Feature exploration for almost zero-resource ASR-free keyword spotting using a multilingual bottleneck extractor and correspondence autoencoders

Raghav Menon[1], *Herman Kamper*[1], *Ewald van der Westhuizen*[1], *John Quinn*[2,3,4], *Thomas Niesler*[1]

[1]Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa
[2]UN Global Pulse, Kampala, Uganda
[3]Department of Computer Science, Makerere University, Uganda
[4]School of Informatics, University of Edinburgh, UK

rmenon@sun.ac.za kamperh@sun.ac.za, ewaldvdw@sun.ac.za, trn@sun.ac.za

Abstract

We compare features for dynamic time warping (DTW) when used to bootstrap keyword spotting (KWS) in an almost zero-resource setting. Such quickly-deployable systems aim to support United Nations (UN) humanitarian relief efforts in parts of Africa with severely under-resourced languages. Our objective is to identify acoustic features that provide acceptable KWS performance in such environments. As supervised resource, we restrict ourselves to a small, easily acquired and independently compiled set of isolated keywords. For feature extraction, a multilingual bottleneck feature (BNF) extractor, trained on well-resourced out-of-domain languages, is integrated with a correspondence autoencoder (CAE) trained on extremely sparse in-domain data. On their own, BNFs and CAE features are shown to achieve a more than 2% absolute performance improvement over baseline MFCCs. However, by using BNFs as input to the CAE, even better performance is achieved, with a more than 11% absolute improvement in ROC AUC over MFCCs and more than twice as many top-10 retrievals for two evaluated languages, English and Luganda. We conclude that integrating BNFs with the CAE allows both large out-of-domain and sparse in-domain resources to be exploited for improved ASR-free keyword spotting.

Index Terms: Keyword spotting, low-resource speech processing, multilingual features, correspondence autoencoder, zero-resource speech technology

1. Introduction

In Uganda, internet infrastructure is often poorly developed, precluding the use of social media to gauge sentiment. Instead, community radio phone-in talk shows are used to voice views and concerns. In a project piloted by the United Nations (UN), radio browsing systems have been developed to monitor such radio shows [1, 2]. Currently, these systems are actively and successfully supporting relief and developmental programmes by the organisation. However, the deployed radio browsing systems use automatic speech recognition (ASR) and are therefore highly dependent on the availability of substantial transcribed speech corpora in the target language. This has proved to be a serious impediment when quick intervention is required, since the development of such a corpus is always time-consuming.

In a conventional keyword spotting system, where a speech database is searched for a set of keywords, ASR is used to generate lattices which are in turn searched for the presence or absence of keywords [3, 4]. In resource-constrained settings where ASR is not available and cannot be developed, ASR-free keyword spotting approaches become attractive, because these are developed without substantial labelled data [5–10]. One approach to ASR-free keyword spotting is to extend query-by-example search (QbE), where the search query is provided as audio rather than a written keyword. QbE can be performed by using dynamic time warping (DTW) to perform a direct match between a search query and utterances in the search collection [11–14]. This approach uses a number of labelled spoken keyword instances as templates. Each template is used as a query for the DTW-based QbE. Since the class of each template is known, the individual per-exemplar QbE results can be aggregated to determine whether a certain keyword occurs in a particular utterance. The advantage of this approach is that only a small set of labelled keywords is required and not a large transcribed corpus as used for ASR-based keyword spotting [6, 7].

Recent interest in zero-resource QbE has led researchers to consider the use of various features [15–21]. Among these, multilingual bottleneck feature (BNF) extractors, trained on well-resourced but out-of-domain languages, have been shown to improve on the performance of MFCCs [7, 22–30].

Our goal is to improve DTW-based keyword spotting by combining the advantages of using labelled resources from well-resourced languages for learning features, with the advantage of fine-tuning on extremely sparse labelled data in the low-resource target language. For fine-tuning on target data, we use the correspondence autoencoder (CAE), a model originally developed for the zero-resource setting where only unlabelled data is available [21, 31]. As target language data, we use a small number of labelled isolated keywords that can be easily and quickly gathered. These keyword instances do not form part of the radio talk show training and evaluation data and can thus be considered out-of-corpus augmentation data. By learning a mapping between all possible combinations of alternative utterances of the same keyword type, the CAE can learn to disregard aspects not common to the keywords, such as speaker, gender and channel, while capturing aspects that are, such as word identity. Our work builds on the ideas established in [22, 23], where a CAE trained on BNFs using a large set of in-corpus, ground truth word pairs outperformed other methods in intrinsic evaluations. This improvement, however, did not hold consistently when automatically discovered word segments were used, in which case the CAE training was completely unsupervised. In contrast, we show here that consistent improvements can be obtained by combining BNFs with a CAE when fine-tuning on a small number of out-of-corpus gathered keyword instances, i.e. lightly supervised.

We benchmark CAE features against MFCCs and BNFs and show that, when a CAE is trained on top of the BNFs, best keyword spotting results are achieved. This indicates that multilingual feature extraction and target language fine-tuning can

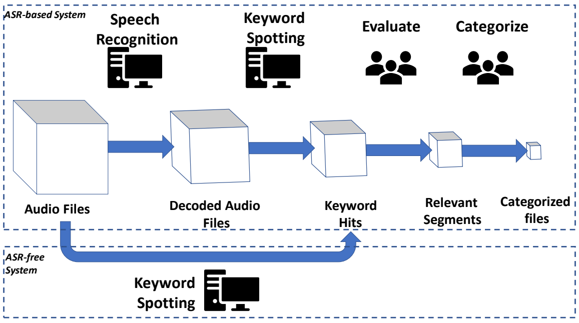

Figure 1: *The United Nations (UN) radio browsing system.*

be complementary. We evaluate our approach for two languages: English, which is a proxy language for experimentation; and Luganda, which is a low-resource language of current interest for humanitarian relief efforts.

2. Radio browsing system

The existing UN radio browsing system, shown in the top half of Figure 1, uses ASR to decode the audio and produces lattices that are searched for keywords. Human analysts filter the detected keywords and their metadata is compiled into a structured, categorised and searchable format. The ASR-free system (bottom half) bypasses the ASR and lattice search by detecting occurrences of the keywords directly in the incoming audio [6, 7]. High false positive keyword spotting rates can be accommodated due to the presence of the human analysts, and the output of the system as a whole has been in continuous successful operation for several months. A more detailed discussion on the role of human analysts and the detected topics of interest has been presented in [2].[1]

3. Data

We used a 23-hour English corpus of South African Broadcast News (SABN) [32] and a 9.6-hour corpus of Luganda phone-in talk radio speech as search data in two separate experiments. Since transcriptions are available for these data sets, it allows system performance to be experimentally evaluated. However, in all other respects we consider the data as untranscribed. English is used as a proxy on which we can perform extensive evaluation, while the implementation in Luganda is a practical application of the system in a truly low-resource language. Table 1 shows how the corpora have been split into training, development and test sets.

To train the English keyword spotter, we use a small independent corpus of 40 isolated keywords, each uttered at least once by 24 South African speakers (12 male, 12 female). The resulting set of 1160 isolated keyword utterances represents the only labelled in-domain data the English keyword spotter uses for training. There is no speaker overlap with the SABN dataset, which is treated exclusively as search data.

Table 1: *The South African English Broadcast News (SABN) and Luganda datasets. (#utts: Number of utterances; dur: Speech duration in hours; Dev: Development set.)*

Set	**English**		**Luganda**	
	#utts	**dur**	**#utts**	**dur**
Train	5 231	7.94	6 052	5.57
Dev	2 740	5.37	1 786	2.04
Test	5 005	10.33	1 420	1.99
Total	12 976	23.64	9 258	9.06

To train the Luganda keyword spotter, we use a small independent corpus of 18 isolated keywords uttered by various male and female speakers in varying recording conditions. Approximately 32 utterances per keyword type were retained after performing quality control on the recordings. The resulting set of 603 isolated keyword utterances represents the only labelled in-domain data our keyword spotter uses for training. There is no speaker overlap with the Luganda talk radio dataset, which is treated exclusively as search data. Seven keyword types which had frequencies higher than 10 in the corpus development were retained for evaluation against the development set. This was done to avoid errors in calculating the metrics caused by very low and zero frequency keywords. For the test set, the full set of keywords was used for evaluation.

The mismatch between the query and search datasets for both languages is intentional as it reflects the operational setting of the radio browsing systems.

4. Dynamic time warping-based keyword spotting

Dynamic time warping (DTW) is an appropriate approach to keyword detection when only a few isolated exemplars of keywords are available, because it requires as little as a single audio template. DTW aligns two time series, represented as feature vector sequences, by warping the relative time axes iteratively until an optimal match is found.

For DTW-based keyword spotting, features are extracted for both the keyword exemplar and the search utterance in which the keyword is to be detected. In our straightforward implementation, the keyword exemplar is slid progressively over the search utterance and at each step DTW computes the alignment cost between the keyword and the portion of the utterance under alignment. Using a step of 3 frames, the overall best alignment for each search utterance is determined and taken as a score indicating how likely it is that the search utterance contains the keyword. Since we have more than one exemplar of the same keyword type, the best score across all templates of the same keyword type is used. By applying an appropriate threshold to this score, a decision can be taken regarding the presence or absence of the keyword in each search utterance. More refined DTW-based search approaches have been proposed [11–14], mainly to improve efficiency, but here we restrict ourselves to this straightforward implementation. Future work will consider more advanced matching approaches.

5. Neural network feature extraction

We investigate different types of input features for our DTW-based keyword spotter. While transcribed in-domain data is difficult, time-consuming and expensive to compile, untranscribed in-domain speech audio data is much easier to obtain in substantial quantities. We investigate the use of autoencoders and correspondence autoencoders as a means of taking advantage of such untranscribed data. The latter requires a sparse set of labelled examples in the target language. In addition, although

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

large amounts of transcribed in-domain speech data may not be available, large annotated speech resources do exist for several well-resourced languages. These datasets can be used to train multilingual bottleneck feature extractors.

5.1. Autoencoder features

An autoencoder (AE) is a feedforward neural network trained to reconstruct its input at its output. A single-layer AE consists of an input layer, a hidden layer and an output layer. The AE takes input $\mathbf{x} \in \mathbb{R}^D$ and maps it to a hidden representation $\mathbf{h} = \sigma(\mathbf{W}^{(0)}\mathbf{x} + \mathbf{b}^{(0)})$, with σ denoting a non-linear activation (we use $\tanh$). The output of the AE is obtained by decoding the hidden representation: $\mathbf{y} = \sigma(\mathbf{W}^{(1)}\mathbf{h} + \mathbf{b}^{(1)})$. The network is trained to reconstruct the input using the loss $||\mathbf{x} - \mathbf{y}||^2$.

A stacked AE [33] is obtained by stacking several AEs, each AE-layer taking as input the encoding from the previous layer. The stacked network is trained one layer at a time, each layer minimizing the loss of its output with respect to its input. A number of studies have shown that hidden representations from an intermediate layer in such a stacked AE are useful as features in speech applications [31, 33–38].

We train an 8-layer stacked AE feature extractor on the training set shown in Table 1, disregarding the transcriptions. 39-dimensional MFCCs consisting of 13 cepstra, delta and delta-delta coefficients are used as input. All layers have 100 hidden units, apart from the last hidden layer, which has 39 units. This layer provides the features used in the AE_{MFCC} and AE_{BNF} experiments. This last hidden layer feeds into a linear output layer, producing the predicted MFCC vector.

5.2. Correspondence autoencoder features

While an AE is trained using the same speech frames as input and output, a correspondence autoencoder (CAE) uses frames from different instances of the same keyword type as input and output. Using the set of isolated keywords, we consider all possible pairs of words of the same type. For each pair, DTW is used to find the minimum-cost frame-level alignment between the two words, as illustrated in Figure 2. Individual aligned frame pairs are then used as input-output pairs to the CAE. The CAE is therefore trained on pairs of speech features $(\mathbf{x}^{(a)}, \mathbf{x}^{(b)})$, where $\mathbf{x}^{(a)}$ is a frame from one keyword, and $\mathbf{x}^{(b)}$ the corresponding aligned frame from another keyword of the same type. Given input $\mathbf{x}^{(a)}$, the output of the network $\mathbf{y}$ is then trained to minimise the the CAE loss $||\mathbf{y} - \mathbf{x}^{(b)}||^2$, as shown in Figure 2.

To obtain useful features, it is essential to pretrain the CAE as a conventional AE [31]. Our CAE has the same structure as the AE described in Section 5.1 and pretraining follows the same procedure described there. The pretrained network is then fine-tuned on the set of isolated keywords using the CAE loss described above. Hence, the CAE takes advantage of a large amount of untranscribed data for initialisation, and then combines this with a weak form of supervision on a small amount of labelled keyword data. Output features are extracted from the last 39-dimensional hidden layer.

The intention is to use the CAE to obtain features that are insensitive to factors not common to keyword pairs, such as speaker, gender and channel, while remaining dependent on factors that are, such as the word identity. Furthermore, the number of input-output pairs on which the CAE is fine-tuned is much larger than the total number of frames in the keyword segments themselves, because all pairwise combinations of different instances of a keyword type are considered. For example, for the SABN dataset, the keywords contain approximately 120k frames in total, while the pairwise combinations yield approximately two million unique aligned frame pairs. Furthermore, frame pairs are presented to the CAE in both input-output directions, thereby doubling the number of training instances to four million.

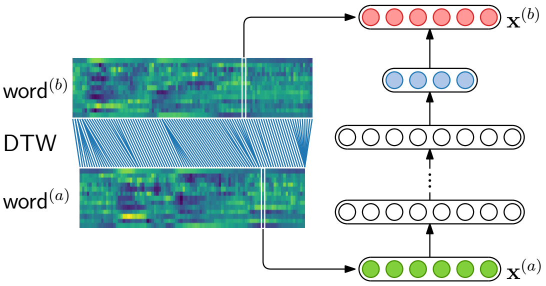

Figure 2: *The correspondence autoencoder (CAE) is trained to reconstruct a frame in one word from a frame in another.*

5.3. Bottleneck features

Multilingual bottleneck feature (BNF) extractors trained on a set of well-resourced languages have been shown to perform well in a number of studies [7, 22–30], and can be applied directly in an almost zero-resource setting. BNFs are obtained by training a deep neural network jointly on transcribed data from multiple languages. The lower layers of the network are shared among all languages. The output layer has phone or HMM state labels as targets and may either be shared by or be separate for each language. The layer directly preceding the output layer often has a lower dimensionality than the preceding layers, because it should capture aspects that are common to all the languages, hence, the term “bottleneck.”

Different neural network architectures can be used to obtain BNFs. We used the 6-layer time-delay neural networks (TDNN) trained on 10 languages from the GlobalPhone corpus described in [22]. The network uses ReLU activations and batch normalisation, with a 39-dimensional bottleneck layer. 40-dimensional high resolution MFCCs appended with 100-dimensional i-vectors for speaker adaptation are used as inputs to the network.

6. Experimental setup

In addition to MFCCs, we use each of the neural networks described above as feature extractors, using features from the intermediate/bottleneck layers of the CAE, AE and BNF as input to our DTW-based keyword spotter. All the neural networks take MFCCs as input. Each takes advantage of resources in a particular way: the AE is trained on untranscribed target language data; the CAE is initialised on untranscribed data and then fine-tuned on a small amount of labelled target language data; and the BNFs use larger amounts of labelled non-target language data. The complimentary effect of these approaches are also investigated by performing experiments in which the AE and CAE are trained with BNFs rather than MFCCs as input. Hyperparameters for the CAE were taken directly from [31], i.e., no further tuning was performed on the development set, hence, it can be considered a second test set.

Keyword spotting performance is assessed using a number of standard metrics. The receiver operating characteristic (ROC) is obtained by plotting the false positive rate against the true positive rate as the keyword detection threshold is varied. The

Table 2: *English and Luganda keyword spotting performance on development and test data using the different feature representations. Subscripts are in the column headings to distinguish whether MFCCs or BNFs were used as inputs to the AE and CAE.*

Metric	*Development* (%)						*Test* (%)			
	MFCC	**AE$_{\text{MFCC}}$**	**CAE$_{\text{MFCC}}$**	**BNF**	**AE$_{\text{BNF}}$**	**CAE$_{\text{BNF}}$**	**MFCC**	**CAE$_{\text{MFCC}}$**	**BNF**	**CAE$_{\text{BNF}}$**
					English					
AUC	73.32	73.01	77.14	77.81	78.38	**86.98**	74.10	76.86	76.99	**86.39**
EER	32.34	33.51	28.91	28.72	28.23	**19.24**	32.19	30.05	30.12	**20.12**
P@10	15.75	16.50	25.25	17.00	17.75	**42.25**	17.00	30.25	22.75	**45.75**
$P@N$	9.43	9.68	14.66	13.99	13.64	**30.88**	9.75	16.45	12.85	**29.99**
					Luganda					
AUC	66.51	67.52	69.62	71.24	72.73	**78.09**	69.57	69.74	73.33	**80.59**
EER	38.68	38.57	37.20	33.26	31.20	**29.33**	37.20	37.24	33.73	**29.00**
P@10	11.43	14.29	28.57	11.43	10.00	**45.71**	18.89	27.78	26.11	**41.67**
$P@N$	10.72	10.13	13.95	9.99	11.61	**26.11**	13.87	18.77	18.21	**28.80**

area under this curve (AUC) is used as a single metric across all operating points. The equal error rate (EER) is the point at which the false positive rate equals the false negative rate, i.e. a lower EER indicates better system performance. Precision at 10 (P@10) and precision at N ($P@N$) are the proportion of correct keyword detections among the top 10 and top N hits, respectively.

7. Results

The keyword spotting results for both languages are presented in Table 2. The column headings with 'MFCC' and 'BNF' are used to distinguish between networks trained using MFCCs and BNFs as input features. The results for MFCC, AE$_{\text{MFCC}}$ and CAE$_{\text{MFCC}}$ features show that the CAE consistently outperforms the MFCC baseline, while the AE does not provide any benefit in this case. The BNF and CAE$_{\text{MFCC}}$ results are comparable in the case of SABN English, while BNFs outperform CAE$_{\text{MFCC}}$ for Luganda. Using a small amount of labelled data in a target language can therefore be just as beneficial as using large amounts of labelled data from several non-target languages for feature learning. This may be important in situations where large out-of-domain datasets are not available.

Our best overall model on both the development and test data is the CAE$_{\text{BNF}}$. It achieves precision values of approximately 1.7 times better than the closest competitor, while the AUC and EER are approximately 7–9% and 4–10% better than standard BNFs, respectively. Compared to the baseline MFCCs, AUC and EER improve by 8–12% when using the CAE$_{\text{BNF}}$ features. The AE$_{\text{BNF}}$ can also achieve improvements over its MFCC counterpart, but not to the same degree as CAE$_{\text{BNF}}$. The CAE$_{\text{BNF}}$ shows the benefits of incorporating features learned from well-resourced non-target languages with fine-tuning on a small amount of labelled target language data after pretraining on untranscribed in-domain speech. We show this directly in an extrinsic keyword spotting task that uses features obtained from a lightly supervised neural network model. In contrast to the work of [22, 23], where discovered word pairs were used for unsupervised CAE training and the benefit of CAE training on top of BNFs were inconclusive, we obtain consistent improvements in our setting.

8. Conclusion

We investigated the use of different neural network features for improving ASR-free DTW-based keyword spotting in an almost zero-resource setting. The only labelled data used were a small number of isolated keyword utterances. Features were extracted using a multilingual bottleneck network (BNF), a stacked autoencoder (AE) and a correspondence autoencoder (CAE). We also considered combining these, feeding the AE and CAE with BNFs instead of MFCCs. The best performance was achieved with a CAE trained on BNFs. This model combines the benefit of labelled data in well-resourced out-of-domain languages with a technique that can be used on extremely sparse in-domain data. Another interesting finding is that, in the absence of multilingual resources to train a BNF extractor, features from a CAE trained on MFCCs can yield comparable performance. Future work includes integrating this model into our larger keyword spotting framework [6] and applying it to languages such as Somali, Rutooro and Lugbara, which are spoken in areas where the system will be deployed next.

9. Acknowledgements

We thank NVIDIA Corporation for donating GPU equipment used for this work, and acknowledge the support of Telkom South Africa. HK is supported by a Google Faculty Award. We would also like to thank Enno Hermann for assisting with bottleneck feature extraction.

10. References

[1] R. Menon *et al.*, "Radio-browsing for developmental monitoring in Uganda," in *Proc. ICASSP*, 2017.

[2] A. Saeb *et al.*, "Very low resource radio browsing for agile developmental and humanitarian monitoring," in *Proc. Interspeech*, 2017.

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

[4] A. Mandal, K. P. Kumar, and P. Mitra, "Recent developments in spoken term detection: a survey," *Int. J. of Speech Technol.*, vol. 17, no. 2, pp. 183–198, 2014.

[5] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and

B. Kingsbury, “End-to-end ASR-free keyword search from speech,” in *Proc. ICASSP*, 2017.

[6] 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*, 2018.

[7] ——, “ASR-free CNN-DTW keyword spotting using multilingual bottleneck features for almost zero-resource languages,” in *Proc. SLTU*, 2018.

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

[9] G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks.” in *Proc. ICASSP*, 2014.

[10] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” in *Proc. ICASSP*, 2018.

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

[12] Y. Zhang and J. R. Glass, “Unsupervised spoken keyword spotting via segmental DTW on gaussian posteriorgrams,” in *Proc. ASRU*, 2009.

[13] A. S. Park and J. R. Glass, “Unsupervised pattern discovery in speech,” *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 1, pp. 186–197, 2008.

[14] A. Jansen and B. Van Durme, “Indexing raw acoustic features for scalable zero resource search,” in *Proc. Interspeech*, 2012.

[15] J. Vavrek, M. Pleva, and J. Juhár, “Tuke mediaeval 2012: Spoken web search using DTW and unsupervised SVM,” in *MediaEval*, 2012.

[16] M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky, “Rapid evaluation of speech representations for spoken term discovery,” in *Proc. Interspeech*, 2011.

[17] A. Jansen, B. Van Durme, and P. Clark, “The JHU-HLTCOE spoken web search system for MediaEval 2012.” in *MediaEval*, 2012.

[18] P. Lopez-Otero, L. Docio-Fernandez, and C. Garcia-Mateo, “Finding relevant features for zero-resource query-by-example search on speech,” *Speech Commun.*, vol. 84, pp. 24–35, 2016.

[19] E. Dunbar *et al.*, “The Zero Resource Speech Challenge 2017,” in *Proc. ASRU*, 2017.

[20] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, “The Zero Resource Speech Challenge 2015: Proposed approaches and results,” in *Proc. SLTU*, 2016.

[21] D. Renshaw, H. Kamper, A. Jansen, and S. J. Goldwater, “A comparison of neural network methods for unsupervised representation learning on the Zero Resource Speech Challenge,” in *Proc. Interspeech*, 2015.

[22] E. Hermann and S. J. Goldwater, “Multilingual bottleneck features for subword modeling in zero-resource languages,” in *Proc. Interspeech*, 2018.

[23] E. Hermann, H. Kamper, and S. J. Goldwater, “Multilingual and unsupervised subword modeling for zero resource languages,” *In Submission*, 2018.

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

[25] N. T. Vu, W. Breiter, F. Metze, and T. Schultz, “An investigation on initialization schemes for multilayer perceptron training using multilingual data and their effect on asr performance,” in *Proc. Interspeech*, 2012.

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

[27] T. Alumäe, S. Tsakalidis, and R. M. Schwartz, “Improved multilingual training of stacked neural network acoustic models for low resource languages,” in *Proc. Interspeech*, 2016.

[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 *et al.*, “Extracting bottleneck features and word-like pairs from untranscribed speech for feature representation,” in *Proc. ASRU*, 2017.

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

[32] H. Kamper, F. De 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,” *Comput. Speech Language*, vol. 28, no. 6, pp. 1255–1268, 2014.

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

[34] M. D. Zeiler *et al.*, “On rectified linear units for speech processing,” in *Proc. ICASSP*, 2013.

[35] L. Badino, C. Canevari, L. Fadiga, and G. Metta, “An auto-encoder based approach to unsupervised learning of subword units,” in *Proc. ICASSP*, 2014.

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

[37] 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 *Proc. Interspeech*, 2010.

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