PRE-FILTERED DYNAMIC TIME WARPING FOR POSTERIORGRAM BASED KEYWORD SEARCH

Gozde Cetinkaya, Batuhan Gundogdu, Murat Saraclar

Bogazici University, Electrical and Electronics Engineering Department, Bebek, Istanbul, 34342, Turkey
{gozde.cetinkaya, batuhan.gundogdu, murat.saraclar}@boun.edu.tr

ABSTRACT

In this study, we present a pre-filtering method for dynamic time warping (DTW) to improve the efficiency of a posteriorgram based keyword search (KWS) system. The ultimate aim is to improve the performance of a large vocabulary continuous speech recognition (LVCSR) based KWS system using the posteriorgram based KWS approach. We use phonetic posteriorgrams to represent the audio data and generate average posteriorgrams to represent the given text queries. The DTW algorithm is used to determine the optimal alignment between the posteriorgrams of the audio data and the queries. Since DTW has quadratic complexity, it can be relatively inefficient for keyword search. Our main contribution is to reduce this complexity by pre-filtering based on a vector space representation of the two posteriorgrams without any degradation in performance. Experimental results show that our system reduces the complexity and when combined with the baseline LVCSR based KWS system, it improves the performance both for the out-of-vocabulary (OOV) queries and the in-vocabulary (IV) queries.

Index Terms— keyword search, posteriorgram, dynamic time warping

1. INTRODUCTION

The aim of keyword search (KWS) is to detect the time interval of the occurrence of the written queries within unsegmented and untranscribed audio data. The state-of-the-art KWS systems use large vocabulary continuous speech recognition (LVCSR) systems which in some cases suffer from out-of-vocabulary (OOV) and under-represented terms [1, 2]. Typically, subword lattices [3] or confusion models [4, 5, 6] are used to overcome the OOV problem. In this study, phone posteriorgrams, which are sequences of posterior probability vectors, are used to represent the data. Posteriorgrams represent the data in a speaker independent statistical form and contain phonetic information [7]. This representation is extensively used in query-by-example (QbE) spoken term detection applications [7, 8, 9] and for instance, in [8], along with dynamic time warping.

Dynamic time warping is a technique that determines the optimal alignment between two sequences by stretching or compressing the time axis and warps one sequence onto another in a nonlinear fashion. This warping can then be a basis to compute the similarity between these sequences. Dynamic time warping is a widespread method used in various disciplines [10], including speech recognition, robotics and data mining, that study time series and need to determine their similarity. However, to align the two time series, the DTW algorithm has O(xy) complexity where x and y are the lengths of the time series X and Y respectively [11]. For the applications that have long sequences, this complexity constrains the effectiveness of the approach.

Aside from parallelization, many approaches have been proposed in order to speed up DTW by means of constraints, indexing and data abstraction [11]. Constraints on DTW limit the number of cells of the cost matrix and DTW computes the optimal path throughout the constraint cells. Indexing reduces the number of times DTW must be run by using lower-bound estimation approaches [12, 13]. The idea is to form a bounding envelope that contains the sequence. Then, an approximation of the DTW distance is computed to build an index for search. Data abstraction methods reduce the representation of the data by reducing the size of the time series to make the cost matrix more manageable [14]. Also, computationally less expensive approaches are proposed, where the main idea is to divide the search space into smaller parts and to calculate the DTW block by block. In [15], the search space is divided into parts that have equal size with the length of the query. To speed up the algorithm, DTW is computed if the sum of diagonal elements in each block is lower than a predefined distance threshold. In [16], segment-based DTW is proposed. Segments are generated by a clustering algorithm which starts with each frame being a segment and in each iteration the two neighboring segments that yield the minimum increment of total variance are merged and DTW is performed using these segments.

In this study, the DTW algorithm is carried out on the full resolution cost matrix without any reduction of the time series. The size of the search space is decreased by a pre-filtering step, which eliminates improbable parts according to the similarity between the query and the document. Thus,

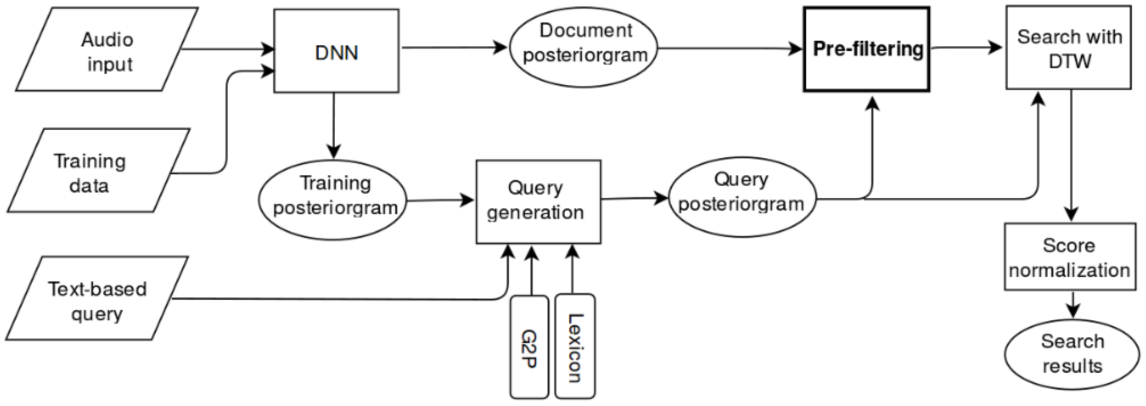

Fig. 1. Block diagram of the proposed system

our approach avoids the mass calculation with overall data by removing unlikely matches and the DTW algorithm is applied only on possible candidates.

The remainder of this paper is organized as follows: Section 2 introduces the proposed system. Section 3 provides the experimental setup. The results are given in Section 4 and the paper is concluded in Section 5.

2. METHODS

In this study, the baseline KWS system uses deep neural network (DNN) acoustic models for LVCSR [17]. These acoustic models are used to generate state level posteriorgrams which are converted to phone level posteriorgrams. We use these phonetic posteriorgrams where each class corresponds to a distinct phone as our input feature vectors. Since the queries and the data should be in the same form for the search process, posteriorgrams for the given text queries need to be generated. For in-vocabulary (IV) queries, a pronunciation lexicon is used to convert the query into a phone sequence. For the out-of-vocabulary (OOV) queries grapheme-to-phoneme (G2P) conversion is used [18]. The query posteriorgrams are generated by using the average posterior vectors and the average durations for each phone in the query. The average phone posterior vectors and average phone durations are determined using the training data alignments and posteriorgrams. In the following part of this section, generation of the average vectors and filtering step are presented.

2.1. Generation of average vectors

Since, in classic DTW, the query is searched in the whole document posteriorgram, for each query the corresponding cost matrix is generated recurrently. Thus, this grand matrix calculation happens for every query and the algorithm searches the whole document posteriorgram including unlikely candidates for the query. Our aim is to eliminate these unlikely candidates. The basic idea of our approach is to divide the document posteriorgram matrix into segments, or submatrices, and turn them into average vectors, which are used to determine the possible candidates for query matching. Thus, DTW search will be performed on the possible candidates instead of the entire document.

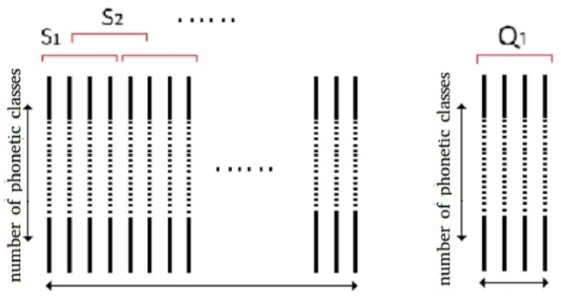

Fig. 2. Illustration for average vector generation

The posteriorgrams of the document and the query are matrices whose rows and columns correspond to the phonetic classes and the time frames, respectively. In Figure 2, an illustration for average vector generation is shown. As seen in the figure, the document posteriorgram is divided into segments. $\mathbf{S}_1$ and $\mathbf{S}_2$ denote the first and second average vectors for the document segments. The beginning point of the query may not be found if it is not included within the segments. To avoid this problem, the document posteriorgram is divided into overlapping segments with shift. The algorithm generates the average vectors for each submatrix representing a segment by summing the entries for each phonetic class and normalizing the sum by the number of frames. Both query

Table 1. Configurations for the average vector generation

Config.	Segment Size	Shift
P	variable (query-sized)	150 msec
F	fixed (average frame number of all queries= 60)	150 msec

and segments are calculated as average vectors as:

$$S_{kj} = \frac{1}{f}\sum_{i=1}^{f} s(i,j), \qquad j = 1, ..., m$$

$$Q_j = \frac{1}{y}\sum_{i=1}^{y} q(i,j), \qquad j = 1, ..., m$$

where $\mathbf{S}_k$ denotes the average vector of the k^{th} segment, $\mathbf{Q}$ is the average vector for the query, f is the number of the frames of the segment, y is the number of the frames of the query and m is the number of the phonetic classes. Each $\mathbf{S}_k$ average vector is a possible candidate for matching that gets filtered.

Two different configurations for the segment size are used to generate average vectors, which are shown in Table 1. The document posteriorgram is divided into either query-sized (variable) or fixed-size segments, which are denoted as P and F, respectively. The number of the frames for the fixed-size is taken to be the average number of frames of all queries, which is 60 for our task. The shift amount is 150 msec for both configurations.

2.2. Pre-filtering

After the document average vectors are calculated, cosine distance is used for the distance measure to find the similarity between the document average vectors and the query average vectors. Note that since the average vectors have non-negative entries the similarity will always be non-negative.

$$\text{sim}(\mathbf{S}_k, \mathbf{Q}) = \frac{\sum_{j=1}^{m} S_{kj} \times Q_j}{\sqrt{\sum_{j=1}^{m}(S_{kj})^2} \times \sqrt{\sum_{i=j}^{m}(Q_j)^2}} \tag{1}$$

For each query, after we calculate the scores of all possible matches as in Equation 1, filtering is implemented by comparing the scores with a threshold value that specifies the amount of filtering. The lower the threshold value, the more candidate matches are searched by DTW. For a coarse filtering, a low threshold value is sufficient whereas for a fine filtering, the threshold value should be higher. The algorithm stores the beginning points of segments, thus, after filtering segments are recomposed. In order to avoid boundary problems, the segments are extended by 50 msec on both sides before DTW.

For the classic DTW, space complexity is O(xy) where y is the number of frames in the query and x is the number of frames in the document. For a query search, the x-by-y cost matrix should be calculated to perform DTW. In our system, DTW is performed only on likely matches and by pre-filtering, the complexity is reduced to O(ny^2) where n is the number of pre-filtered possible matches. The block diagram of the pre-filtered DTW KWS system can be seen in Figure 1.

3. EXPERIMENTAL SETUP

For the experiments, the IARPA Babel Turkish limited language pack (LimitedLP) dataset containing conversational telephone speech is used. For KWS evaluation, the 307 development queries, 88 (28.7%) of which are OOV, are used.

Since one of the aims of this work is to improve the baseline LVCSR lattice based KWS system by combining with the proposed system, the baseline system should be introduced. The baseline system uses LVCSR to generate lattices, which are processed by lattice indexing [19] for IV queries. The lattices are converted from individual weighted finite state transducers (WFST) to a single generalized factor transducer structure. The baseline uses proxy keywords to cope with OOV queries. The basic idea is to find acoustically similar IV terms for the OOV terms and use these IV terms as proxy keywords [20]. For the baseline LVCSR based KWS system, the Babel KWS setup in the Kaldi toolkit is used[1]. The system uses a DNN with p-norm activations [17].

For the proposed system, the audio data are processed with the same DNN and a state level posteriorgram is produced. To obtain the phone posteriorgram, the posterior values of states corresponding to each phone are accumulated which yields a 43 dimensional posteriorgram. The text based queries are modeled by using average query modelling, which uses the phone averages obtained from the training data. For the duration of each phone, average phone durations are used. Hence in the model, the average posterior vectors are repeated with the number of frames corresponding to the average duration of the pertinent phone.

For KWS evaluation, a term-weighted value (TWV) is used:

$$\text{TWV}(\theta) = 1 - \frac{1}{|Q|}\sum_{q \epsilon Q} P_{\text{miss}}(q,\theta) + \beta P_{\text{FA}}(q,\theta) \tag{2}$$

where Q denotes the set of queries, $|Q|$ denotes the number of queries in Q, $\text{P}_{\text{miss}}(q,\theta)$ and $\text{P}_{\text{FA}}(q,\theta)$ show the miss and

[1]http://svn.code.sf.net/p/kaldi/code/trunk/egs/babel/s5c

false alarm probabilities for a query q when a global threshold θ is applied to the search scores. β is a constant (999.9) determining the relative cost of miss and false alarms. The TWV gives 1.0 for the maximum value corresponding to correct detection of all the queries with no false alarm. The TWV can get negative values and for the empty hit list, TWV is 0. Maximum TWV (MTWV), optimal TWV (OTWV) and supremum TWV (STWV) are used in this study. MTWV is the maximum TWV which obtained from all possible threshold values. The OTWV is calculated using a separate threshold for each query, and STWV is calculated by taking only the correct decisions into account [21].

4. RESULTS AND DISCUSSION

In this section, the results from the LVCSR lattice based system and the proposed method are presented individually. Merged results are also given by adding and normalizing the scores of the corresponding hits from each system to perform combination.

In Figure 3, the MTWV of IV, OOV and all queries with respect to the run time is shown, for the query-sized average vectors case without any combination (P). The run time of no-filtering DTW is approximately 45 hours and corresponds to 1.0 in Figure 3. For the experiments, 8 different threshold values are used. The filtering time is included in the run time and is 19 minutes for all threshold values. By setting the threshold appropriately the run time can be reduced by approximately 12 times without significant loss in performance. The fixed-size average vectors yield similar performance.

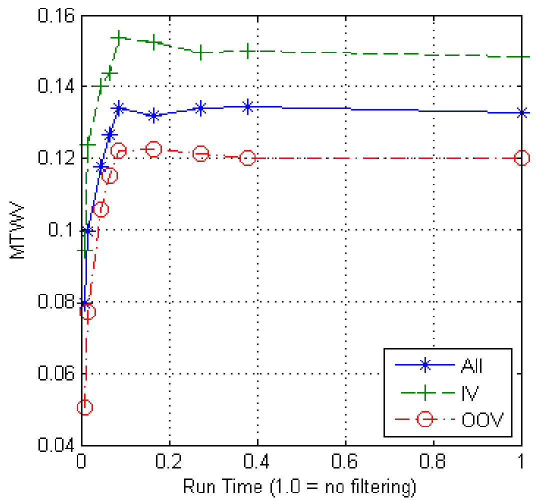

Fig. 3. MTWV as a function of run time for P

In Figure 4, the comparison of run times for two different configurations to generate average vectors is shown. For both cases, it can be seen that as the threshold increases, the number of candidate matches decreases. The generation of fixed-size average vectors requires less than 4 seconds, since the document posteriorgram is divided into segments only once before the DTW.

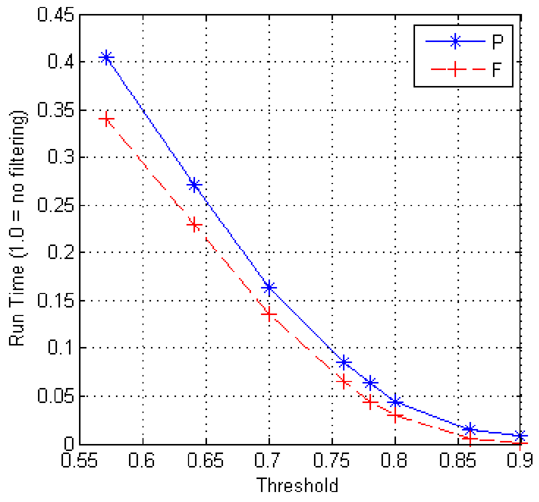

Fig. 4. Run time as a function of threshold for P and F

For the fixed-size segment approach, the average number of frames of all queries (60) is used to generate average vectors, which yields similar performance to the query-sized approach. Figure 5 shows the dependence of the performance on the segment size. The best result is obtained when the number of frames is between 60-100. The MTWV of IV, OOV and all queries reduce as the number of frames in the segment is either decreased or increased considerably.

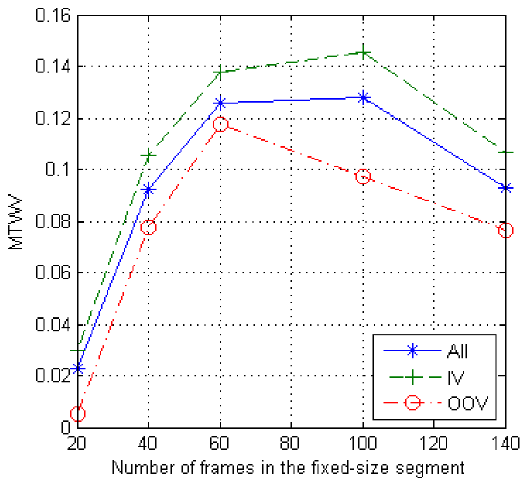

Fig. 5. The dependence of the performance on the segment size

Table 2. MTWV, OTWV and STWV results for LVCSR lattice based system (L) and merged (L+P, L+F) systems

System	MTWV			OTWV			STWV		
	All	IV	OOV	All	IV	OOV	All	IV	OOV
L	0.3745	0.4500	0.1887	0.4157	0.4896	0.2240	0.6624	0.7567	0.4220
$L+P$	0.3987	0.4603	0.2655	**0.4909**	**0.5469**	0.3482	**0.7495**	**0.8105**	**0.5938**
$L+F$	**0.3991**	**0.4619**	**0.2662**	0.4894	0.5447	**0.3485**	0.7476	0.8079	**0.5938**

In order to merge scores of the baseline system and the proposed system, the scores of the corresponding hits from different systems are summed and then normalized by the number of systems that contributed to the sum, which is a similar method to using the CombMNZ method described in [22]. Table 2 shows MTWV, OTWV and STWV results, where L denotes the LVCSR lattice based system. L+P and L+F represent the combination of the baseline and the proposed method with two different configurations for the lowest threshold. For the merged systems, MTWV scores of the fixed-size case are slightly higher than the MTWV scores of the query-sized case. The proposed system improves the IV MTWV from 0.3745 to 0.3991 and the OOV MTWV from 0.1887 to 0.2662. Similar gains are observed in OTWV and STWV for both IV and OOV queries.

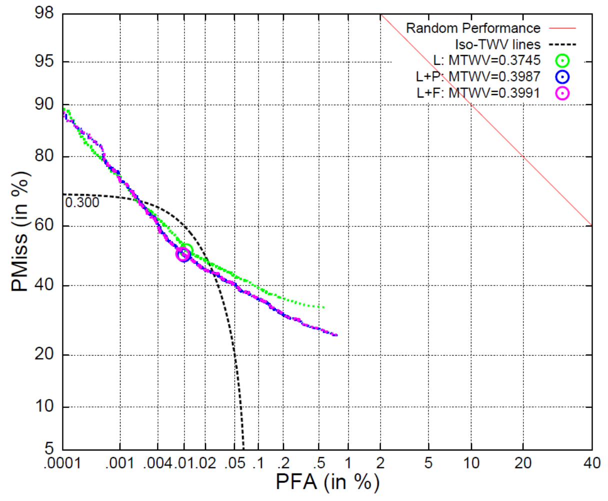

Fig. 6. DET curves of the baseline (L) and the combined systems (L+P, L+F)

In Figure 6, detection error trade-off (DET) curves are shown, for the LVCSR lattice based system and the combination of the baseline and the proposed method. The DET curve is the plot of the miss rate versus the false alarm rate as the threshold is varied. The smooth red line on the upper right corner shows the random performance. For TWV<0.30 , the DET curve should be under the Iso-TWV line in the bottom left corner. The green line represents the DET curve of the baseline system. The blue and magenta lines are the DET

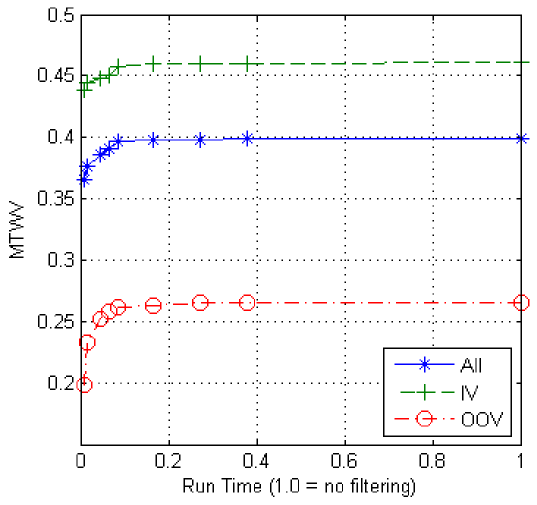

Fig. 7. MTWV as a function of run time for L+P

curves of the combinations of the baseline and the proposed systems with query-sized and fixed-size average vectors, respectively. It can be seen that the performances of the two merged systems are very similar and they are both better than the baseline system.

In Figure 7, the results for the combination of the proposed system with the baseline for the MTWV of IV, OOV and all queries with respect to the run time is shown for the query-sized average vectors case. For the OOV queries, the performance of the baseline is increased even with the lowest threshold (22 minutes). After the third threshold, the MTWV for the combined system of IV, OOV and all queries are higher than the baseline system scores, given in the first row of the Table 2. A similar behavior is observed for the fixed-size average vectors case.

The experimental results show that the individual scores of the proposed pre-filtered DTW KWS system and the combined results with the baseline are higher than (16% relative for individual, 9.2% relative for merged) the results of the previous work [23], which uses a similar KWS scenario with a different DTW method called subsequence DTW and without any pre-filtering.

A related approach is proposed in [24], where a two-pass DTW is used to integrate segment-based DTW [16] and classical frame-based DTW, where the segments contain frames with the size roughly of a phoneme. First, the segment-based DTW locates the possible regions within each document. Then, in the second pass frame-based DTW is used to recalculate the distance between the possible regions. Although the approaches are similar, the query-sized and fixed-size windows used in our study are much longer than the phoneme-sized segments used in [24].

5. CONCLUSIONS

In this study, we proposed a pre-filtering method for DTW based KWS using phone posteriorgrams obtained by DNN acoustic models. We extract submatrices from the document posteriorgram and turn them into average vectors. These vectors are peaky and by careful selection of the segment size, the averaged posteriorgrams retain sufficient information for the search. To generate the average vectors, we use two different configurations. The document posteriorgram is divided into query-sized segments with 150 msec shift and fixed-size segments with 150 msec shift. By filtering some of these average vectors, the query search is performed only on possible matches. Thus, the complexity of the DTW is decreased. The individual results show that the run time for fixed-size average vectors is shorter than the run time for query-sized average vectors. Finally, we merged the results of our method with the LVCSR lattice based system and observed improvements in MTWV for both IV (2.6% relative) and OOV (41.1% relative) queries, resulting in an overall improvement (6.6% relative).

6. ACKNOWLEDGEMENTS

We are grateful to Leda Sari for her kind assistance with the baseline system and posteriorgram generation.
This study uses the IARPA Babel Program base period language collection release babel105b-v0.4, and is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. REFERENCES

[1] Lin-shan Lee, James Glass, Hung-yi Lee, and Chun-an Chan, “Spoken content retrieval beyond cascading speech recognition with text retrieval,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1389–1420, 2015.

[2] Ciprian Chelba, Timothy J Hazen, and Murat Saraçlar, “Retrieval and browsing of spoken content,” *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 39–49, 2008.

[3] Jonathan Mamou, Bhuvana Ramabhadran, and Olivier Siohan, “Vocabulary independent spoken term detection,” in *Proceedings of ACM SIGIR*, 2007, pp. 615–622.

[4] Kenney Ng and Victor W Zue, “Subword-based approaches for spoken document retrieval,” *Speech Communication*, vol. 32, no. 3, pp. 157–186, 2000.

[5] Panagiota Karanasou, Lukas Burget, Dimitra Vergyri, Murat Akbacak, and Arindam Mandal, “Discriminatively trained phoneme confusion model for keyword spotting.,” in *Proceedings of Interspeech*, 2012.

[6] Murat Saraclar, Abhinav Sethy, Bhuvana Ramabhadran, Lidia Mangu, Jia Cui, Xiaodong Cui, Brian Kingsbury, and Jonathan Mamou, “An empirical study of confusion modeling in keyword search for low resource languages,” in *Proceedings of IEEE ASRU*, 2013, pp. 464–469.

[7] Yaodong Zhang, *Unsupervised speech processing with applications to query-by-example spoken term detection*, Ph.D. thesis, Massachusetts Institute of Technology, 2013.

[8] Timothy J Hazen, Wade Shen, and Christopher White, “Query-by-example spoken term detection using phonetic posteriorgram templates,” in *Proceedings of IEEE ASRU*, 2009, pp. 421–426.

[9] Yaodong Zhang and James R Glass, “Unsupervised spoken keyword spotting via segmental DTW on gaussian posteriorgrams,” in *Proceedings of IEEE ASRU*, 2009, pp. 398–403.

[10] Eamonn J Keogh and Michael J Pazzani, “Derivative dynamic time warping.,” in *Proceedings of SIAM Data Mining*, 2001, vol. 1, pp. 5–7.

[11] Stan Salvador and Philip Chan, “FastDTW: Toward accurate dynamic time warping in linear time and space,” in *Proceedings of Mining Temporal and Sequential Data*, 2004.

[12] Eamonn Keogh and Chotirat Ann Ratanamahatana, “Exact indexing of dynamic time warping,” *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[13] Yaodong Zhang and James R Glass, "An inner-product lower-bound estimate for dynamic time warping," in *Proceedings of IEEE ICASSP*, 2011, pp. 5660–5663.

[14] Eamonn J Keogh and Michael J Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of ACM SIGKDD*. ACM, 2000, pp. 285–289.

[15] Jozef Vavrek, Peter Viszlay, Eva Kiktová, Martin Lojka, Jozef Juhár, and Anton Čižmár, "Query-by-example retrieval via fast sequential dynamic time warping algorithm," in *Proceedings of IEEE TSP*, 2015, pp. 1–5.

[16] Chun-an Chan and Lin-shan Lee, "Unsupervised spoken-term detection with spoken queries using segment-based dynamic time warping," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[17] Jan Trmal, Guoguo Chen, Dan Povey, Sanjeev Khudanpur, Pegah Ghahremani, Xiaohui Zhang, Vimal Manohar, Chunxi Liu, Aren Jansen, Dietrich Klakow, et al., "A keyword search system using open source software," in *Proceedings of IEEE SLT*, 2014, pp. 530–535.

[18] Maximilian Bisani and Hermann Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[19] Doğan Can and Murat Saraclar, "Lattice indexing for spoken term detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[20] Guoguo Chen, Ozgur Yilmaz, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proceedings of IEEE ASRU*, 2013, pp. 416–421.

[21] *KWS14 keyword search evaluation plan*, http://www.nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf.

[22] Jonathan Mamou, Jia Cui, Xiaodong Cui, Mark JF Gales, Brian Kingsbury, Kate Knill, Lidia Mangu, David Nolden, Michael Picheny, Bhuvana Ramabhadran, et al., "System combination and score normalization for spoken term detection," in *Proceedings of IEEE ICASSP*, 2013, pp. 8272–8276.

[23] Leda Sarı, Batuhan Gündoğdu, and Murat Saraçlar, "Fusion of LVCSR and posteriorgram based keyword search," in *Proceedings of Interspeech*, 2015.

[24] Chun-an Chan and Lin-shan Lee, "Integrating frame-based and segment-based dynamic time warping for unsupervised spoken term detection with spoken queries," in *Proceedings of IEEE ICASSP*, 2011, pp. 5652–5655.