The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages
29-31 August 2018, Gurugram, India

ASR-free CNN-DTW keyword spotting using multilingual bottleneck features for almost zero-resource languages

Raghav Menon[1], *Herman Kamper*[1], *Emre Yilmaz*[2,3], *John Quinn*[4,5], *Thomas Niesler*[1]

[1]Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa
[2]CLS/CLST, Radboud University, Nijmegen, Netherlands
[3]Department of Electrical and Computer Engineering, National University of Singapore, Singapore
[4]UN Global Pulse, Kampala, Uganda & [5]University of Edinburgh, UK

rmenon@sun.ac.za, kamperh@sun.ac.za, eleey@nus.edu.sg, john.quinn@unglobalpulse.org, trn@sun.ac.za

Abstract

We consider multilingual bottleneck features (BNFs) for nearly zero-resource keyword spotting. This forms part of a United Nations effort using keyword spotting to support humanitarian relief programmes in parts of Africa where languages are severely under-resourced. We use 1920 isolated keywords (40 types, 34 minutes) as exemplars for dynamic time warping (DTW) template matching, which is performed on a much larger body of untranscribed speech. These DTW costs are used as targets for a convolutional neural network (CNN) keyword spotter, giving a much faster system than direct DTW. Here we consider how available data from well-resourced languages can improve this CNN-DTW approach. We show that multilingual BNFs trained on ten languages improve the area under the ROC curve of a CNN-DTW system by 10.9% absolute relative to the MFCC baseline. By combining low-resource DTW-based supervision with information from well-resourced languages, CNN-DTW is a competitive option for low-resource keyword spotting.

Index Terms: relief and developmental monitoring, keyword spotting, convolutional neural networks, dynamic time warping, under-resourced, zero-resource, multilingual bottleneck features.

1. Introduction

Social media has become a popular medium for individuals to express opinions and concerns on issues impacting their lives [1–3]. In countries without adequate internet infrastructure, like Uganda, communities often use phone-in talk shows on local radio stations for the same purpose. In an ongoing project by the United Nations (UN), radio-browsing systems have been developed to monitor such radio shows [4, 5]. These systems are actively and successfully supporting UN relief and developmental programmes. The development of such systems, however, remains dependent on the availability of transcribed speech in the target languages. This dependence has proved to be a key impediment to the rapid deployment of radio-browsing systems in new languages, since skilled annotators proficient in the target languages are hard to find, especially in crisis conditions.

In a conventional keyword spotting system, where the goal is to search through a speech collection for a specified set of keywords, automatic speech recognition (ASR) is typically used to generate lattices which are then searched to predict the presence or absence of keywords [6, 7]. State-of-the-art ASR, however, requires large amounts of transcribed speech audio [8, 9]. In this paper we consider the development of a keyword spotter without such substantial and carefully-prepared data. Instead, we rely only on a small number of isolated repetitions of keywords and a large body of untranscribed data from the target domain. The motivation for this setting is that such isolated keywords should be easier to gather, even in a crisis scenario.

Several studies have attempted ASR-free keyword spotting using a query-by-example (QbyE) retrieval procedure. In QbyE, the search query is provided as audio rather than text. Dynamic time warping (DTW) is typically used to search for instances of the query in a speech collection [10, 11]. As an alternative, several ways of obtaining fixed-dimensional representations of input speech have been considered [12]. Recurrent neural networks (RNNs) [13, 14], autoencoding encoder-decoder RNNs [15, 16], and Siamese convolutional neural networks (CNNs) [17] have all been used to obtain such fixed-dimensional representations, which allow queries and search utterances to be directly compared without alignment. For keyword spotting, a variant of this approach has been used where textual and acoustic inputs are mapped into a shared space [18]. Most of these neural approaches, however, rely on large quantities of training data.

In this paper, we extend the ASR-free keyword spotting approach first presented in [19]. A small seed corpus of isolated spoken keywords is used to perform DTW template matching on a large corpus of untranscribed data from the target domain. The resulting DTW scores are used as targets for training a CNN-based keyword spotter. Hence we take advantage of DTW-based matching—which can be performed with limited data—and combine this with CNN-based searching, giving speed benefits since it does not require alignment. In our previous work, we used speech data only from the target language. Here we consider whether data available for other (potentially well-resourced) languages can be used to improve performance. Specifically, multilingual bottleneck features (BNFs) have been shown to provide improved performance by several authors [20–23]. We investigate whether such multilingual bottleneck feature extractors (trained on completely different languages) can be used to extract features for our target data, thereby improving the overall performance of our CNN-DTW keyword spotting approach.

To perform a thorough analysis of our proposed approach (which requires transcriptions), we use a corpus of South African English. We use BNFs trained on two languages and on ten languages as input features to the CNN-DTW system, and compare these to MFCCs. We also consider features from unsupervised autoencoders, trained on unlabelled datasets from five languages. We show that the 10-language BNFs work best overall, giving results that makes CNN-DTW a viable option for practical use.

2. Radio browsing system

The first radio browsing systems implemented as part of the UN's humanitarian monitoring programmes rely on ASR systems [4]. Human analysts filter speech segments identified by

10.21437/SLTU.2018-38

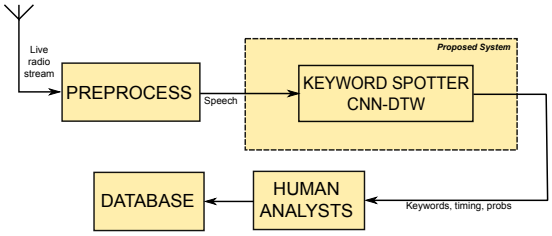


Figure 1: *Radio browsing using the CNN-DTW keyword spotter.*

the system and add these to a searchable database to support decision making.[1] To develop the ASR system, at least a small amount of annotated speech in the target language is required [5]. However, the collection of even a small fully transcribed corpus has proven difficult or impossible in some settings. In recent work, we have therefore proposed an ASR-free keyword spotting system based on CNNs [19]. CNN classifiers typically require a large number of training examples, which are not available in our setting. Instead, we therefore use a small set of recorded isolated keywords, which are then matched against a large collection of untranscribed speech drawn from the target domain using a DTW-based approach. The resulting DTW scores are then used as targets for a CNN. The key is that it is not necessary to know whether or not the keywords do in fact occur in this untranscribed corpus; the CNN is trained simply to emulate the behaviour of the DTW. Since the CNN does not perform any alignment, it is computationally much more efficient than DTW. The resulting CNN-DTW model can therefore be used to efficiently detect the presence of keywords in new input speech. Figure 1 shows the structure of this CNN-DTW radio browsing system.

3. Data

We use the same datasets used in our previous work [19]. As templates, we use a small corpus of isolated utterances of 40 keywords, each spoken twice by 24 South African speakers (12 male and 12 female). This set of 1920 labelled isolated keywords constitutes the only transcribed target-domain data that we use to train our keyword spotter. As untranscribed data, we use a corpus of South African Broadcast News (SABN). This 23-hour corpus consists of a mix of English newsreader speech, interviews, and crossings to reporters, broadcast between 1996 and 2006 [24]. The division of the corpus into different sets is shown in Table 1. The SABN training set is used as our untranscribed data to obtain targets for the CNN. The models then perform keyword spotting on the SABN test set. Since this set is fully transcribed, performance evaluation and analysis is possible. The isolated keywords were recorded under fairly quiet conditions and there was no speaker overlap with the SABN dataset. Hence there is a definite mismatch between the datasets. This is intentional as it reflects the intended operational setting of our system.

Table 1: *The South African Broadcast News (SABN) dataset.*

	Utterances	**Speech (h)**
Train	5231	7.94
Dev	2988	5.37
Test	5226	10.33
Total	13445	23.64

[1] Examples at `http://radio.unglobalpulse.net`.

4. Keyword Spotting Approaches

Here we describe the combined CNN-DTW keyword spotting method. We also use direct DTW and a CNN classifier as baselines, and hence these are also briefly discussed.

4.1. Dynamic time warping (DTW)

In low-resource settings with scarce training data, DTW is an attractive approach, but it can be prohibitively slow since it requires repeated alignment. We make use of a simple DTW implementation in which isolated keywords slide over search audio, with a 3-frame-skip, and a frame-wise comparison is performed while warping the time axis. From this, a normalized per-frame cosine cost is obtained, resulting in a value $c \in [0, 2]$, with 0 indicating a portion of speech that matches the keyword exactly. The presence or absence of the keyword is determined by applying an appropriate threshold to c.

4.2. Convolutional neural network (CNN) classifier

As a baseline, we train a CNN classifier as an end-to-end keyword spotter. This would be typical in high resource settings [17, 25, 26]. We perform supervised training using the 1920 recorded isolated keywords with negative examples drawn randomly from utterances in the SABN training set. For testing, a 60-frame window slides over the test utterances. The presence or absence of keyword is again based on a threshold.[2]

4.3. CNN-DTW keyword spotting

Rather than using labels (as in the CNN classifier above), the CNN-DTW keyword spotting approach uses DTW to generate sufficient training data as targets for a CNN. The CNN-DTW is subsequently employed as the keyword spotter; this is computationally much more efficient than direct DTW. DTW similarity scores are computed between our small set of isolated keywords and a much larger untranscribed dataset, and these scores are subsequently used as targets to train a CNN, as shown in Figure 2. Our contribution here over our previous work [19] is to use multilingual BNFs instead of MFCCs, both for performing the DTW matching and as inputs to the CNN-DTW model. In Figure 2, the upper half shows how the supervisory signals are obtained using DTW, and the lower half shows how the CNN is trained. Equation (1) shows how keyword scores are computed, resulting in a vector $[c_1, \ldots, c_j, \ldots c_L]$ for each utterance $\mathcal{U}$.

$$c = \min_{i \in 1 \ldots N} \left[\min_{u_p \in \mathcal{U}} \text{DTW}\{k_i, u_p\} \right] \tag{1}$$

Here, k_i is the sequence of speech features for the i^{th} exemplar of keyword $\mathcal{K}$, u_p is a successive segment of utterance $\mathcal{U}$, and $\text{DTW}\{k_i, u_p\}$ is the DTW alignment cost between the speech features of exemplar k_i and the segment u_p. Each value c_j, which is between $[0, 2]$, is then mapped to $y_j \in [0, 1]$, with 1 indicating a perfect match and 0 indicating dissimilarity thus forming the target vector $\boldsymbol{y}$ for utterance $\mathcal{U}$. A CNN is then trained using a summed cross-entropy loss (which is why the scores are mapped to the interval $[0, 1]$) with utterance $\mathcal{U}$ as input and $\boldsymbol{y}$ as target. The CNN architecture is the same as used in [19]. Finally, the trained CNN is applied to unseen utterances.

[2] The CNN has 3 convolutional layers (filters with 64, 128, 256 units) with max pooling, followed by 3 dense layers (500, 100 and 300 neural units). We use a dropout of 0.5 for the first and last dense layer.

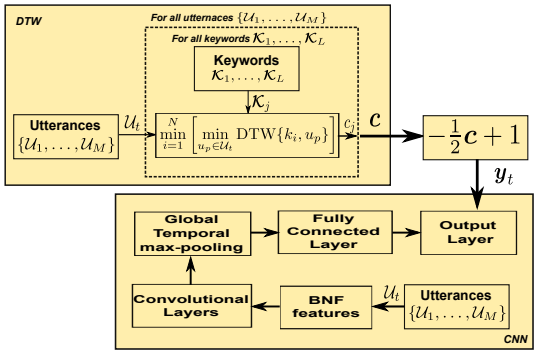


Figure 2: *The CNN-DTW keyword spotter using BNFs. The top half shows how the supervisory signal is obtained and the bottom half how this signal is used to train the CNN.*

5. Bottleneck and Autoencoder Features

Our previous work focused purely on using data from the low-resource target language. However, large annotated speech resources exist for several well-resourced languages. We investigate whether such resources can be used to improve the CNN-DTW system in the unseen low-resource language.

5.1. Bottleneck features

One way to re-use information extracted from other multilingual corpora is to use multilingual bottleneck features (BNFs), which have been shown to perform well in conventional ASR as well as intrinsic evaluations [20, 21, 27–30]. These features are typically obtained by training a deep neural network jointly on several languages for which labelled data is available. The bottom layers of the network are normally shared across all training languages. The network then splits into separate parts for each of the languages, or has a single shared output. The final output layer has phone labels or HMM states as targets. The final shared layer often has a lower dimensionality than the input layer, and is therefore referred to as a 'bottleneck'.[3] The intuition is that this layer should capture aspects that are common across all the languages. We use such features from a multilingual neural network in our CNN-DTW keyword spotting approach. The BNFs are trained on a set of well-resourced languages different from the target language.

Different neural architectures can be used to obtain BNFs following the above methodology. Here we use time-delay neural networks (TDNNs) [31]. We consider two models: a multilingual TDNN trained on only two languages, and a TDNN trained on ten diverse languages. Our aim is to investigate whether it is necessary to have a large set of diverse languages, or whether it is sufficient to simply obtain features from a supervised model trained on a smaller set of languages.

5.1.1. 2-language TDNN

An 11-layer 2-language TDNN was trained using 40-high resolution MFCC features as input on a combined set of Dutch and Frisian speech, as described in [32]. Speaker adaptation is used with lattice-free maximum mutual information training, based on the Kaldi Switchboard recipe [33]. Each layer uses ReLU activations with batch normalisation. By combining the the FAME [34] and CGN [35] corpora, the training set consists of a combined 887 hours of data in the two languages. 40-dimensional BNFs are extracted from the resulting model.

[3] Confusingly, the term 'bottleneck' is now also sometimes used even when the layer does not have a smaller dimensionality than the input.

5.1.2. 10-language TDNN

A 6-layer 10-language TDNN was trained on the GlobalPhone corpus, also using 40-high resolution MFCC features as input, as described in [21]. For speaker adaptation, a 100-dimensional i-vector was appended to the the MFCC input features. The TDNN was trained with a block-softmax, with the hidden layers shared across all languages and a separate output layer for each language. Each of the six hidden layers had 625 dimensions, and was followed by a 39-dimensional bottleneck layer with ReLU activations and batch normalisation. Training was accomplished using the Kaldi Babel receipe using 198 hours of data in 10 languages (Bulgarian, Czech, French, German, Korean, Polish, Portuguese, Russian, Thai, Vietnamese) from GlobalPhone.

5.2. Autoencoder features

BNFs are trained in a supervised fashion with acoustic feature presented at the input and phone targets at the outputs. A more general scenario, however, is one in which training data is unlabelled, and these targets are therefore not known. In this case, it may be possible to learn useful representations by using an unsupervised model. An autoencoder is a neural network trained to reconstruct its input. By presenting the same data at the input and the output of the network while constraining intermediate connections, the network is trained to find an internal representation that is useful for reconstruction. These internal representations can be useful as features [36–41]. Like BNFs, autoencoders can be trained on languages different from the target language (often resulting in more data to train on).

Here we use a stacked denoising autoencoder [42]. In this model, each layer is trained individually like an autoencoder with added noise to reconstruct the output of the previous layer. Once a layer has been trained, its weights are fixed and its outputs become the inputs to the next layer to be trained. After all the layers are pre-trained in this fashion, the layers are stacked and fine-tuned. We use mean squared error loss and Adam optimisation [43] throughout. We trained a 7-layer stacked denoising autoencoder on an untranscribed dataset consisting of 160 h of Acholi, 154 h of Luganda, 9.45 h of Lugbara, 7.82 h of Rutaroo and 18 h of Somali data. We used 39-dimensional MFCCs (13 cepstra with deltas and delta-deltas) as input and extracted features from the 39-dimensional fourth layer.

6. Experimental setup

The experimental setup is similar to that of [19]. We consider three baseline systems: two DTW systems and a conventional CNN classifier.

1. **DTW-QbyE**, where DTW is performed for each exemplar keyword on each utterance, and the resulting scores averaged (§4.1).
2. **DTW-KS**, where the minimum (best) score over all exemplars of a keyword type is used per utterance (§4.1).
3. **CNN**, an end-to-end CNN classifier trained only on the isolated words (§4.2).

Our proposed approach, CNN-DTW, is supervised by the DTW-KS system. Hyper-parameters for CNN-DTW were optimized using the target loss on the development set.[4] Hence, the SABN transcriptions are not used for training or validation. Performance is reported in terms of the area under the curve (AUC) of

[4] Final system: 10 convolutional layers (between 80 and 512 filters), two 3000-unit fully connected layers with a dropout of 0.5, and a learning rate changing linearly from 10^{-4} to 10^{-5} used with Adam optimisation.

Table 2: *Performance of the different features on the development set when used in a DTW-based keyword spotter.*

Model	*dev*	
	AUC	EER
MFCC	0.7556	0.3092
SAE	0.5247	0.4844
TDNN-BNF-2lang	0.7273	0.3356
TDNN-BNF-10lang	0.7725	0.2884
TDNN-BNF-10lang-SPN	**0.7781**	**0.2872**

the receiver operating characteristic (ROC) and equal error rate (EER). The ROC is obtained by varying the detection threshold and plotting the false positive rate against the true positive rate. AUC, therefore, indicates the performance of the model independent of a threshold, with higher AUC indicating a better model. EER is the point at which the false positive rate equals the false negative rate and hence lower EER indicates a better model.

7. Experimental results

We consider four feature extractors in our experiments:

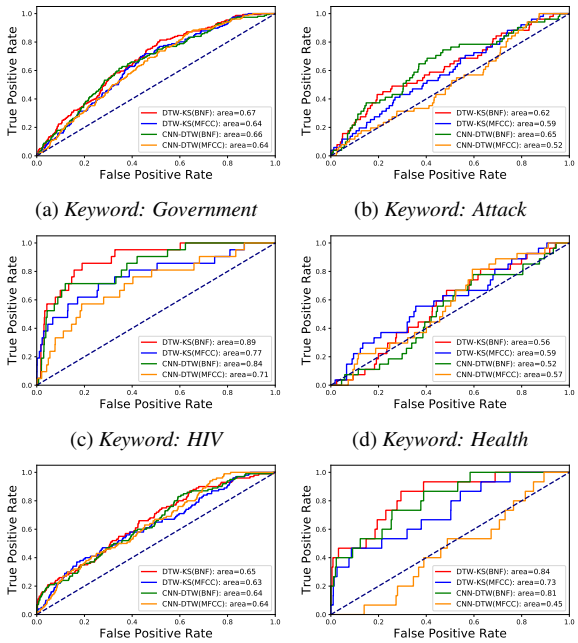
1. **SAE**, the stacked autoencoder (§5.2).
2. **TDNN-BNF-2lang**, the 2-language TDNN without speaker normalisation (§5.1.1).
3. **TDNN-BNF-10lang**, the 10-language TDNN without speaker normalisation (§5.1.2).
4. **TDNN-BNF-10lang-SPN**, the 10-language TDNN with speaker normalisation (§5.1.2).

In initial experiments, we first consider the performance of these features on development data. Specifically, we use the features as representations in the DTW-based keyword spotter (DTW-KS). Results are shown in Table 2. BNFs trained on 10 languages outperform all other approaches, with speaker normalisation giving a further slight improvement. Both the stacked autoencoder and the BNFs trained on two languages perform worse than the MFCC baseline. This seems to indicate that a larger number of diverse languages is beneficial for training BNFs, and that supervised models are superior to unsupervised models when applied to an unseen target language. However, further experiments are required to verify this definitively. Based on these development experiments, we compare MFCCs and TDNN-BNF-10lang-SPN features when used for keyword spotting on evaluation data.

Table 3 shows the performance of the three baseline systems and CNN-DTW when using MFCCs and BNFs. In all cases except the CNN classifier, BNFs lead to improvements over MFCCs. Furthermore, we see that, when using BNFs, the CNN-DTW system performs almost as well as its DTW-KS counterpart. The DTW-KS system provided the targets with which the CNN-DTW system was trained, and hence represents an upper bound on the performance we can expect from the CNN-DTW wordspotter. When using BNFs, we see that the difference between the DTW-KS and CNN-DTW approaches becomes smaller compared to the difference for MFCCs. This results in the CNN-DTW system using BNFs almost achieving the performance of the DTW-KS system. The former, however, is computationally much more efficient since alignment is not required. On a conventional desktop PC with a single NVIDIA GeForce GTX 1080 GPU, CNN-DTW takes approximately 5 minutes compared to DTW-KS which takes 900 minutes on a 20-core CPU machine. Table 3 shows that, in contrast to when MFCCs are used, a Gaussian noise layer (CNN-DTW-GNL) does not give further performance benefits for the BNF systems.

Table 3: *Performance of different keyword spotting systems using MFCCs and BNFs (TDNN-BNF-10lang-SPN).*

Model	AUC				EER			
	dev		*test*		*dev*		*test*	
	MFCC	BNF	MFCC	BNF	MFCC	BNF	MFCC	BNF
CNN	0.5698	0.5298	0.5448	0.5364	0.4435	0.4813	0.4771	0.4725
DTW-QbyE	0.6639	0.6899	0.6612	0.6873	0.3864	0.3556	0.3885	0.3661
DTW-KS	0.7556	0.7781	0.7515	0.7699	0.3092	0.2872	0.3162	0.3012
CNN-DTW	0.6360	0.7537	0.6285	0.7422	0.4073	0.3058	0.4161	0.3214
CNN-DTW-GNL	0.6443	0.7535	0.6357	0.7518	0.4036	0.3091	0.4092	0.3153



(a) *Keyword: Government* (b) *Keyword: Attack*

(c) *Keyword: HIV* (d) *Keyword: Health*

(e) *Keyword: War* (f) *Keyword: Wounded*

Figure 3: *Receiver operating characteristic curves for selected keywords for the DTW keyword spotter and the CNN-DTW system when using MFCCs and BNFs.*

Figures 3(a-f) show ROC plots for a selection of keywords which are representative of cases with both good and bad performance. AUC improves in all cases when switching from MFCCs to BNFs, except for *health*, where the difference is relative small (all scores are close to chance on this keyword). In some cases, e.g. for *wounded*, the benefits of switching to BNFs in CNN-DTW is substantial. Interestingly, for keywords such as *attack*, the CNN-DTW system using BNFs actually marginally outperforms the DTW-KS system which is used to supervise it.

8. Conclusion

We investigated the use of multilingual bottleneck (BNFs) and autoencoder features in a CNN-DTW keyword spotter. While autoencoder features and BNFs trained on two languages did not improve performance over MFCCs, BNFs trained on a corpus of 10 languages lead to substantial improvements. We conclude that our overall CNN-DTW based approach, which combines the low-resource advantages of DTW with the speed advantages of CNNs, further benefits by incorporating labelled data from well-resourced languages through the use of BNFs when these are obtained from several diverse language.

Acknowledgements: We thank the NVIDIA corporation for the donation of GPU equipment used for this research. We also gratefully acknowledge the support of Telkom South Africa.

9. References

[1] S. Vosoughi and D. Roy, "A human-machine collaborative system for identifying rumors on Twitter," in *Proc. ICDMW*, 2015.

[2] K. Wegrzyn-Wolska, L. Bougueroua, and G. Dziczkowski, "Social media analysis for e-health and medical purposes," in *Proc. CASoN*, 2011.

[3] P. Burnap, G. Colombo, and J. Scourfield, "Machine classification and analysis of suicide related communication on Twitter," in *Proc. ACM-HT*, 2015.

[4] R. Menon, A. Saeb, H. Cameron, W. Kibira, J. Quinn, and T. Niesler, "Radio-browsing for developmental monitoring in Uganda," in *Proc. ICASSP*, 2017.

[5] A. Saeb, R. Menon, H. Cameron, W. Kibira, J. Quinn, and T. Niesler, "Very low resource radio browsing for agile developmental and humanitarian monitoring," in *Proc. Interspeech*, 2017.

[6] M. Larson and G. J. F. Jones, "Spoken content retrieval: A survey of techniques and technologies," *Foundations and Trends in Information Retrieval*, vol. 5, no. 4-5, pp. 235–422, 2012.

[7] A. Mandal, K. R. P. Kumar, and P. Mitra, "Recent developments in spoken term detection: A survey," *International Jour. of Speech Technology*, vol. 17, no. 2, pp. 183–198, 2014.

[8] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks recognition," in *Proc. ICASSP*, 2015.

[9] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. ICASSP*, 2017.

[10] T. J. Hazen, W. Shen, and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *Proc. ASRU*, 2009.

[11] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams," in *Proc. ASRU*, 2009.

[12] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. ASRU*, 2013.

[13] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *Proc. ICASSP*, 2015.

[14] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," in *Proc. SLT*, 2016.

[15] Y. Chung, C. Wu, C. Shen, H. Lee, and L. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," in *Proc. Interspeech*, 2016.

[16] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Proc. Interspeech (accepted)*, 2018.

[17] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *Proc. ICASSP*, 2016.

[18] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end ASR-free keyword search from speech," in *Proc. ICASSP*, 2017.

[19] R. Menon, H. Kamper, J. Quinn, and T. Niesler, "Fast ASR-free and almost zero-resource keyword spotting using DTW and CNNs for humanitarian monitoring," in *Proc. Interspeech (accepted)*, 2018.

[20] K. Veselỳ, M. Karafiát, F. Grézl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Proc. SLT*, 2012.

[21] E. Hermann and S. Goldwater, "Multilingual bottleneck features for subword modeling in zero-resource languages," in *Proc. Interspeech (accepted)*, 2018.

[22] Y. Yuan, C.-C. Leung, L. Xie, H. Chen, B. Ma, and H. Li, "Pairwise learning using multi-lingual bottleneck features for low-resource query-by-example spoken term detection," in *Proc. ICASSP*, 2017.

[23] ——, "Learning acoustic word embeddings with temporal context for query-by-example speech search," in *Proc. Interspeech (accepted)*, 2018.

[24] H. Kamper, F. D. Wet, T. Hain, and T. Niesler, "Capitalising on North American speech resources for the development of a South African English large vocabulary speech recognition system," *Computer Speech and Language*, vol. 28, no. 6, pp. 1255–1268, 2014.

[25] D. Palaz, G. Synnaeve, and R. Collobert, "Jointly learning to locate and classify words using convolutional networks," in *Proc. Interspeech*, 2016.

[26] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech*, 2015.

[27] J. Cui *et al.*, "Multilingual representations for low resource speech recognition and keyword search," in *Proc. ASRU*, 2015.

[28] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in *Proc. ICASSP*, 2012.

[29] H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li, "Multilingual bottle-neck feature learning from untranscribed speech," in *Proc. ASRU*, 2017.

[30] Y. Yuan, C.-C. Leung, L. Xie, H. Chen, B. Ma, and H. Li, "Extracting bottleneck features and word-like pairs from untranscribed speech for feature representation," in *Proc. ASRU*, 2017.

[31] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015.

[32] E. Yılmaz, H. Van den Heuvel, and D. A. Van Leeuwen, "Acoustic and textual data augmentation for improved ASR of code-switching speech," in *Proc. Interspeech (accepted)*, 2018.

[33] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016, pp. 2751–2755.

[34] E. Yılmaz, M. Andringa, S. Kingma, F. Van der Kuip, H. Van de Velde, F. Kampstra, J. Algra, H. Van den Heuvel, and D. Van Leeuwen, "A longitudinal bilingual Frisian-Dutch radio broadcast database designed for code-switching research," in *Proc. LREC*, 2016, pp. 4666–4669.

[35] N. Oostdijk, "The spoken Dutch corpus. overview and first evaluation." in *LREC*, 2000.

[36] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints," in *Proc. ICASSP*, 2015.

[37] L. Badino, C. Canevari, L. Fadiga, and G. Metta, "An auto-encoder based approach to unsupervised learning of subword units," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014.

[38] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[39] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[40] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proc. ICASSP*, 2012.

[41] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Proc. ICASSP*, 2013.

[42] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.