012) and the corresponding maximum term weighted values (MTWV), average miss probability (MP) and false alarm probability (FAP) are reported.

3 QbE-STD using NS-DTW

In this paper, a variant of DTW-based technique referred to as non-segmental DTW (NS-DTW) is used for QbE-STD search (Mantena et al., 2014). Let Q and R be a spoken query and a spoken audio (or reference) containing n and m feature vectors respectively and are given as follows:

$$Q = [\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_i}, \ldots, \mathbf{q_n}]_{d \times n} \quad (1)$$
$$R = [\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_j}, \ldots, \mathbf{u_m}]_{d \times m} \quad (2)$$

Each of these feature vectors represent a d dimensional Gaussian posteriorgrams (as described in Section 4). The distance measure between a query vector $\mathbf{q_i}$ and a reference vector $\mathbf{u_j}$ is the negative logarithm of the cosine similarity of the two vectors and is given by:

$$d(i, j) = -log\left(\mathbf{\hat{q}_i}^T \mathbf{\hat{u}_j}\right), \quad (3)$$

where $\mathbf{\hat{q}_i} = \frac{\mathbf{q_i}}{||\mathbf{q_i}||}$ and $\mathbf{\hat{u}_j} = \frac{\mathbf{u_j}}{||\mathbf{u_j}||}$.

A similarity matrix S of size $m \times n$ is computed to align the reference and query feature vectors. Let i, j represent a column and a row of a matrix. For DTW search, the start and end time stamps are approximated by allowing the query to start from any point in the reference and is given by $S(1, j) = d(1, j)$. Once the matrix S is initialized the update equations for alignment are given by Eq. 4.

$$S(i,j) = min \left\{ \begin{array}{c} \frac{d(i,j) + S(i-1,j-2)}{T(i-1,j-2)+1} \\ \frac{d(i,j) + S(i-1,j-1)}{T(i-1,j-1)+2} \\ \frac{d(i,j) + S(i-1,j)}{T(i-1,j)+1} \end{array} \right\}, \tag{4}$$

where $T(i,j)$ is a transition matrix which represents the number of transitions required to reach i, j from a start point. On computing the similarity matrix, the end point of the query within a reference is given by $\min_j\{S(n,j)\}$ and followed by a path traceback to obtain the start time stamp.

In segmental-DTW the reference is partitioned based on the length of the query and a DTW is performed for each partition resulting in a computational upper bound of $O(mn^2)$ for search. However, in NS-DTW, the reference is not partitioned and a similarity matrix of size $m \times n$ is computed resulting in a computational upper bound of $O(mn)$ (Mantena et al., 2014).

During the QbE-STD search, there is a possibility of the query to be present in the reference more than once. Hence, 5 best alignment scoring indices are considered from the similarity matrix (Mantena et al., 2014).

4 Feature Representation using Gaussian Posteriorgrams

In general, Gaussian posteriorgrams are obtained by a two step process (Anguera, 2012), (Mantena et al., 2014). In the first step, acoustic parameters such as frequency domain linear prediction cepstral coefficients (FDLP) are extracted from the speech signal (Mantena et al., 2014). A 25 ms window length with 10 ms shift is considered to extract 13 dimensional features along with delta and acceleration coefficients for FDLP. An all-pole model of order 160 poles/sec and 37 filter banks are considered to extract FDLP.

In general, spectral features such as Mel-frequency cepstral coefficients (MFCC) are used to compute Gaussian posteriorgrams. However, in (Mantena et al., 2014), we have shown that the FDLP parameters were working better than the conventional features such as MFCC.

In the second step, Gaussian posteriorgrams are computed by training a Gaussian mixture model (GMM) with d number of Gaussians using the spoken audio data and the posterior probability obtained from each Gaussian is used to represent the acoustic parameter. Thus, 39 dimensional FDLP parameters are mapped to d dimensional Gaussian posteriorgrams. In Section 4.1 we provide experimental results to show the effect of d on the search speed in the context of QbE-STD.

4.1 Effect of Gaussian Posteriorgram Dimension on the Search Time

In this Section, we compute the search performance and search time on dev dataset by varying the number of Gaussians (d). We consider the search time as the time required to search all the queries within a reference dataset. Table 2 show the miss probability (MP), false alarm probability (FAP), maximum term weighted value (MTWV), and search time (in minutes) obtained using the Gaussian posteriorgrams of FDLP for various values of d.

Table 2: Miss probability (MP), false alarm probability (FAP), maximum term weighted value (MTWV) and search time on dev dataset for various Gaussian posteriorgram dimensions (d) (Mantena et al., 2014).

d	**MP**	**FAP** (10^{-2})	**MTWV**	**Search Time (mins)**
8	0.824	0.595	0.084	18.39
16	0.652	0.917	0.207	22.57
32	0.540	1.098	0.292	30.60
64	0.465	1.207	0.349	47.53
128	0.426	1.136	0.399	80.24
256	0.400	1.241	0.410	145.07
512	0.476	0.658	0.422	274.98
1024	0.413	1.009	0.432	534.15

From Table 2, it can be seen that MTWV increases with an increase in d. However, with an increase in d there is increase in search time and thus resulting a slower QbE-STD search. In (Mantena et al., 2014), $d = 128$ is considered as an optimum value of the Gaussian posteriorgram dimension based on the MTWV and the search time. A more detailed description of the performance of NS-DTW by varying the dimensions of the Gaussian posteriorgrams is given in (Mantena et al., 2014).

To better understand the computation intensive components in NS-DTW, we calculate the time required for distance computation (as given by Eq. (3)) and to perform the update equations (as given by Eq. (4)) along with the path traceback for $d = 128$ (as shown in Table 3). It is to be noted that the path traceback includes the selection of 5 best alignment scoring indices from each of the reference file and thereby obtaining the start and end time stamps.

Table 3: Time taken for distance computation, $d(i, j)$, and for update equations, $S(i, j)$, along with the alignment path traceback. It is to be noted that we use $d = 128$ as the dimension of the Gaussian posteriorgrams.

	Time (mins)
$d(i, j)$	66.15
$S(i, j)$ + Path traceback	14.08

From Table 3, it can be seen that the distance computation occupies 82.44% of the total search time. Thus, we are motivated to use GPUs for fast distance computation. A more detailed description of GPU implementation of NS-DTW is provided in Section 5.

5 GPU Accelerated NS-DTW

In this Section, we use NVIDIA CUDA framework to exploit parallel processing for fast QbE-STD search. CUDA follows a single instruction multiple data (SIMD) paradigm where the GPU cores executes the same instruction on different portions of the data (Nickolls et al., 2008). DTW-based variants perform the update equations in a sequential manner and thus an issue for GPU implementations (Zhang et al., 2012), (Sart et al., 2010). A solution to overcome the problem is to parallelize a part of the computation such as the distance calculation for a search speedup. In this paper, we use NVIDIA GT 610 graphic card with 48 cores and a GPU memory of 2048 MB.

CUDA is known for fast matrix operations such as multiplication and thus we exploit its use for distance computation. To exploit the computing power of the GPU, we use matrix multiplication of the complete reference (R) and query (Q) feature vectors. Let $\hat{S}$ represent an $m \times n$ matrix such that $\hat{S}(i, j) = d(i, j)$. $\hat{S}$ can be obtained as follows: $\hat{S} = -log(R^T Q)$, where R and Q are the reference and query feature vectors (as described in Eq. (1) and Eq. (2)). It is to be noted that $R^T Q$ represents an $m \times n$ matrix and $log(R^T Q)$ performs a logarithmic operation on all the $m \times n$ elements in the matrix.

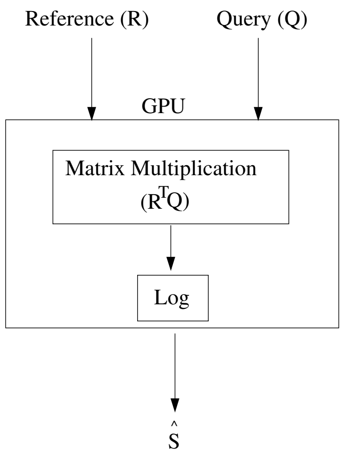

Figure 1: A general block diagram of the distance computation in a GPU for NS-DTW.

The GPU implementation of the NS-DTW is as follows:

1. CPU copies the reference and query Gaussian posteriorgrams into the GPU memory.

2. To initialize $\hat{S}$, CUDA kernels (or functions) are used and is obtained in a two step process: (a) Firstly, the dot product is performed using matrix multiplication given by $R^T Q$, and (b) Then the log operation is performed. Fig. 1 shows a general block diagram of the operations performed using GPU to obtain $\hat{S}$.

3. The CPU, then copies the $\hat{S}$ into the system memory (RAM) and then performs the update equations as described in Eq. (4). It is to be noted that the distance of each of the query and reference Gaussian posteriorgrams have been computed in *Step 2* and thus we use $\hat{S}(i, j)$ instead of $d(i, j)$ in the update equations given by Eq. (4).

5.1 Use of Batch Processing for Search

To maximize the number of parallel computations on the GPU we use batch processing wherein NS-DTW is performed on a query Q and the entire database of references pooled to a single sequence of Gaussian posteriorgrams. This single sequence of Gaussian posteriorgrams is referred to as a reference batch (R_b) and is given as follows:

$$\begin{aligned} R_b \;=\; & [R^1_{d\times m_1}, R^2_{d\times m_2}, \ldots, \\ & \ldots, R^k_{d\times m_k}, \ldots, R^L_{d\times m_L}]_{d\times M}, \end{aligned} \quad (5)$$

where $R^k_{d \times m_k}$ is a matrix of size $d \times m_k$ which represents the k^{th} reference containing m_k sequence of Gaussian posteriorgrams of dimension d. The size of R_b is $d \times M$, where $M = \sum_{k=1}^{L} m_k$.

On obtaining R_b, $\hat{S}$ is then computed as follows: $\hat{S} = -log(R_b^T Q)$. If R_b is very large we split the data into a smaller batches and processes a single batch at a time. On computing the similarity matrix S, we select 500 best alignment score indices and perform a path trace back to obtain the start time stamps of the possible search hits. From the dev dataset, we have observed that there were no queries which are present in more than 500 spoken audio and thus we select 500 best alignment score indices for detection.

5.2 Comparison of Search Time: GPU vs CPU

Fig. 2 shows the search speed of NS-DTW using CPU and GPU cores. It is to be noted that we use batch processing on the GPU for QbE-STD search. From Fig. 2, it can be seen that the GPU implementation is faster than that of the CPU and the search speed is independent (an approximation) of the dimension (d).

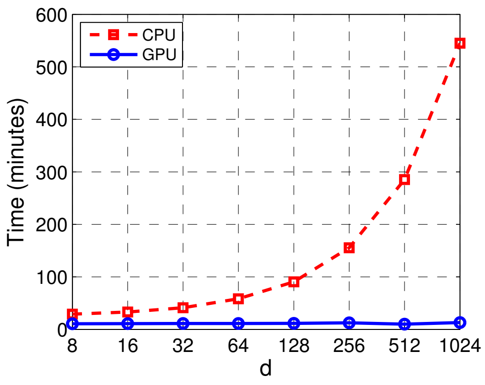

Figure 2: NS-DTW search time on dev data by varying the dimensions of the Gaussian posteriorgrams using CPU and GPU cores.

To summarize the search performance and speed of NS-DTW, in Table 4 we show MP, FAP, MTWV and speedup of QbE-STD search for $d \geq 128$ using a GPU. We define speedup as the ratio of NS-DTW search time on CPU and to that of the search time obtained using a GPU.

From Table 4, it can be seen that there is an improvement in the search performance (MTWV) for d = 128, 256, 512, 1024 as compared to the MTWVs as shown in Table 2. It is to be noted that, to compute the values in Table 2 we have selected 5 best alignment scores from each reference R (as described in Section 3) and to compute the values in Table 4, we have selected 500 best alignment scores from the reference batch R_b (as described in Section 5.1). The results indicate a decrease in the miss probability in Table 4 as compared to that of Table 2 for d = 128, 256, 512, 1024 and thus an improvement in the search performance (MTWV).

Table 4: Miss probability (MP), false alarm probability (FAP), maximum term weighted value (MTWV) and search speed on dev dataset using a GPU for d = 128, 256, 512, 1024.

d	**MP**	**FAP** (10^{-2})	**MTWV**	**Speedup**
128	0.363	1.214	0.450	6.91x
256	0.393	1.010	0.452	11.48x
512	0.359	1.078	0.475	27.06x
1024	0.334	1.149	0.489	40.80x

5.3 Use of Feature Reduction for Search

In this Section, we intend to further improve the search time by modifying the NS-DTW. In this paper, we reduce the query and reference Gaussian posteriorgram vectors before performing search. In this method, we average the successive Gaussian posteriorgrams to reduce the length of the spoken audio and the spoken query.

Consider a reduction factor $\alpha \in \mathbb{N}$. Let $\hat{Q}$, $\hat{R}$ be the sequence of reduced set of feature vectors representing the query and reference. $\hat{Q}$ and $\hat{R}$ are obtained as follows:

$$\begin{aligned} \hat{Q} &= [\hat{\mathbf{q}}_\mathbf{1}, \hat{\mathbf{q}}_\mathbf{2}, \ldots, \hat{\mathbf{q}}_\mathbf{i}, \ldots, \hat{\mathbf{q}}_{\hat{\mathbf{n}}}]_{d \times \hat{n}} \\ \hat{R} &= [\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{j}, \ldots, \hat{\mathbf{u}}_{\hat{\mathbf{m}}}]_{d \times \hat{m}}, \end{aligned}$$

where $\hat{n} = \frac{n}{\alpha}$, $\hat{m} = \frac{m}{\alpha}$ and

$$\begin{aligned} \hat{\mathbf{q}}_\mathbf{i} &= \frac{1}{\alpha} \sum_{k=1}^{\alpha} \mathbf{q}_{(\mathbf{i}-\mathbf{1})\alpha+\mathbf{k}} \\ \hat{\mathbf{u}}_\mathbf{j} &= \frac{1}{\alpha} \sum_{j=1}^{\alpha} \mathbf{u}_{(\mathbf{j}-\mathbf{1})\alpha+\mathbf{k}} \end{aligned}$$

Given a reduction factor $\alpha \in \mathbb{N}$, a window of size α is considered over the posteriorgram features and a mean is computed. The window is then shifted by α and another mean vector is computed.

Table 5: Maximum term weighted value (MTWV), speedup and memory usage (%) obtained on dev and eval datasets for $\alpha = 1, 2, 3$ and $d = 1024$ using a GPU. It is to be noted that for $\alpha = 1$, the GPU memory is not sufficient to load the whole reference dev and eval database. Thus, for $\alpha = 1$, the reference dev and eval datasets are partitioned into 2 and 3 smaller batches respectively.

α	**dev**			**eval**		
	MTWV	Speedup	Memory	MTWV	Speedup	Memory
1	0.489	40.80x	83.15%	0.469	41.37x	63.46%
2	0.474	116.37x	88.63%	0.453	117.34	91.03%
3	0.462	211.12x	61.73%	0.4353	217.60x	63.31%

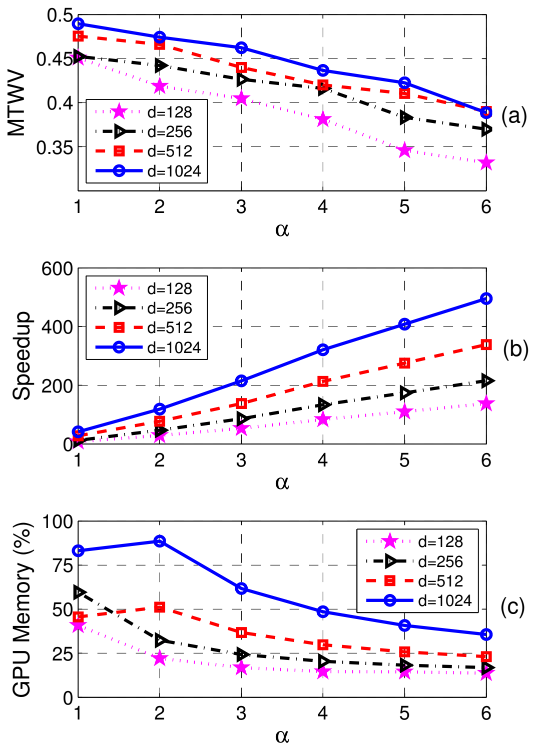

Figure 3: (a) MTWV, (b) Speedup, and (c) GPU memory usage (in percentage) obtained using $d = 128, 256, 512, 1024$ for various values of α on dev dataset. It is to be noted that $\alpha = 1$ represents the NS-DTW without feature reduction.

A more detailed description about the algorithm is provided in (Mantena et al., 2014).

In Fig. 3, we show the MTWV, speedup, and GPU memory usage (in percentage) obtained using $d = 128, 256, 512, 1024$ for $\alpha = 1, 2, 3, 4, 5, 6$ on dev dataset. It is to be noted that $\alpha = 1$ represents the NS-DTW without feature reduction. From Fig. 3, it can be seen that the search performance decreases with an increase in α but there is an improvement in speedup and GPU memory usage. For $d = 512, 1024$, GPU memory is not sufficient to load the whole reference batch and so it is partitioned to 2 smaller batches for search. Thus, for $d = 512, 1024$, the memory usage is lower for $\alpha = 1$ as compared to that of $\alpha = 2$. From Fig. 3, it can be also be seen that by using feature reduction one can use higher dimensions of Gaussian posteriorgrams such as $d = 1024$ and thus enable searching a query in large reference files.

To summarize the QbE-STD performance, in Table 5, we show the MTWV, speedup and memory usage (%) on dev and eval datasets for $\alpha = 1, 2, 3$ and $d = 1024$. It is to be noted that on increasing the α the MTWV decreases and thus resulting in a poor search performance (as shown in Fig. 3). From Table 5, it can be seen that there is a good improvement in the speedup for $\alpha = 2$ resulting in a marginal drop in search performance. Thus, with the use of a GPU and feature reduction for $\alpha = 2$ we could attain a speedup of about 100x and thereby enable QbE-STD search in real time.

6 Conclusions

In this paper, we used a graphical processing unit (GPU) and improved the computation time of the distance calculation of non-segmental dynamic time warping (NS-DTW) algorithm in the context query-by-example spoken term detection (QbE-STD) search. We have shown with experimental results that the NS-DTW search speed can be made independent (an approximation) of the dimension of the Gaussian posteriorgram using a GPU. We have also used a faster method of searching by reducing the length of the spoken audio and the spoken query. The reduction of the feature vectors was done via arithmetic mean and it is shown that for a reduction factor of $\alpha = 2$, there is an improvement in the search speed of about 100x with a marginal drop in search performance using 1024 dimensional Gaussian posteriorgrams.

7 Acknowledgements

We would like to thank P. J. Narayanan, Kishore Kothapalli, Manoj Maramreddy and Sivanand Achanta from IIIT-H, India for their feedback on the GPU implementations of NS-DTW. We would also like to thank TCS, India for partially supporting Gautam's PhD fellowship.

References

X. Anguera and M. Ferrarons. 2013. Memory efficient subsequence DTW for query-by-example spoken term detection. In *Proc. of ICME*.

X. Anguera. 2012. Speaker independent discriminant feature extraction for acoustic pattern-matching. In *Proc. of ICASSP*, pages 485–488.

E. Barnard, M. H. Davel, and C. J. V. Heerden. 2009. ASR corpus design for resource-scarce languages. In *Proc. of INTERSPEECH*, pages 2847–2850.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *in Proc. of the Python for Scientific Computing Conference*, June. Oral Presentation.

P. Cardinal, P. Dumouchel, G. Boulianne, and M. Comeau. 2008. GPU accelerated acoustic likelihood computations. In *Proc. of INTERSPEECH*, pages 964–967.

J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington. 2007. Results of the 2006 spoken term detection evaluation. In *Proc. of Workshop on Searching Spontaneous Conversational Speech*, pages 45–50.

S. Ganapathy, S. Thomas, and H. Hermansky. 2010. Temporal envelope compensation for robust phoneme recognition using modulation spectrum. *Journal of Acoustical Society of America*, 128:3769–3780.

V. Gupta, J. Ajmera, A., and A. Verma. 2011. A language independent approach to audio search. In *Proc. of INTERSPEECH*, pages 1125–1128.

T. J. Hazen, W. Shen, and C. White. 2009. Query-by-example spoken term detection using phonetic posteriorgram templates. In *Proc. of ASRU*, pages 421–426.

G. Mantena, S. Achanta, and K. Prahallad. 2014. Query-by-example spoken term detection using frequency domain linear prediction and non-segmental dynamic time warping. *IEEE/ACM Trans. on Audio, Speech and Lang. Processing*, 22(5):946–955, May.

F. Metze, E. Barnard, M. H. Davel, C. J. V. Heerden, X. Anguera, G. Gravier, and N. Rajput. 2012. The spoken web search task. In *MediaEval*.

D. R. H. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish. 2007. Rapid and accurate spoken term detection. In *Proc. of INTERSPEECH*, pages 314–317.

J. Nickolls, I. Buck, M. Garland, and K. Skadron. 2008. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. 2011. The Kaldi speech recognition toolkit. In *in Proc. of ASRU*. IEEE Signal Processing Society, December.

M. Saraclar and R. Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *Proc. of HLT-NAACL*, pages 129–136.

D. Sart, A. Mueen, W. Najjar, E. Keogh, and V. Niennattrakul. 2010. Accelerating dynamic time warping subsequence search with GPUs and FPGAs. In *in Proc. of International Conference on Data Mining (ICDM)*, pages 1001–1006, Dec.

Y. Shi, F. Seide, and F. K. Soong. 2008. GPU-accelerated gaussian clustering for fMPE discriminative training. In *Proc. of INTERSPEECH*, pages 944–947.

I. Szöke, M. Fapso, L. Burget, and J. Cernocky. 2008. Hybrid word-subword decoding for spoken term detection. In *Workshop on Searching Spontaneous Conversational Speech*, pages 4–11.

S. Thomas, S. Ganapathy, and H. Hermansky. 2008. Recognition of reverberant speech using frequency domain linear prediction. *IEEE Signal Processing Letters*, 15:681 –684.

Y. Zhang and J. R. Glass. 2009. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In *Proc. of ASRU*, pages 398–403.

Y. Zhang and J. Glass. 2011. An inner-product lower-bound estimate for dynamic time warping. In *Proc. of ICASSP*, pages 5660–5663.

Y. Zhang, K. Adl, and J. Glass. 2012. Fast spoken query detection using lower-bound dynamic time warping on graphical processing units. In *in Proc. of ICASSP*, pages 5173–5176, March.