Multilingual Query-by-Example KWS for Indian Languages using Transliteration

Kirandevraj R[1]*, Vinod K Kurmi*[2]*, Vinay Namboodiri*[3]*, CV Jawahar*[1]

[1]IIIT Hyderabad, India
[2]IISER Bhopal, India
[3]University of Bath, UK

kirandevraj.r@research.iiit.ac.in, vinodkk@iiserb.ac.in, vpn22@bath.ac.uk, jawahar@iiit.ac.in

Abstract

Query-by-Example Keyword Spotting (QbE KWS) detects query audio within target audio. A common approach for multilingual QbE KWS uses phoneme posteriors as representations, with a shared phoneme dictionary across languages. We propose a novel method that replaces phoneme-based representations with transliteration, unifying transcripts from multiple Indian languages into the Devanagari script, a text script used for Hindi and Marathi. We train a Multilingual ASR model to predict transliterated Devanagari text from audio across 10 Indian languages. The character logits from this ASR serve as both query and target audio features. Using the Kathbath dataset for training and the IndicSUPERB QbE evaluation set, our approach achieves significant improvements. The average MTWV increased from 0.015 (IndicSUPERB) to 0.504, and performance rose from 0.387 to 0.504, surpassing the best-performing Marathi ASR baseline. This demonstrates the effectiveness of transliteration for multilingual KWS.

Index Terms: keyword spotting, multilingual, transliteration, automatic speech recognition

1. Introduction

In real-world applications such as voice assistants, call center analytics, and accessibility tools, detecting specific audio keywords is crucial for providing meaningful and efficient services. With the growing volume of audio data being generated and stored, efficiently retrieving relevant audio documents has become critical. A common approach involves transcribing audio into text and performing text-based searches. However, searching in the embedding space offers a practical alternative for languages where Automatic Speech Recognition (ASR) systems lack robustness or sufficient training resources. QbE KWS addresses this challenge by determining whether a query phrase exists in target audio by comparing their respective embeddings [1].

The alignment between query and target embeddings is typically achieved using either neural networks or a dynamic time warping (DTW) algorithm applied to their similarity matrix [2]. We adopt a DTW-based alignment approach due to its simplicity and effectivenes in this work. Existing multilingual QbE KWS systems often rely on bottleneck features and phoneme-based representations but these systems lose higher-level linguistic and contextual details, introduce cross-language ambiguities, and require extensive phoneme dictionaries, making them resource-intensive and hard to scale [3, 4, 5]. To address these challenges, we propose a transliteration-based approach that converts text from multiple Indic languages—Tamil, Malayalam, Kannada, Telugu, Odia, Gujarati, Punjabi, and Bengali—into the Devanagari script, a common text representation used for Hindi and Marathi. Transliteration preserves linguistic information, reduces ambiguities, and eliminates the need for extensive phoneme resources, enabling scalable and effective multilingual KWS.

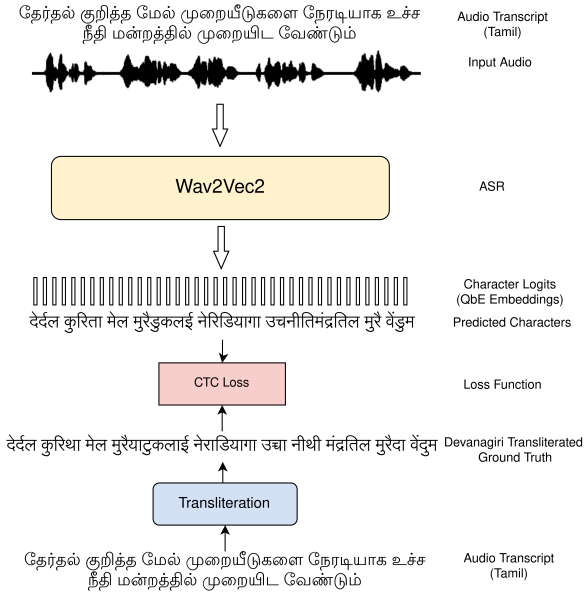

Figure 1: *Multilingual Transliteration ASR for QbE KWS*

We build the transliteration model using the Aksharantar dataset [6], adopting a methodology similar to IndicXlit [6]. The character logits produced by a Wav2Vec2 ASR model trained on transliterated text serve as embeddings for the queries and target audio. These embeddings provide a robust, language-independent representation, eliminating the need for word-level alignments. We compute a cosine similarity matrix between their embeddings, followed by applying the DTW algorithm to determine the presence of a query in target audio.

Our main contribution is the development of a transliteration-based multilingual QbE KWS system for regional languages. By converting text from multiple Indic languages into a unified script, we overcome the limitations of phoneme-based systems and address the scalability challenges inherent in multilingual keyword spotting. Using the Kathbath dataset for training and evaluating the IndicSUPERB QbE evaluation set, we demonstrate significant improvements. This work underscores the potential of transliteration as a scalable and effective solution for multilingual QbE KWS across diverse regional languages.

2. Related Work

QbE KWS has seen significant advancements with the emergence of deep learning. Early approaches relied on dynamic time warping and phonetic posteriorgrams for similarity comparison [7], but modern methods leverage neural embeddings and attention mechanisms to enhance performance [3]. CNN-based spoken term detection has also been explored for QbE KWS, demonstrating improved robustness [2]. Other approaches use text-based queries to search within audio embeddings [8, 9, 10, 11], while ASR posterior probabilities are utilized to predict fixed command sets [12].

Recent advancements in QbE KWS include multi-head attention with SoftTriple loss for improved feature extraction [13], spectral-temporal graph attentive pooling for speaker-invariant embeddings [14], and the MLP-based QbyE-MLPMixer for efficient open-vocabulary detection with reduced complexity [15].

Recent advancements in multilingual KWS focus on scalability, accuracy, and adaptability. Few-shot learning enables effective KWS with minimal data [16], while metric learning with phoneme-to-embedding mapping improves generalization across languages [4]. Transliteration has been used to convert Tamil keywords into English to leverage pretraining in low-resource KWS [17], and to map English keywords to Chinese phonemes for KWS without accented speech data [18]. In the Indic context, IndicSUPERB provides a comprehensive multilingual benchmark for speech processing evaluation [19].

3. Proposed Multilingual KWS Approach

3.1. Text Transliteration

We developed monolingual transliteration models using the Aksharantar dataset [6]. Parallel transliteration pairs were created for eight non-Devanagari script languages—Bengali, Gujarati, Kannada, Malayalam, Punjabi, Tamil, Telugu, and Odia—mapping their scripts to Devanagari. The original text was included in Hindi and Marathi, which natively use Devanagiri. Few examples of transliteration shown in Figure 2.

Language	Source	Transliteration
Bengali	দিল (dil)	दिल (dil)
Gujarati	જમણ (jaman)	जमान (jaman)
Kannada	ಅಂಚೊ (anchu)	अंचू (anchu)
Malayalam	ആകർഷ (aakarsha)	आकर्षा (aakarsha)
Odia	ନହର (nahara)	नहर (nahar)
Punjabi	ਖੜ੍ਹਕ (khadak)	खडक (khadak)
Tamil	ஆதவ் (aadhav)	आठव (aathav)
Telugu	అకాల (akala)	अकला (akala)

Figure 2: *Devanagiri Transliteration Samples with approximate english pronunciation in paranthesis*

The vocabulary from these languages is converted into Devanagiri script using the trained transliteration model for eight languages. This formed the ground truth for building a multilingual ASR dataset tailored to our keyword spotting task combined with existing Hindi and Marathi text.

3.2. Transliteration ASR

We build a multilingual transliteration-based ASR model for QbE KWS. We propose using the character logits from the ASR model as representations for both the input query and target audio. The ASR system is designed to predict transcripts in Devanagari, regardless of the input language. This is achieved by training the model with audio from multiple languages paired with their Devanagari transliterations obtained from Sec. 3.1.

The model outputs logits corresponding to 62 unique tokens, representing the Devanagari character set. These include vowels, consonants, diacritics, and special symbols such as word-boundary markers. We apply a log-softmax function to the model outputs, converting them into probabilities for each token. These probabilities are then used as embeddings for query and target audio in the keyword spotting task.

The ASR model is trained using the Connectionist Temporal Classification (CTC) loss function, which minimizes the difference between the predicted token sequences and the transliterated ground truth text in Devanagari script. This loss function ensures that the predicted sequences align effectively with the transliterated transcripts, enabling accurate token-level embeddings for QbE KWS. These steps are illustrated in Figure 1. The CTC loss function can be mathematically expressed as:

$$\mathcal{L}_{CTC} = -\log p(l|x) \tag{1}$$

where the probability of a label sequence l given an input sequence x is defined as:

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} \mathrm{ASR}_t^K(\mathbf{x})\left[\pi_t\right] \tag{2}$$

where $\mathcal{B}$ removes blanks and repeated symbols, $\mathcal{B}^{-1}$ is its inverse image, $\mathcal{T}$ is the length of the label sequence l, and $\mathrm{ASR}_t^K(\mathbf{x})\left[\pi_t\right]$ is unit j of the model output after the top softmax layer at time t, interpreted as the probability of observing label j at time t. The model outputs one frame of 62-dimensional logits every 20 ms at the final layer. The ASR predicts the transliteration of input audio from eight non-Devanagari script languages into the Devanagari character vocabulary, along with direct transcription for two native Devanagari script languages.

3.3. Query-by-Example KWS using DTW

The trained transliteration-based Wav2Vec2 ASR model generates softmax character logits as embeddings for both the query and audio document, which are compared using cosine similarity to construct a similarity matrix between their frames.

The Dynamic Time Warping (DTW) alignment [20, 21] is applied to locate the query within the audio. The DTW algorithm computes the optimal alignment between the query and document embeddings, providing a matching score that reflects the similarity between the two sequences. A slope-constrained DTW algorithm is employed to ensure a valid alignment, where the warping path is normalized by its partial path length at each step. This normalization helps manage variable query lengths and ensures robust alignment.

The warping path is flexible, allowing it to start and end at any point within the audio document, making it suitable for detecting queries regardless of their position. The DTW scores are normalized to have zero mean and unit variance for each query to account for variability across different queries and ensure consistency in score interpretation. This normalization step enhances comparability and helps reduce bias across queries of varying difficulty.

Dataset	Entries	bn	gu	hi	kn	ml	mr	or	pa	ta	te
Kathbath Dataset [19]	Vocabulary Size	61,061	88,677	53,621	159,957	157,962	141,376	30,959	49,037	191,494	136,189
	Clean Audio Hours	115.8	129.3	150.2	165.8	147.3	185.2	111.6	136.9	185.1	154.9
IndicSUPERB QbE Eval Set [19]	Utterances	839	1,046	1,049	1,046	1,049	1,031	942	732	1,047	1,048
	Queries	50	50	50	50	50	50	50	50	50	50
Aksharantar Devanagiri Pairs [6]	Training pairs	144,076	174,081	-	145,700	87,581	-	48,531	122,850	66,443	132,682
	Validation pairs	77	108	-	54	52	-	12	122	37	27

Table 1: *Datasets used for training transliteration models (Aksharantar), KWS models (Kathbath) and evaluation set (IndicSUPERB)*

4. Experimentation

4.1. Dataset

Aksharantar Dataset The Aksharantar dataset [6] is a transliteration resource for English-to-Indic and Indic-to-English text. The dataset does not provide direct transliteration pairs from non-Devanagari languages to Devanagari. Instead, these pairs are created by identifying common words between non-Devanagari to English and English to Devanagari datasets, enabling the mapping of non-Devanagari scripts to Devanagari. The number of pairs generated for each language is shown in the table. This dataset is used to train transliteration models to convert non-Devanagari scripts into Devanagari, standardizing text representation for multilingual ASR and QbE KWS.

Kathbath Dataset The Kathbath [19] training dataset consists of 1,506 hours of clean audio across 10 Indic languages, with a combined vocabulary size of 1.4M words. Details of audio duration and vocabulary size for each language are provided in Table 1. Audio from the train split is used to train our Multilingual KWS system, with the corresponding transcripts transliterated into Devanagari for training.

IndicSUPERB QbE Eval A separate QbE evaluation dataset, derived from the Kathbath dataset, consists of approximately 50 queries and 1,000 utterances per language Table 1. Each language features 20 speakers, with 10 providing the queries and 10 contributing to the target utterances. This dataset is used to evaluate the performance of our system.

4.2. Training

4.2.1. Transliteration Model

To enable text conversion from non-Devanagari scripts into a unified Devanagari script, we trained 8 monolingual transliteration models using the Fairseq framework. Each model was designed to transliterate between a specific non-Devanagari language—Bengali, Gujarati, Kannada, Malayalam, Punjabi, Tamil, Telugu, and Urdu—and Devanagari script.

The models were trained using parallel transliteration pairs prepared from the Aksharantar dataset. A transformer architecture with 6 encoder and decoder layers, multi-head attention, and GELU activation was employed. The training configuration included a batch size of 1024, a learning rate of 0.001 with an inverse square root scheduler, a 4000-step warmup phase, and a dropout rate of 0.5. The models were fine-tuned over 51 epochs to ensure robust performance. This approach closely followed the training methodology of IndicXlit [6], but with specialized monolingual models tailored for transliteration between each non-Devanagari language and Devanagari script.

4.2.2. Transliteration ASR-KWS Training

We use the Fairseq toolkit to train our transliteration ASR model, adopting the Wav2Vec2 large architecture for our experiments. The model comprises a feature extractor with 6 convolutional layers, a transformer encoder with 24 layers, and a projection layer for predicting character labels from the input audio. The Wav2Vec2-CTC large model implemented in fairseq comprises approximately 315 million parameters.

The Wav2Vec2 model is initialized with pre-trained weights from IndicWav2Vec [22], which was trained on 17,000 hours of audio data spanning 40 Indian languages. Our ASR model is trained to directly predict the transliterated text for the input audio. The transliterated text generated by the trained transliteration model is used as ground truth to compute the CTC loss.

Training is conducted with a maximum token size of 1.28M per batch using the Adam optimizer. The feature extractor is frozen for the first 10,000 updates to stabilize training. A masking probability of 0.5 and a layer drop probability of 0.1 are applied during training, while input features remain unnormalized to maintain compatibility with the pretraining setup. The model is trained for 10 epochs, and we report results from the best-performing model evaluated on the development set.

We used an NVIDIA RTX 4090 GPU for training, which took 24 hours to complete 10 epochs. The WER stopped decreasing after 10 epochs. The code is available here[1].

4.3. Baseline

IndicSUPERB We evaluate our system against the IndicSUPERB baseline, which uses a Wav2Vec2-based model pretrained [22] on 17,000 hours of audio data from 40 Indian languages. This model is initialized with a publicly available pretrained Wav2Vec2 model and fine-tuned using the training split of the Kathbath dataset. For the QbE KWS task, results are extracted from the best-performing layer of the baseline model and evaluated on the IndicSUPERB QbE evaluation set. These results are directly taken from [19].

Monolingual Wav2Vec2 ASR Logits The training approach involved is similar to IndicWav2Vec [22]. We also train monolingual ASR models for each of the languages using the Kathbath dataset. Pretrained weights from IndicWav2Vec are used to initialize the models, and each dataset is fine-tuned with the CTC loss specific to the respective language. The models are trained to predict the character dictionary of their respective languages, with training parameters kept consistent with those used for the transliteration ASR. These monolingual ASR models predict character labels tailored to their respective languages. We extract the character logits from each monolingual model as representations for both the query and the target audio, similar to how embeddings are generated by our transliteration ASR. The performance of these monolingual models serves as the second baseline for comparison in our evaluation. We used the character dictionaries for all languages provided from the IndicWav2Vec [22].

A comparison of the performance of these monolingual models for the multilingual QbE KWS task helped in selecting the Devanagari script as the target for transliteration.

[1] https://github.com/Kirandevraj/TranslitASR-KWS

Model Full Name	Bengali	Gujarati	Hindi	Kannada	Malayalam	Marathi	Punjabi	Tamil	Telugu	Odia	Average
IndicSUPERB [19]	0.026	0.003	0.004	0.023	0.017	0.046	0.008	0.021	0.012	0.007	0.017
Kannada ASR-KWS [22]	0.234	0.344	0.421	0.570	0.234	0.333	0.285	0.369	0.218	0.259	0.326
Punjabi ASR-KWS [22]	0.289	0.382	0.522	0.571	0.215	0.365	0.420	0.415	0.261	0.259	0.370
Marathi ASR-KWS [22]	0.282	0.392	0.537	0.598	0.253	0.445	0.372	0.439	0.273	0.274	0.387
Tamil ASR-KWS [22]	0.198	0.331	0.425	0.586	0.150	0.314	0.283	0.539	0.205	0.225	0.326
Gujarati ASR-KWS [22]	0.191	0.399	0.447	0.487	0.172	0.256	0.267	0.250	0.171	0.219	0.286
Odia ASR-KWS [22]	0.254	0.359	0.449	0.520	0.204	0.342	0.252	0.292	0.177	0.288	0.314
Hindi ASR-KWS [22]	0.175	0.295	0.442	0.517	0.084	0.351	0.295	0.331	0.188	0.217	0.290
Bengali ASR-KWS [22]	0.212	0.247	0.340	0.453	0.113	0.289	0.181	0.228	0.146	0.168	0.238
Telugu ASR-KWS [22]	0.199	0.315	0.352	0.464	0.212	0.218	0.231	0.333	0.260	0.257	0.284
Malayalam ASR-KWS [22]	0.196	0.359	0.345	0.594	0.234	0.313	0.276	0.431	0.251	0.264	0.326
Phoneme ASR-KWS	0.061	0.223	0.245	0.176	0.060	0.076	0.104	0.126	0.074	0.034	0.118
Devanagiri Translit ASR-KWS (hi-pairs)	0.399	**0.555**	0.649	0.624	0.299	0.469	**0.575**	**0.559**	**0.331**	0.484	0.495
Devanagiri Translit ASR-KWS (mr-pairs)	**0.407**	0.531	**0.664**	**0.670**	**0.343**	**0.512**	0.547	0.534	0.321	**0.512**	**0.504**

Table 2: *MTWV values of IndicSUPERB, Monolingual trained ASR, Phoneme ASR, Devanagiri Translit ASR for QbE KWS task evaluated on IndicSUPERB QbE KWS eval dataset.*

Multilingual Phoneme ASR The third baseline is a multilingual phoneme-based ASR model, which predicts phoneme sequences instead of characters. This system utilizes the eSpeak [23] module to convert textual transcriptions from all languages into their corresponding phoneme representations. These phoneme sequences are then used as training targets for the multilingual Phoneme ASR model. All the baseline models use Wav2Vec2 large architecture.

4.4. Metrics

We use the Maximum Term-Weighted Value (MTWV) as the primary metric to evaluate the performance of our QbE KWS system. MTWV balances the trade-off between missed detections and false alarms, with the cost of a false alarm set to 1 and the cost of a missed detection set to 100. The first baseline is also evaluated using this metric ensuring consistency in comparison. The DTW score derived from the similarity matrix evaluates the alignment between these representations. By varying the threshold on the DTW score, the MTWV score is computed. The MTWV is defined as:

$$\text{MTWV} = 1 - \min_{\theta} \left[P_{\text{miss}}(\theta) + \beta \cdot P_{\text{fa}}(\theta) \right] \tag{3}$$

where $P_{\text{miss}}(\theta)$ is the probability of missed detections at threshold θ, $P_{\text{fa}}(\theta)$ is the probability of false alarms at threshold θ, and β is a parameter balancing the cost of misses and false alarms.

4.5. Results

Table 2 summarizes the results for the baseline models (IndicSUPERB, monolingual ASR, and Phoneme ASR) and the transliteration-based ASR models on the QbE KWS task across ten Indic languages: Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Punjabi, Tamil, Telugu, and Odia. We applied Voice Activity Detection [24] to the eval set before running inference on these models as well as baseline models, as we observed that this led to improved performance.

Baseline Performance It can be observed that the performance of various baseline models ranges from 0.017 for IndicSUPERB to 0.387 for the Marathi Monolingual ASR. Unexpected trends in cross-lingual evaluation, where models perform better on a different language than their training language, can arise from linguistic similarity across languages and variability in evaluation conditions such as speaker characteristics, audio quality, and keyword distribution.

Transliteration ASR Models Transliteration-based ASR models significantly outperform all baseline models. The Transliteration ASR model (using Marathi pairs for Devanagari transliteration) achieves the highest average score of 0.504, delivering strong performance across all languages. The Transliteration ASR model (using Hindi pairs for Devanagari transliteration) achieves a comparable score of 0.495. The closer linguistic and phonetic alignment of Marathi and Hindi with other Indic languages likely contributes to its superior performance. The results demonstrate that transliteration-based approaches significantly enhance multilingual QbE KWS performance for diverse regional languages. Among these, the Transliteration ASR model (using Marathi pairs) showcases robust scalability and cross-lingual capabilities.

The KWS performance improved after unfreezing the model's backbone (after epoch 2) and training it end-to-end. However, when training exceeded 10 epochs, we observed a decrease in WER alongside a decline in KWS performance. This suggests that early stopping is essential to ensure the character logits retain maximum information for optimal KWS performance.

Text Transliteration The character error rates (CER) for the transliteration models range from 16.94% (Odia to Hindi) to 35.34% (Tamil to Hindi). The weighted average CER across all language pairs is 21.07%, calculated over 753 valid entries from the transliteration models.

Ablation We conducted an experiment in which all text was transliterated into English using IndicXlit [6] transliteration models. We then trained and evaluated the English Transliteration ASR-KWS QbE performance. This setup resulted in an average MTWV value of 0.243, highlighting the importance of selecting the appropriate target transliteration language.

We also conducted an experiment using the Wav2Vec2 base (small) model with 95M parameters to train the Devanagari Transliteration ASR. The pretrained weights from the IndicWav2Vec base [22] were used. We observed that the average MTWV value was 0.272, indicating that the model size significantly influences overall performance.

5. Conclusion

We proposed a transliteration-based approach for multilingual QbE KWS, demonstrating its scalability and effectiveness in diverse regional languages. Our method eliminates reliance on phoneme-based systems and achieves significant improvements, attaining an MTWV score of 0.504 by leveraging information from multiple languages. Future work could extend this framework to languages of diverse origins and further refine transliteration models to enhance performance.

6. Acknowledgement

This work was partly supported by Meity, Govt. of India.

7. References

[1] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *ArXiv*, vol. abs/1611.04496, 2016.

[2] D. Ram, L. Miculicich, and H. Bourlard, "Cnn based query by example spoken term detection," in *Interspeech*, 2018.

[3] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 532–538, 2019.

[4] P. Reuter, C. Rollwage, and B. T. Meyer, "Multilingual query-by-example keyword spotting with metric learning and phoneme-to-embedding mapping," *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[5] D. Ram, L. Miculicich, and H. Bourlard, "Multilingual bottleneck features for query by example spoken term detection," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 621–628.

[6] Y. Madhani, S. Parthan, P. A. Bedekar, R. Khapra, V. Seshadri, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Aksharantar: Towards building open transliteration tools for the next billion users," *ArXiv*, vol. abs/2205.03018, 2022.

[7] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5236–5240, 2015.

[8] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning audio-text agreement for open-vocabulary keyword spotting," in *Interspeech*, 2022.

[9] A. Navon, A. Shamsian, N. Glazer, G. Hetz, and J. Keshet, "Open-vocabulary keyword-spotting with adaptive instance normalization," *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11 656–11 660, 2023.

[10] Y.-H. Lee and N. Cho, "Phonmatchnet: Phoneme-guided zero-shot keyword spotting for user-defined keywords," *ArXiv*, vol. abs/2308.16511, 2023.

[11] K. Nishu, M. Cho, P. Dixon, and D. Naik, "Flexible keyword spotting based on homogeneous audio-text embedding," *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5050–5054, 2023.

[12] D. Seo, H.-S. Oh, and Y. Jung, "Wav2kws: Transfer learning from speech representations for keyword spotting," *IEEE Access*, vol. 9, pp. 80 682–80 691, 2021.

[13] J. Huang, W. Gharbieh, H. S. Shim, and E. Kim, "Query-by-example keyword spotting system using multi-head attention and soft-triple loss," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6858–6862, 2021.

[14] Z. Wang, S. Kong, L. Wan, B. Zhang, Y. Huang, M. Jin, M. Sun, X. Lei, and Z. Yang, "Query-by-example keyword spotting using spectral-temporal graph attentive pooling and multi-task learning," *ArXiv*, vol. abs/2409.00099, 2024.

[15] J. Huang, W. Gharbieh, Q. Wan, H. S. Shim, and C. Lee, "Qbye-mlpmixer: Query-by-example open-vocabulary keyword spotting using mlpmixer," *ArXiv*, vol. abs/2206.13231, 2022.

[16] M. Mazumder, C. R. Banbury, J. Meyer, P. Warden, and V. J. Reddi, "Few-shot keyword spotting in any language," in *Interspeech*, 2021.

[17] K. R., V. Kurmi, V. Namboodiri, and C. V. Jawahar, "Generalized keyword spotting using asr embeddings," in *Interspeech*, 2022.

[18] S. Zhang, Z. Shuang, and Y. Qin, "Automatic pronunciation transliteration for chinese-english mixed language keyword spotting," *2010 20th International Conference on Pattern Recognition*, pp. 1610–1613, 2010.

[19] T. Javed, K. Bhogale, A. Raman, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Indicsuperb: A speech processing universal performance benchmark for indian languages," *ArXiv*, vol. abs/2208.11761, 2022.

[20] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial intelligence in medicine*, vol. 45 1, pp. 11–34, 2009.

[21] T. Giorgino, "Computing and visualizing dynamic time warping alignments in r: The dtw package," *Journal of Statistical Software*, vol. 31, pp. 1–24, 2009.

[22] T. Javed, S. Doddapaneni, A. Raman, K. S. Bhogale, G. Ramesh, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Towards building asr systems for the next billion users," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[23] eSpeak NG contributors, "eSpeak NG: Speech Synthesizer," https://github.com/espeak-ng/espeak-ng/tree/master, 2025.

[24] M. Wise, "py-webrtcvad," https://github.com/wiseman/py-webrtcvad, 2017.