End-to-End Open Vocabulary Keyword Search With Multilingual Neural Representations

Bolaji Yusuf, *Graduate Student Member, IEEE*, Jan Černocký, *Senior Member, IEEE*, and Murat Saraçlar, *Member, IEEE*

***Abstract*—Conventional keyword search systems operate on automatic speech recognition (ASR) outputs, which causes them to have a complex indexing and search pipeline. This has led to interest in ASR-free approaches to simplify the search procedure. We recently proposed a neural ASR-free keyword search model which achieves competitive performance while maintaining an efficient and simplified pipeline, where queries and documents are encoded with a pair of recurrent neural network encoders and the encodings are combined with a dot-product. In this article, we extend this work with multilingual pretraining and detailed analysis of the model. Our experiments show that the proposed multilingual training significantly improves the model performance and that despite not matching a strong ASR-based conventional keyword search system for short queries and queries comprising in-vocabulary words, the proposed model outperforms the ASR-based system for long queries and queries that do not appear in the training data.**

***Index Terms*—Keyword search, spoken term detection, end-to-end keyword search, asr-free keyword search, keyword spotting.**

I. Introduction

KEYWORD search (KWS) is one of the technologies that arose out of the need to efficiently index and search the ever-growing catalog of spoken content online. Known alternatively as spoken term detection (STD), it entails locating short query phrases within large speech archives. Given a written query, which may or may not have been encountered at training time, a KWS system is expected to return which utterances in the archive, if any, contain the query, the time stamps within those utterances hypothesized to correspond to the query and scores indicating the system's confidence in each hypothesis.

Conventional KWS involves using an automatic speech recognition (ASR) system to decode the archives, constructing an inverted index from the resulting lattices and searching the query therein [2]. The inverted index is typically implemented as a timed factor finite-state transducer (FST) [3], [4], which is constructed offline and composed with an FST of the query. While this approach has proven quite successful, operating downstream of ASR has its pitfalls.

One such pitfall is that indexing any utterance involves full ASR decoding, which incurs a nontrivial computational cost. Furthermore, since the lattices of ASR systems can only contain tokens in the training vocabulary, ASR-based KWS systems with words as the ASR units cannot naturally retrieve out-of-vocabulary (OOV) queries, such as proper nouns and rare morphological inflections, and therefore have to resort to a host of other methods such as subword indexing or query expansion for retrieval [5], [6], [7], [8], [9]. Therefore, it is natural that recent research has focused on ASR-free KWS systems with a simpler indexing pipeline and natural handling of OOV queries [10], [11]. Since KWS is open-vocabulary and cannot therefore be cast as a keyword classification problem, these approaches typically feature a pair of encoders for speech and text trained to classify whether or not the text is spoken in speech segment under consideration.

In our recent conference article [1], we proposed a dual-encoder-based keyword search model trained to predict frame-wise probabilities of existence of a query in an utterance. It significantly outperformed other neural approaches in literature in terms of search accuracy, while also improving the search efficiency by using dot-products for search instead of more complicated feedforward neural networks. This article extends that preliminary work:

- We conduct more comprehensive analysis of the model with experiments measuring the impacts of various components and parameters of the models as well as the performance of the models for various kinds of queries.
- We show that with slight modifications, the model can be trained multilingually and that finetuning such a

Manuscript received 30 December 2022; revised 29 May 2023 and 12 July 2023; accepted 12 July 2023. Date of publication 2 August 2023; date of current version 11 August 2023. This work was supported in part by European Union's Horizon 2020 Project under Grant 870930 - WELCOME, in part by the Czech National Science Foundation (GACR) Project NEUREM3 under Grant 19-26934X, in part by the Turkish Directorate of Strategy and Budget through the TAM Project under Grant 2007K12-873, in part by the ROYAL Project under Grant 2019K12-149250, and in part by the Boğaziçi University Research Fund under Grant 16903. Computing on IT4I supercomputer was supported by the Czech Ministry of Education, Youth and Sports through the e-INFRA CZ under Grant ID:90140. An earlier version of this paper was presented at the 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, September 2021 [doi: 10.21437/Interspeech.2021-1399]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Rohit Prabhavalkar. *(Corresponding author: Bolaji Yusuf.)*

Bolaji Yusuf is with the Department of Electrical and Electronics Engineering, Boğaziçi University, 34342 Istanbul, Turkey, and also with the Faculty of Information Technology, Speech@FIT, Brno University of Technology, Brno 612 00, Czechia (e-mail: bolaji.yusuf@boun.edu.tr).

Jan Černocký is with the Faculty of Information Technology, Speech@FIT, Brno University of Technology, Brno 612 00, Czechia (e-mail: cernocky@fit.vut.cz).

Murat Saraçlar is with the Department of Electrical and Electronics Engineering, Boğaziçi University, 34342 Istanbul, Turkey (e-mail: saraclar@boun.edu.tr).

Digital Object Identifier 10.1109/TASLP.2023.3301239

multilingually-pretrained model significantly and consistently improves performance across target languages.

The rest of the article is organized as follows: Section II covers previous related work and highlights their differences and, where appropriate, their similarities to our method; Section III describes the proposed model; Section IV details the experiments conducted and discusses the results of those experiments; Section V concludes the article with a summary and future research directions.

II. Related Work

A. Open-Vocabulary ASR-Based KWS

Since ASR language models are trained with a limited number of words due to computational and data availability constraints, the ASR output is limited to the closed vocabulary used in training. However, user queries can—and often do—include words that are not part of this limited vocabulary. Therefore, dealing with the challenge of seamlessly searching such OOV queries has been studied extensively within the context of ASR-based KWS systems. The solutions in literature generally fall into two categories: using subword units and query expansion.

Based on the rationale that the words in a language—even OOV ones—can be composed from a limited set of subword units, using subword-based ASR has been the cornerstone of KWS in morphologically-rich languages [7], [12], as well as open-vocabulary search in other languages [6], [13], [14], [15]. Most works use linguistic units such as syllables, morphs and phones, while others use data-driven units like graphones and multigrams [12], [16], [17]. Although subword units increase the recall of the model, this comes at the cost of larger lattices which are more costly to index and search, as well as lower precision for in-vocabulary (IV) queries. Therefore, it is common to use a hybrid of word lattices for IV queries and subword lattices for OOV ones. This has the drawback of incurring the cost of double-indexing for every new utterance. This drawback can be partly reduced by converting word lattices into subwords ones for OOV search [5], [18], an approach limited in that it can only generate phone sequences which are substrings of some IV words.

Query expansion is an alternative approach to OOV search involving searching for phrases that are acoustically-similar to the query to account for ASR errors [9], [19], [20] This is typically implemented by composing a query FST with an FST of phone confusions, before composing the expanded query FST with the index. While query expansion can be used with subword indices, it can be leveraged to avoid double indexing by composing the expanded-query FST with the decoder vocabulary, resulting in acoustically-similar IV "proxy" queries which can be searched in a word-based index.

While these approaches alleviate ASR-based KWS' inability to handle OOV queries, they invariably further complicate either the indexing or the search for the already complex ASR-based KWS system. Our proposed method, on the other hand, not only offers simpler indexing and search than ASR-based KWS, it makes no distinction between IV and OOV queries.

B. End-to-End Keyword Search

Leveraging the ability of neural networks to model complex relationships, KWS traditionally comprising several disparate, separately-optimized, modules can now be simplified by formulating and optimizing appropriately designed neural architectures and objectives. One such simplification involves using end-to-end ASR models to construct the KWS index as in [21], [22], [23], [24]. While these works simplify ASR training, they still have complex KWS indexing pipelines since the simpler decoding algorithms for end-to-end ASR do not readily yield the timing and confidence information necessary for KWS. Therefore, another direction, in which our work falls, involves training a model able to avoid the ASR decoding entirely while indexing or searching.

The authors of [10] propose a Siamese neural architecture which jointly learns a distance metric for speech documents represented as phone posteriorgrams along with a query representation and conduct search with subsequence dynamic time warping (DTW). The method was extended in [25] to account for query dynamics and in [26] to learn better document representations. While it showed impressive search accuracy, especially for OOV queries, this approach is limited in practice by the significant computational cost of DTW.

The authors of [27] similarly proposed a Siamese architecture which learns text and speech representations for the related task of open-vocabulary hotword spotting.[1] Since the task there does not involve localization of the keywords, there is no associated cost of DTW. In [28], a meta-network was proposed that, for a given query, generates the parameters of a model to classify whether or not a speech segment contains that query. Thus the parameters of the model grow with the number of queries, which makes the model more suitable for limited, but adaptable, query sets as opposed to the unlimited vocabulary as in KWS. Moreover, like the model featured in [27], it also lacks the ability to localize the queries, which makes both approaches unsuitable for keyword search where the timestamps of each query's occurrence are required.

In [11], a model was proposed with a pair of encoders for computing fixed-length representations of speech utterances and text respectively, and a feedforward search network classifying whether or not the encoded text occurs in the encoded utterance. While the model was innovative in showing that the open-vocabulary search pipeline can be greatly simplified with this dual-encoder structure, it was limited to the utterance classification task where the probability of existence of a query was artificially set to 0.5 (by sampling positive and negative test utterances with equal probability at test time) and could not work in the highly imbalanced scenario of realistic KWS, where the number of negative trials far outnumber the number of positive ones. Moreover, since the speech encoder outputted a fixed-length representation of each utterance, the model could not temporally localize the keywords, although the authors did

[1] Hotword spotting involves spotting a limited set of phrases and has also been referred to as keyword spotting in some literature. We avoid that term to avoid confusion as some other literature use keyword spotting to refer to keyword search of the kind that we tackle.

experiment with a coarse form of localization by classifying whether the query occurs in the first or second half of the utterance. This method was improved in [29] by using better pretraining objectives, although it still had the limitations of [11] with regards to handling realistic KWS settings.

The most relevant related work to ours are those in [30] and [31] who contemporaneously with us proposed dual encoder architectures capable of open-vocabulary keyword search in realistic settings, complete with the ability to temporally localize the queries. Like us, they ensure that the speech encoders do not lose temporal information, and use forced alignment to obtain the query locations at training time.

In [30], the authors extend their prior work on closed-set keyword detection and localization [32] to cover open vocabularies. A feedforward network takes the speech and text encodings from a pair of convolutional and recurrent encoders and returns a vector to be compared with the text encoding to determine whether that text occurs in the utterance, as well as a pair of floating points corresponding to the hypothesized locations of the query. By directly predicting the temporal locations, they avoid the need to have any post processing step. This however comes at the cost of having to run the feedforward network for every query-utterance pair at search time, as opposed to the simpler matrix-vector product that we use for the query-utterance interaction. Therefore, the search can become orders of magnitude slower since each feedforward layer with output dimension of F is a matrix-matrix product having F times the cost of matrix-vector product.

In [31], the authors propose a very similar structure to our model. The main difference is that, instead of acoustic features, they use phonetic confusion networks output from an external ASR system as the speech representation. This allows language model information to be indirectly incorporated into the KWS model. However, it also means that creating the document representation requires full ASR decoding. Note that while we also use bottleneck features (BNF) obtained from an external ASR model, extracting BNF only incurs the cost of passing data through the acoustic model and not the costs of searching the ASR decoding graph.

C. Multilingual Data for KWS

Using multilingual data to improve KWS has been explored in prior work in the context of both ASR-based and ASR-free KWS. In the context of ASR-based KWS systems, a common recipe is to improve the ASR, and consequently KWS, in low-resource settings by training the acoustic model multilingually [33], [34], [35], [36], [37]. This generally entails sharing the lower layers of the acoustic model and using either a shared output layer [38], [39], [40] or separate output layers for each language [41], [42], [43], [44].

In [26], a joint metric and representation learning method is used to incorporate multilingual bottleneck features into dynamic-time-warping-based KWS. Multilingual bottleneck features and posteriorgrams have also been used for query-by-example (where both query and audio are spoken) with [45], [46], [47] or without [48] dynamic time warping. While these works use multilingual data to train the feature extractor for ASR-free search, they do not train the search model itself multilingually.

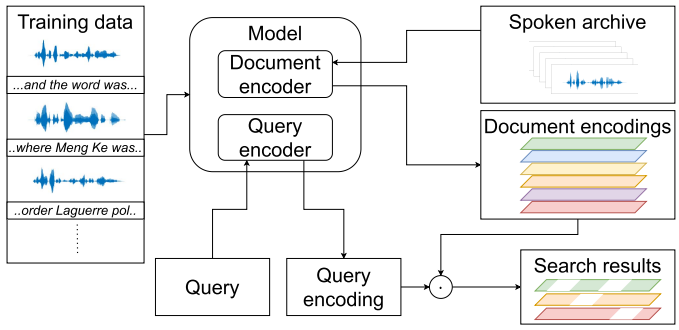

Fig. 1. Flowchart illustrating the proposed system. A keyword search model, comprising a query encoder and a document encoder, is trained on forced-aligned speech and text data. The document encoder is used to encode spoken archives for efficient search, and the query encoder converts each query into a vector form which is searched by computing frame-wise inner products with the document representation.

III. Methods

In this section, we describe the method we have proposed for keyword search. The method, illustrated in Fig. 1, involves a soft-indexing and matching approach. Where ASR-based KWS methods index a spoken archive by decoding it into a graph of symbolic units and conduct search by matching with a corresponding graph of the query, our approach uses a dense representation in a vector space and conducts search by matching the query with dot-products, for which modern CPUs, GPUs and linear algebra libraries have efficient implementations.

Sections III-A to III-D replicate the model definition, training and search procedures from [1] for completeness and readers' convenience. Section III-E describes multilingual training.

A. Problem Formulation

We formulate keyword search as the task of classifying whether a keyword occurs at any given location in the document. Given a query phrase $\mathbf{q} = (q_1, q_2, \ldots, q_K)$ where each q_k is a letter, and an utterance represented as a sequence of acoustic features $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{N_x})$, we seek the sequence of occurrence indicators $\mathbf{y}(\mathbf{q}, \mathbf{X}) = (y_1, \ldots, y_{N_x}) \in \{0, 1\}^{N_x}$ such that:

$$y_n = \begin{cases} 1, & \text{if } \mathbf{q} \text{ is spoken in } \mathbf{X} \text{ in a time span including } n \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Given a training set of utterances $\mathcal{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(S)}\}$ and query phrases $\mathcal{Q} = \{\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(L)}\}$, we train a neural network with parameters $\boldsymbol{\theta}$ to minimize the negative log-likelihood of the occurrence indicators:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{l=1}^{L} \sum_{s=1}^{S} \sum_{n=1}^{N_{x(s)}} -\log p_{\boldsymbol{\theta}}\left(y_n | \mathbf{q}^{(l)}, \mathbf{X}^{(s)}\right). \tag{2}$$

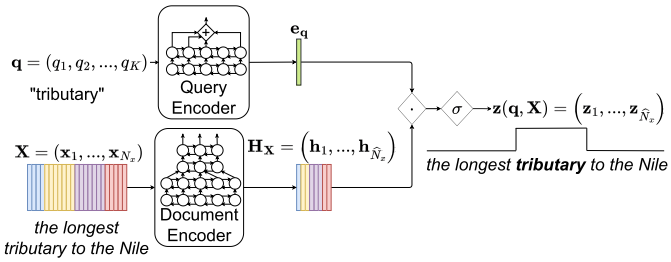

Fig. 2. Overview of the neural keyword search model. The document encoder is a bidirectional recurrent neural network which outputs a subsampled representation of the spoken document. The query encoder is a recurrent neural network which outputs a fixed length representation of the written query. A dot-product between the two representations gives logits which are passed through a logistic sigmoid to predict the likelihood of the query occurring at each subsampled time frame.

Note that training requires timing information of the phrases in the training set. We obtain it by training an HMM-GMM-based ASR system on the training data and using it to generate the required word-level forced alignments.

B. Model Definition

Our keyword search model, depicted in Fig. 2, comprises a recurrent document encoder and a recurrent query encoder. We conduct the search via a matrix-vector multiplication between the outputs of the encoders. We apply the logistic sigmoid function to the resulting vector of logits to obtain frame-wise posterior probabilities $p_{\boldsymbol{\theta}}(y_n|\dots)$ which we post-process to detect the locations of each keyword. Since the document encoder is intended to act as an offline indexer, while the query encoder must be called whenever a query is received, we ensure that the query encoder is a much smaller neural network than the document encoder.

1) Query Encoder: The input to the query encoder is a sequence of letters $\mathbf{q} = (q_1, \dots, q_K)$ that constitute the query and the output is a fixed-length representation $\mathbf{e_q} \in \mathbb{R}^D$. A trainable input embedding layer converts the sequence of letters into a sequence of vectors that are input into a stack of bidirectional gated recurrent unit (GRU) layers. The GRU outputs another sequence of vectors $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_K)$. The final query representation is then computed from the sum of these GRU output vectors along the sequence axis:

$$\mathbf{e_q} = \sum_{k=1}^{K} \mathbf{W}_1 \mathbf{v}_k + \mathbf{b_1}, \tag{3}$$

where $\mathbf{W}_1$ and $\mathbf{b}_1$ are the weight and bias of a trainable affine transform that changes the dimensionality of the query representation to ensure it matches the output of the document encoder. The affine transform also ensures that the dynamic range of the query encoding is not limited by the hyperbolic tangent output of the GRU to $(-1, 1)$.

We use summation instead of taking the output of the GRU at the final step, i.e., instead of setting $\mathbf{e_q} = \mathbf{W}_1 \mathbf{v}_K + \mathbf{b_1}$, because we empirically found it to be better. We also experimented with having a unidirectional query encoder but we found the bidirectional encoder to be superior.

2) Document Encoder: The input to the document encoder is the sequence of speech features $\mathbf{X}$ of length N_x. First, $\mathbf{X}$ is passed through a stack of bidirectional long short term memory (BLSTM) layers which output $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_{\hat{N}_x})$ of length $\hat{N}_x$. We down-sample the hidden representations between some of the BLSTM layers so that $\hat{N} = \lfloor \frac{N}{4} \rfloor$. This decreases the computational cost of storage and search, and we found empirically that it improves the search accuracy as it reduces the durations processed by higher layers, making it easier to model long-range dependencies. The final encoder output is then $\mathbf{H_X} = (\mathbf{h}_1, \dots, \mathbf{h}_{\hat{N}_x})$, such that for each $\hat{n}$:

$$\mathbf{h}_{\hat{n}} = \mathbf{W}_2 \mathbf{u}_{\hat{n}} + \mathbf{b}_2, \tag{4}$$

where $\mathbf{W}_2$ and $\mathbf{b}_2$ constitute an affine transformation similar to that at the output of the query encoder.

We choose to make the query encoder a GRU instead of an LSTM in order to reduce its computational cost since, unlike the document encoder which we expect to operate as an offline indexer, the query encoder would be run whenever a user queries the system. Moreover, in preliminary experiments, we found using a GRU as the query encoder performs as well as an LSTM with the same dimensions while having three-quarter the size and computational cost of the latter. However, using a GRU as the document encoder instead of an LSTM significantly degrades search performance.

3) Search Function: The search output is given by multiplying the encoded document matrix with the encoded query vector, followed by a logistic sigmoid, resulting in the desired vector of per-frame occurrence probabilities $\mathbf{z}(\mathbf{q}, \mathbf{X}) = (z_1, \dots, z_{\hat{N}}) \in (0, 1)^{\hat{N}}$, where $z_n := p_{\boldsymbol{\theta}}(y_n|\mathbf{q}, \mathbf{X})$:

$$\mathbf{z}(\mathbf{q}, \mathbf{X}) = \sigma(\mathbf{H}_\mathbf{X}^\top \mathbf{e_q}). \tag{5}$$

Since the document and query representations only interact through this product and are otherwise independent, we can pre-compute and store the encodings of the documents. Thus, at search time, only the cost of computing the query representation (from the much smaller query encoder) and the cost of the dot product is incurred.

As the only interaction between the query and a time frame of the document is this inner-product, which is independent of other time frames at the encoding level, the document encoding at each frame clearly needs to encode enough information to disambiguate between similar queries. For example, if the document contains the word "predict", the encodings of each frame corresponding to "pre-" need to be distinguishable from the encodings of "pre-" in, say, "prelude" or "preface". Similarly, the encodings corresponding to "-sion" in "confusion", need to be distinguishable from those in "television" and "intrusion". Therefore the document encoder needs to be bidirectional because the LSTM's forward direction is needed to disambiguate between shared suffixes, while the reverse direction is necessary for separating phrases with shared prefixes. In other words, the reverse direction controls when occurrence probabilities should

start spiking and the forward direction controls when they should stop spiking.

C. Model Training

Training the model involves optimizing (2) with gradient descent. However, doing so directly is impractical as computing the gradient involves summation over all phrases and utterances in the training set. A corpus of S utterances with W words each has $\mathcal{O}(SW^2)$ elements in the double summation. Therefore, we approximate this large sum with a smaller sum whose gradient approximates the gradient of the original and optimize the approximate sum instead:

$$\boldsymbol{\theta}^* \approx \arg\min_{\boldsymbol{\theta}} \sum_{l=1}^{L_b} \sum_{m=1}^{M} \sum_{n=1}^{N_{x(m)}} -\log p_{\boldsymbol{\theta}}\Big(y_n | \mathbf{q}^{(l)}, \mathbf{X}^{(m)}\Big), \quad (6)$$

where $L_b \ll L$ is the mini-batch size for each training step, and $M = |\mathcal{X}_{\mathbf{q}^{(l)}}| \ll S$ is the number of utterances sampled for each training phrase.

When looping over the phrases in the training data, we only consider such ls that $\mathbf{q}^{(l)}$ is either a unigram, bigram or trigram. When sampling the utterances $\mathcal{X}_{\mathbf{q}^{(l)}}$ for each step, we ensure that at least one of them is a "positive" utterance, i.e., it contains the training phrase $\mathbf{q}^{(l)}$, while the others are sampled truly randomly. While this constraint biases the gradient, without it, an overwhelming majority of mini-batches would be "negative", i.e., have all-zero labels, which would make optimization impossible.

For each query-utterance training pair $(\mathbf{q}^{(l)}, \mathbf{X}^{(m)})$, we minimize an objective function $J(\mathbf{q}^{(l)}, \mathbf{X}^{(m)})$ between the sigmoid outputs $\mathbf{z}(\mathbf{q}^{(l)}, \mathbf{X}^{(m)})$ and the labels $\mathbf{y}(\mathbf{q}^{(l)}, \mathbf{X}^{(m)})$:

$$J\Big(\mathbf{q}^{(l)}, \mathbf{X}^{(m)}\Big) = -\sum_{n=1}^{\hat{N}_{x(m)}} \Big(\mathbb{1}_{z_n > 1-\phi} \cdot (1-y_n)\log(1-z_n) + \mathbb{1}_{z_n < \phi} \cdot \lambda \cdot y_n \log z_n\Big), \quad (7)$$

where the labels have been down-sampled to match the output frame rate of the document encoder. This objective function extends the binary cross-entropy objective with the hyper-parameters λ and ϕ. When both are set to 1, the loss reduces to the binary cross-entropy. λ controls the relative importance of frames labeled 1 and frames labeled 0, i.e., the relative cost of misses to false detections; as λ increases, frames labeled 1 contribute more to the total loss. ϕ is a strictness term controlling the sensitivity of the loss function to easily classified frames; frames labeled 1 with sigmoid outputs above ϕ and frames labeled 0 with sigmoid outputs below $1-\phi$ do not contribute to the loss. This prevents the model from learning to better classify frames that are already well classified at the expense of learning to classify difficult frames.

D. Post-Processing for Keyword Search

Having obtained the vector of probabilities from (5), we still need to post-process them to obtain the timestamps in the document hypothesized to contain the query, and the corresponding confidence scores. The procedure is as follows:

1) We zero-out the probabilities (z_n) below some threshold α. This thresholding is a necessary first step because it is otherwise impossible to determine discrete values, $\{0, 1\}$, of y_n since sigmoid outputs are strictly non-zero. We treat α as a hyper-parameter which we tune on development sets to select from among $\{0.2, 0.4, 0.6\}$.
2) We pick the resulting "islands" of non-zero elements as our system hypotheses. Each hypothesis' confidence score is computed as the median probability in its interval. We also experimented with the mean and max operations but found median to be better.

E. Multilingual Training

Neural approaches, in KWS and otherwise, generally struggle in low-resource settings since they require large amounts of data to train. We therefore explore multilingual pretraining to improve the performance of the proposed model in low-resource settings. To do this, we pretrain the model with data pooled from several letter-based languages, and then finetune it on the target language.

During multilingual training, the entire model is shared by all languages, with the exception of the query encoder's input embedding layer. The model is trained with the same objective as in the monolingual setting. The negative utterances sampled for each training phrase can come from any language; we experimented with enforcing that the negative utterances come from the same language but found that this did not lead to consistent improvements.

When finetuning to a new language, we transfer only the pretrained document encoder and reinitialize the entire query encoder with random weights, then train the entire model on the target language's training data. We experimented with transferring the multilingually-pretrained query encoder as well with only the embedding layer reinitialized, but found this to result in significantly worse performance. We also experimented with initially freezing the transferred document encoder for a few epochs so that the query encoder gets trained to a reasonable degree before finetuning the whole model, but we found doing so also worsened performance and made training unstable.

IV. Experiments

In this section, we conduct experiments to analyze various components of the proposed model. First, we describe experiment setup: the datasets, metrics and default hyper-parameters for the proposed model. Then we analyze how various parameters affect the model performance. Finally, we analyze how the performance of the model changes with different keyword properties and compare and contrast to how those same keyword properties affect a conventional LVCSR-based model.

TABLE I
DISTRIBUTION OF IN-VOCABULARY AND OUT-OF-VOCABULARY QUERIES IN EACH TEST LANGUAGE

Query type	Pashto		Turkish		Zulu	
	Eval	Dev	Eval	Dev	Eval	Dev
IV	2044	1319	1173	203	1028	1193
OOV	326	438	452	80	380	793

A. Experimental Setup

1) Dataset: We conduct all experiments on data from the IARPA Babel corpus.[2] We use the limited language packs of Pashto, Turkish and Zulu for KWS experiments. Each of these comprises 10 hours of transcribed data for training, a 10-hour development set for hyper-parameter tuning and a 5-hour evaluation set.[3] Table I shows the distribution of queries for each language's development and evaluation sets. We use the transcribed data from 16 other Babel languages—totaling 170 hours—for multilingual pretraining of the KWS model in IV-D. We also use this dataset to train a multilingual acoustic model which contains a bottleneck layer for extracting 42-dimensional bottleneck features. The BNF extractor has language-specific output layers trained to predict the pretraining languages' senone labels, where each language's context dependent triphones clustered to around 2000 such senones.

2) Metrics: We report results in terms of the variants of the term weighted value (TWV) [49]. The actual term weighted value (ATWV) is a measure of weighted precision and recall at a single predefined threshold. For a set of queries $\mathcal{Q}$ and threshold ξ, the ATWV is defined:

$$\mathrm{ATWV}(\xi, \mathcal{Q}) = 1 - \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} (P_{miss}(q, \xi) + \beta P_{FA}(q, \xi)), \tag{8}$$

where $P_{miss}(q, \xi)$ is the probability of misses, $P_{fa}(q, \xi))$ is the probability false alarms, and β is a parameter that controls the relative weights of false alarms and misses. Following the NIST STD evaluations [50], we set $\beta = 999.9$. On the development set, we report the maximum term weighted value (MTWV) which is the TWV at the threshold that maximizes it. This threshold is then used to compute the actual term weighted value (ATWV) for the evaluation set.

Since different queries tend to have different score distributions and term weighted value requires setting a single global threshold, it is necessary to normalize scores per query. To this end, we adopt the keyword specific thresholding normalization method from [51].

We also report the optimum term weighted value (OTWV)—the upper-bound MTWV computed with query-specific thresholds. OTWV gives a measure of term weighted value without the effects of inter-query score mis-calibration.

Finally, we measure the supremum term weighted value (STWV)—the OTWV with the cost of false alarms set to zero. This gives a measure of overall recall. We however limit our use of STWV except when comparing two systems with similar ATWV because without such a constraint, STWV can be inflated by simply "detecting" the query everywhere.

Term weighted values so defined have a theoretical maximum value of 1. In our results, we multiply all term weighted values by 100 to get scores that can go up to 100.

3) Model Configuration and Default Hyper-Parameters: In general, we base our default architecture off that described in [1]. The document encoder has 6 BLSTM layers with 512 output units, followed by an affine projection layer with output dimension of 400. We apply dropout of 0.4 between successive BLSTM layers and down-sample by a factor of 2 after the first and fourth BLSTM layers. The query encoder has a 32-dimensional input embedding layer, 2 bidirectional GRU layers with 256 units each and a projection layer to match the output of the document encoder. This is almost identical to the setup in [1] except that we remove all Batchnorm layers, as we found finetuning multilingual models trained with Batchnorm to be unstable. Moreover, in the monolingual setting, we did not notice any performance deterioration from excising the Batchnorm layers.

We use graphemes as the query input representations instead of phonemes as they remove the need to train any grapheme-to-phoneme (G2P) converters to use for OOV queries. Moreover, we found them empirically superior to phonemes in terms of KWS performance. We do however concede that different representations would be required for languages that have orthographies with no phonetic correspondence.

Except where stated otherwise, we set $\lambda = 5$ and $\phi = 0.7$ in (7) and use 3 negative examples per positive at training time, i.e., $M = 4$ in Section III-C.

B. Effect of Loss Function Parameters

In this section, we analyze the impact of the hyper-parameters of the loss function, ϕ and λ from (7). Remember that λ is the weight given to positive training frames (frames which contain the training phrase), while ϕ controls the allowable margin beyond which no loss is incurred. When both values are set to 1, the objective becomes the classic binary cross-entropy objective. All models here use bottleneck features as input without any pretraining or speed perturbation.

First we vary the tolerance (ϕ) of the loss function with λ fixed to 5. Fig. 3 depicts the output of models trained with various values of ϕ on an example from the Turkish dev set. All settings of ϕ track the correct shape, outputting high values where the ground truth is 1 and low values where the ground truth is 0. However, the degrees with which they do so vary, with increasing ϕ expectedly resulting in more extreme separation of positives from negatives.

Fig. 4 shows the ATWV as ϕ changes. We observe that ATWV does not vary much with the choice of ϕ except when it is set to 1 where we observe significant ATWV degradation. This implies that the exact value of ϕ is not as important so long as we have some tolerance. Although we do not report those results here, we found that setting higher values of ϕ allows us to use lower

[2][Online]. Available: https://www.iarpa.gov/index.php/research-programs/babel

[3]The full evaluation set used for Babel challenges is 15 hours, of which the references for only 5 hours are openly available.

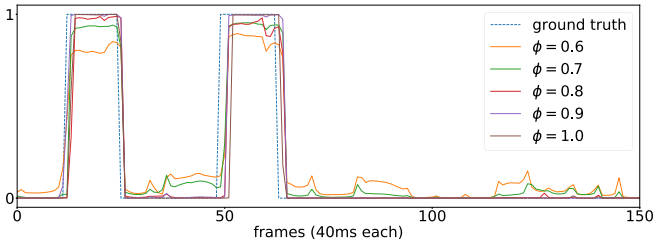

Fig. 3. Outputs from the KWS model for an example query-utterance pair from the Turkish development set under various settings of the training objective tolerance (ϕ).

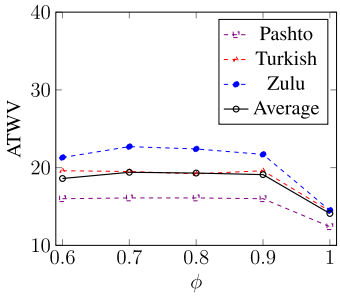

Fig. 4. ATWV on the evaluation sets as the strictness term ϕ in the training objective is varied.

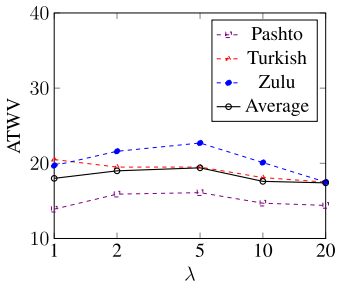

Fig. 5. ATWV on the evaluation sets as the weight of positive frames λ in the training objective is varied.

sigmoid thresholds than 0.5, resulting in higher STWV without sacrificing ATWV.

Fig. 5 shows the ATWV as λ is varied with ϕ fixed to 0.7. The ATWV increases with λ until around 5 where it peaks, and then falls off as λ is further increased. This indicates that the decrease in precision that accompanies increasing λ starts to hurt the overall ATWV. Thus, the correct setting of λ seems tied to the relative costs of false alarms to misses and must be set based on the task at hand. However, we note the recall, as measured by STWV—which is not in the figure—increases monotonically with λ, which is expected as increasing λ makes the training objective prioritize detecting positive locations over suppressing negative ones.

C. Effect of Down-Sampling

Our document encoder features a pair of down-sampling steps after the first and fourth BLSTM layers. Thus, the output

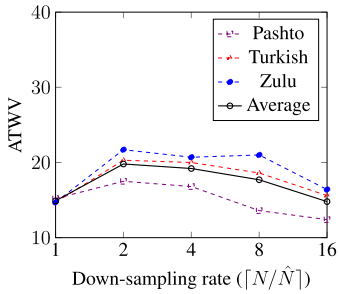

Fig. 6. ATWV on the evaluation sets as the down-sampling rate is varied.

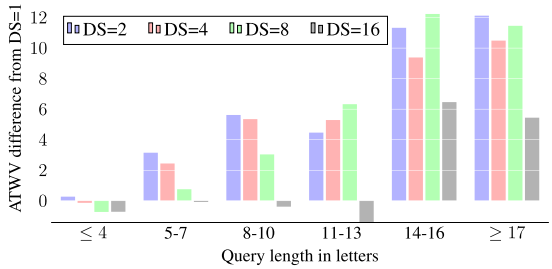

Fig. 7. Average difference in ATWV of systems with various down-sampling factors when compared to the system with no down-sampling. DS=* denotes down-sampling factor of *.

document encodings are down-sampled by a factor of 4, resulting in smaller document storage and computational requirements. In this section, we analyze the effect of down-sampling on keyword search performance.

Fig. 6 shows the ATWV as the total down-sampling factor varies between 1 (-), 2 (4), 4 (1, 4), 8 (1, 3, 4) and 16 (1, 3, 4, 5) where the numbers in the parentheses denote which layers' outputs are down-sampled by 2. We note that the ATWV improves as we introduce down-sampling, decreases slightly as the down-sampling factor is increased from 2 to 4, and starts to degrade upon further down-sampling—with a down-sampling factor of 16, the ATWV gets worse than not having any down-sampling at all.

Overall, we infer that having down-sampling—within some bounds—does not just maintain accuracy but actually improves it, while being faster. We ascribe this improvement to the difficulty of learning very long range dependencies within the model encodings. For instance, to correctly localize a query of length 500 ms at a frame rate of 10 ms, each BLSTM hidden state would have to contain information spanning at least 50 frames; after down-sampling by a factor of 2, this burden reduces to 25 frames etc. Beyond the optimal down-sampling rate, the search fidelity starts to degrade, ostensibly due to the excessive loss in resolution.

These arguments are supported by the results in Fig. 7 which shows the ATWV improvements or degradation for queries of different length as the down-sampling rate is varied. The systems with down-sampling generally perform better as the query length increases supporting the hypothesis that without down-sampling, the system struggles to model long-term dependencies. On the other hand, as the rate of down-sampling

TABLE II
TERM WEIGHTED VALUES ON THE VARIOUS DEVELOPMENT AND EVALUATION SETS

Language	System	Dev MTWV	Eval ATWV
Pashto	MFCC	4.9	6.9
	+sp	7.9	10.6
	+Pretrain	**9.4**	**11.4**
	BNF	11.7	15.2
	+sp	14.5	18.1
	+Pretrain	**16.3**	**20.3**
Turkish	MFCC	18.1	7.9
	+sp	24.4	13.5
	+Pretrain	**31.6**	**18.2**
	BNF	29.1	17.8
	+sp	31.0	23.7
	+Pretrain	**37.6**	**24.7**
Zulu	MFCC	4.7	6.6
	+sp	9.9	12.3
	+Pretrain	**15.9**	**17.7**
	BNF	16.4	17.9
	+sp	23.5	25.3
	+Pretrain	**26.4**	**26.8**

"+sp"denotes the use of speed-perturbation and "+Pretrain" indicates multilingual pretraining.
The bold values indicate the best performance.

is increased, the model struggles with detecting shorter queries (see for instance the performance with down-sampling rate of 16), supporting the idea that loss of resolution eventually limits feasible amount of down-sampling.

D. Impact of Multilingual Pretraining

In this section, we experiment with methods to increase the effective training data by using speed perturbation [52] and multilingual pretraining.

Table II shows the effect of speed perturbation on KWS performance. As is common practice, we create two extra copies of the training data by perturbing the speaking rates by factors of 0.9 and 1.1 respectively. The bottleneck feature (BNF) rows of the table replicate the best results reported in [1]. We observe that speed perturbation leads to significant improvements regardless of input feature, with much higher relative improvements on MFCC. Therefore, in subsequent experiments, we use speed perturbation by default.

Table II also shows the effect of multilingual pretraining on KWS performance. Here, we only use speed perturbation when finetuning, and not when pretraining in order to limit computational costs. We observe significant improvements on the baseline, on top of the improvements from speed perturbation, regardless of input feature type. This is despite the fact that the BNF already contain multilingual information.

E. Effect of the Ratio of Negative to Positive Training Utterances

In Section III-C, for each training phrase, we sample M utterances, one of which is the utterance from which the current training phrase is taken. The other $M-1$ "negative" utterances

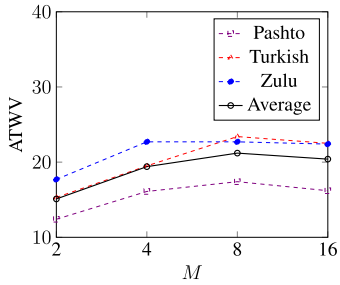

Fig. 8. ATWV on the evaluation sets as the ratio of negative training utterances is varied.

TABLE III
TERM WEIGHTED VALUE AS THE NUMBER OF UTTERANCES PER TRAINING STEP IS VARIED WITH OR WITHOUT MULTILINGUAL PRETRAINING

Language	System	M	Dev MTWV	Eval ATWV
Pashto	BNF + sp	4	14.5	18.1
	BNF + sp	8	**16.1**	**20.2**
	+Pretrain	4	16.3	20.3
	+Pretrain	8	**17.5**	**20.7**
Turkish	BNF + sp	4	31.0	23.7
	BNF + sp	8	**32.6**	**24.8**
	+Pretrain	4	**37.6**	24.7
	+Pretrain	8	35.0	**27.0**
Zulu	BNF + sp	4	23.5	25.3
	BNF + sp	8	**24.1**	25.3
	+Pretrain	4	26.4	26.8
	+Pretrain	8	**27.7**	**27.2**

The bold values indicate the best performance.

are sampled randomly. In the experiments so far, we have set $M = 4$.

Fig. 8 shows the result of varying M with BNF-based models. Increasing M has two obvious effects on the optimization; it reduces the approximation error due to the sampling process and it inadvertently up-weights the contribution of negative samples to the loss function (effectively reduces λ). We argue that the ATWV improvements we observe are a result of the former, since, as we have already seen in Section IV-B (and Fig. 5), decreasing λ does not improve the ATWV and doubling (or even quadrupling it) does not degrade the term weighted value to the extent that setting $M = 1$ does. However, the latter has an impact on STWV, which decreases strictly as M increases although we do not report it here to reduce clutter. This is due to a "broken-clock" effect, where models trained with lower M (similar to models trained with smaller λ) have higher recall simply by virtue of returning far more hits, whether spurious or correct, as a consequence of seeing a lower number and diversity of negative training utterances. Finally, we see that with $M = 8$, we get a good enough approximation and that increasing M further does not lead to significant improvements in term weighted value–even slightly degrading the performance for Pashto.

Table III shows the results of increasing M from 4 to 8 for the speed-pertubed BNF models with and without pretraining. We observe that increasing M generally improves the performance, with the only exception being the Turkish Dev MTWV with

TABLE IV
EVALUATION SET IV AND OOV TERM WEIGHTED VALUES FOR THE PROPOSED SYSTEM, A HYBRID ASR-BASED KWS SYSTEM AND THE FUSION OF BOTH SYSTEMS

Metric	System	Pashto		Turkish		Zulu		Average	
		IV	OOV	IV	OOV	IV	OOV	IV	OOV
ATWV	ASR-based	35.0	12.7	46.2	27.9	37.6	23.5	39.6	21.4
	Proposed	22.0	13.5	29.0	26.7	27.9	25.3	26.3	21.8
	ASR-based + Proposed	**38.0**	**18.9**	**50.3**	**35.0**	**41.3**	**33.2**	**43.2**	**29.0**
OTWV	ASR-based	51.5	24.4	62.5	41.9	50.8	36.5	54.9	34.3
	Proposed	39.3	28.4	46.1	45.4	41.3	39.7	42.2	37.8
	ASR-based + Proposed	**55.1**	**36.2**	**65.9**	**54.0**	**56.3**	**48.8**	**59.1**	**46.3**

The bold values indicate the best performance.

pretraining and the Zulu Eval ATWV without pretraining. We reiterate here that the variation in M is only done for the finetuning. Pretraining is always done with $M = 4$. We do not experiment with higher M for pretraining due to the computational costs that would be involved.

F. Performance of Queries With Various Properties

In this section, we analyze the performance of the proposed model on queries of various properties. Specifically, we analyze how the performance of the model changes depending on whether the query in question is in-vocabulary or out-of-vocabulary, as well as how the performance changes with length of the query. We compare the result to how the same factors affect conventional ASR-based systems.

For this comparison, we build a baseline KWS system based on a TDNN-based [53] hybrid ASR model. To have a fair comparison, the TDNN acoustic model is pretrained on the same data we use for multilingual pretraining of the proposed model, and finetuned on each target language with speed perturbation applied for both pretraining and finetuning. We use a word-subword hybrid index where IV queries are searched in a word-based index while OOV queries are searched in a syllabic one. Both the word and the subword lattices are obtained using respective word and syllabic trigram Kneser-Ney-smoothed [54] language models which we found to perform better than higher order language models in this low-resource setting. We use the official lexicon to get IV word pronunciations and to train a Sequitur grapheme-to-phoneme converter [55] for getting OOV pronunciations.

1) IV vs OOV Queries: Table IV shows the performance of the ASR-based KWS system and proposed system on IV and OOV queries. We find that the proposed model has much smaller discrepancy between IV and OOV performance than the ASR-based system. With the exception of Pashto, the ATWV difference between IV and OOV queries is always under 3 points.

In terms of ATWV, the proposed system slightly outperforms the ASR-based baseline system for OOV queries but lags it significantly for IV queries. This is somewhat expected, especially in the low-resource settings, as the baseline has a strong guide for IV queries in the word language model, an advantage that is diminished for OOV queries, even with a subword language model and index.

We also report the OTWV, which slightly favors the proposed model compared to the baseline. For IV terms, the relative average degradation between the proposed system and the baseline is reduced from 34% in ATWV to 23% OTWV. For OOV terms, the relative improvement is increased from 2% to 10%. This suggests that although the overall trends stay the same, part of the difference in performance is due to the score normalization being better suited to the baseline rather than qualitative differences in the model.

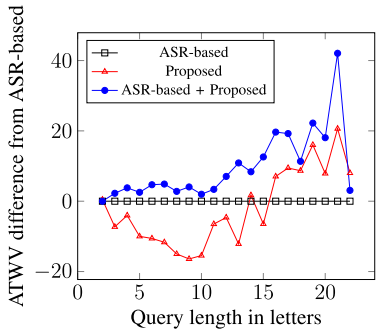

Fig. 9. Average difference in ATWV of various systems when compared to the ASR-based baseline as query length varies.

Finally, we report the results of fusion, where we combine the hitlists from both systems by weighted summation of scores with weights tuned on the development sets. We find that the performance of the baseline is significantly improved by score fusion; around 9% on IV and 36% on OOV ATWV, with similar improvements in OTWV. This underscores the potential benefit of deploying both systems in tandem where computationally feasible.

2) Query Length: We have seen that while our approach does not distinguish much between IV and OOV queries, its IV performance trails that of a strong ASR-based KWS baseline. To find the root of this difference, we compare the performance of the systems on queries of various length.

Figs. 9 and 10 show the average (across languages) difference in ATWV and OTWV respectively between the proposed model and the baseline. Negative values indicate query lengths for which the baseline is better and positive values indicate query lengths for which the proposed system is better. Note that although it is not conveyed in this figures, all systems—including the baseline—have better performance as the query length increases. Even so, we find that in general, the baseline performs better for shorter queries, both systems perform comparably for mid-length queries, and the proposed system performs better for long queries (above 15 characters). Finally, we observe that fusion outperforms the baseline regardless of query length.

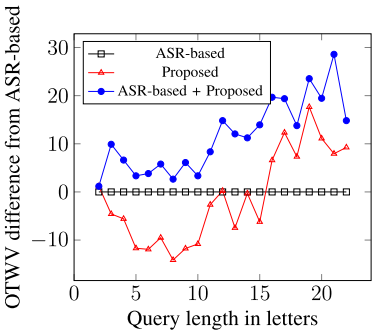

Fig. 10. Average difference in ATWV of various systems when compared to the ASR-based baseline as query length varies.

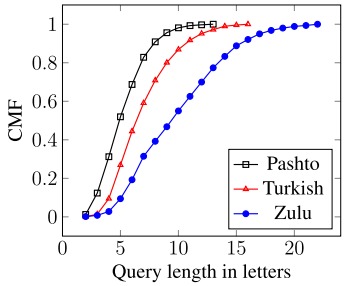

Fig. 11. Cumulative distributions of query lengths for each evaluation set.

These results are not surprising as short queries are both easier to miss and easier to falsely spot, and the ASR-based system being able to leverage contextual information provided by the language model helps it better find short queries. They also explain why our model performs significantly worse for Pashto than the other two languages. As Fig. 11 shows, the queries in the Pashto evaluation set are quite short, with over 50% of queries being less than 5 letters long.

V. Conclusion

In this article, we extend our recent work [1] on end-to-end keyword search. Our model provides a simplified pipeline for keyword search, comprising a pair of encoders: one for encoding spoken archives, and a second for text queries. Keyword search is then effected in the resulting vector-spaces by computing inner-products between the document and query encodings. Compared to [1], in this work, we explore multilingual pretraining and conduct thorough analyses of various components, strengths and weaknesses of the proposed model. Our experiments show that:

- Our model significantly benefits from multilingual pretraining, with considerable increase in term weighted values without making the model more complex.
- Our model retrieves out-of-vocabulary queries almost as well as it retrieves in-vocabulary ones, and slightly outperforms a strong ASR-based keyword search system on OOV queries.
- The simplicity of our model comes at the cost of worse performance than the ASR-based system on IV queries and short queries—two query types where the ASR-based system benefits from contextual clues provided by the language model.
- Our approach is complementary with the ASR-based system, and combining the two improves the performance of the ASR-based system, even for IV queries and short queries.

Our model has two main limitations, which provide avenues for future work:

- The model does not use linguistic context information, making it worse that the ASR-based system on IV queries and short queries. It would therefore be worth exploring methods to incorporate external text without complicating the inference, similar to joint text and speech training for end-to-end ASR [56], [57], [58].
- The document representation grows linearly with the size of the archive. Although inner-products can be efficiently computed even for very large indices, storing those indices in memory becomes untenable for archives larger than a few hundred hours. Potential solutions include quantization techniques such as binary hashing [59] and product-quantization [60] to reduce both the storage and search computation costs.

References

[1] B. Yusuf, A. Gok, B. Gundogdu, and M. Saraclar, "End-to-end open vocabulary keyword search," in *Proc. Interspeech*, 2021 pp. 4388–4392.
[2] C. Chelba, T. J. Hazen, and M. Saraçlar, "Retrieval and browsing of spoken content," *IEEE Signal Process. Mag.*, vol. 25, no. 3, pp. 39–49, May 2008.
[3] D. Can and M. Saraçlar, "Lattice indexing for spoken term detection," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 8, pp. 2338–2347, Nov. 2011.
[4] C. Allauzen, M. Mohri, and M. Saraclar, "General indexation of weighted automata: Application to spoken utterance retrieval," in *Proc. Workshop Interdiscipl. Approaches Speech Indexing Retrieval HLT-NAACL*, 2004, pp. 33–40.
[5] M. Saraçlar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *Proc. Hum. Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2004, pp. 129–136.
[6] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 615–622.
[7] S. Parlak and M. Saraçlar, "Performance analysis and improvement of Turkish broadcast news retrieval," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 3, pp. 731–741, Mar. 2012.
[8] I. Szöke, M. Fapšo, and L. Burget, "Hybrid word-subword decoding for spoken term detection," in *Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 42–48.
[9] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 416–421.
[10] B. Gündoğdu, B. Yusuf, and M. Saraçlar, "Joint learning of distance metric and query model for posteriorgram-based keyword search," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1318–1328, Dec. 2017.
[11] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end ASR-free keyword search from speech," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1351–1359, Dec. 2017.
[12] V. T. Turunen and M. Kurimo, "Speech retrieval from unsegmented finnish audio using statistical morpheme-like units for segmentation, recognition, and retrieval," *ACM Trans. Speech Lang. Process.*, vol. 8, pp. 1–25, 2008.
[13] K. Ng and V. W. Zue, "Subword-based approaches for spoken document retrieval," *Speech Commun.*, vol. 32, no. 3, pp. 157–186, 2000.
[14] D. Karakos and R. Schwartz, "Subword and phonetic search for detecting out-of-vocabulary keywords," in *Proc. Interspeech*, 2014, pp. 2469–2473.
[15] H. Su, V. T. Pham, Y. He, and J. Hieronymus, "Improvements on transducing syllable lattice to word lattice for keyword search," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4729–4733.

[16] E. Arisoy, D. Can, S. Parlak, H. Sak, and M. Saraçlar, "Turkish broadcast news transcription and retrieval," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 5, pp. 874–883, Jul. 2009.

[17] Y. He et al., "Using pronunciation-based morphological subword units to improve OOV handling in keyword search," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 24, no. 1, pp. 79–92, Jan. 2016.

[18] D. Karakos, I. Bulyko, R. Schwartz, S. Tsakalidis, L. Nguyen, and J. Makhoul, "Normalization of phonetic keyword search scores," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 7834–7838.

[19] D. Can, E. Cooper, A. Sethy, C. White, B. Ramabhadran, and M. Saraclar, "Effect of pronounciations on OOV queries in spoken term detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2009, pp. 3957–3960.

[20] M. Saraçlar et al., "An empirical study of confusion modeling in keyword search for low resource languages," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 464–469.

[21] Y. Bai et al., "End-to-end keywords spotting based on connectionist temporal classification for Mandarin," in *Proc. IEEE 10th Int. Symp. Chin. Spoken Lang. Process.*, 2016, pp. 1–5.

[22] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 5280–5284.

[23] G.-X. Shi, W.-Q. Zhang, G.-B. Wang, J. Zhao, S.-Z. Chai, and Z.-Y. Zhao, "Timestamp-aligning and keyword-biasing end-to-end ASR front-end for a KWS system," *EURASIP J. Audio, Speech Music Process.*, vol. 2021, pp. 1–14, 2021.

[24] R. Yang, G. Cheng, H. Miao, T. Li, P. Zhang, and Y. Yan, "Keyword search using attention-based end-to-end ASR and frame-synchronous phoneme alignments," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 3202–3215, 2021.

[25] B. Gundogdu, B. Yusuf, and M. Saraclar, "Generative RNNs for OOV keyword search," *IEEE Signal Process. Lett.*, vol. 26, no. 1, pp. 124–128, Jan. 2019.

[26] B. Yusuf, B. Gundogdu, and M. Saraclar, "Low resource keyword search with synthesized crosslingual exemplars," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 7, pp. 1126–1135, Jul. 2019.

[27] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Proc. Interspeech*, 2019 pp. 3362–3366, doi: 10.21437/Interspeech.2019-1846.

[28] T. Bluche and T. Gisselbrecht, "Predicting detection filters for small footprint open-vocabulary keyword spotting," in *Proc. Interspeech*, 2020, pp. 2552–2556.

[29] Z. Zhao and W.-Q. Zhang, "End-to-end keyword search based on attention and energy scorer for low resource languages," in *Proc. Interspeech*, 2020, pp. 2587–2591.

[30] T. S. Fuchs, Y. Segal, and J. Keshet, "CNN-Based spoken term detection and localization without dynamic programming," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 6853–6857.

[31] J. Švec, L. Šmídl, J. V. Psutka, and A. Pražák, "Spoken term detection and relevance score estimation using dot-product of pronunciation embeddings," in *Proc. Interspeech*, 2021, pp. 4398–4402.

[32] Y. Segal, T. S. Fuchs, and J. Keshet, "SpeechYOLO: Detection and localization of speech objects," in *Proc. Interspeech*, 2019, pp. 4210–4214, doi: 10.21437/Interspeech.2019-1749.

[33] K. M. Knill, M. J. F. Gales, S. P. Rath, P. C. Woodland, C. Zhang, and S. Zhang, "Investigation of multilingual deep neural networks for spoken term detection," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2013, pp. 138–143.

[34] Z. Tüske, P. Golik, D. Nolden, R. Schlüter, and H. Ney, "Data augmentation, feature combination, and multilingual neural networks to improve ASR and KWS performance for low-resource languages," in *Proc. Interspeech*, 2014, pp. 1420–1424.

[35] J. Cui et al., "Multilingual representations for low resource speech recognition and keyword search," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2015, pp. 259–266.

[36] M. Karafíat et al., "2016 BUT Babel system: Multilingual BLSTM acoustic model with i-Vector based adaptation," in *Proc. Interspeech*, 2017, pp. 719–723.

[37] T. Sercu et al., "Network architectures for multilingual speech representation learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 5295–5299.

[38] S. Thomas, S. Ganapathy, and H. Hermansky, "Cross-lingual and multistream posterior features for low resource LVCSR systems," in *Proc. Interspeech*, 2010, pp. 877–880.

[39] 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, pp. 2586–2589.

[40] N. T. Vu, D. Imseng, D. Povey, P. Motlicek, T. Schultz, and H. Bourlard, "Multilingual deep neural network based acoustic modeling for rapid language adaptation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 7639–7643.

[41] S. Scanzio, P. Laface, L. Fissore, R. Gemello, and F. Mana, "On the use of a multilingual neural network front-end," in *Proc. Interspeech*, 2008, pp. 2711–2714.

[42] F. Grézl, M. Karafiát, and M. Janda, "Study of probabilistic and Bottle-Neck features in multilingual environment," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011, pp. 359–364.

[43] K. Veselý, M. Karafiát, F. Grézl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Proc. IEEE Workshop Spoken Lang. Technol.*, 2012, pp. 336–341.

[44] G. Heigold et al., "Multilingual acoustic models using distributed deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 8619–8623.

[45] L. J. Rodriguez-Fuentes, A. Varona, M. Penagarikano, G. Bordel, and M. Diez, "High-performance query-by-example spoken term detection on the SWS 2013 evaluation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 7819–7823.

[46] I. Szöke, M. Skácel, J. Černocký, and L. Burget, "Coping with channel mismatch in query-by-example - but QUESST 2014," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 5838–5842. [Online]. Available: https://www.fit.vut.cz/research/publication/10956

[47] H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li, "Multitask feature learning for low-resource query-by-example spoken term detection," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1329–1339, Dec. 2017.

[48] D. Ram, L. Miculicich, and H. Bourlard, "Neural network based end-to-end query by example spoken term detection," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 1416–1427, 2020.

[49] NIST, "The spoken term detection (STD) 2006 evaluation plan." Accessed: Aug. 7, 2023. [Online]. Available: https://catalog.ldc.upenn.edu/docs/LDC2011S02/std06-evalplan-v10.pdf

[50] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Proc. ACM SIGIR Workshop Searching Spontaneous Conversational Speech*, 2007, pp. 51–57.

[51] D. R. Miller et al., "Rapid and accurate spoken term detection," in *Proc. Interspeech*, 2007, pp. 314–317.

[52] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 3586–3589.

[53] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 3214–3218.

[54] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Comput. Speech Lang.*, vol. 8, no. 1, pp. 1–38, 1994.

[55] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, no. 5, pp. 434–451, 2008.

[56] P. Wang, T. N. Sainath, and R. J. Weiss, "Multitask training with text data for end-to-end speech recognition," in *Proc. Interspeech*, 2021, pp. 2566–2570.

[57] B. Yusuf, A. Gandhe, and A. Sokolov, "USTED: Improving ASR with a unified speech and text encoder-decoder," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 8297–8301. [Online]. Available: https://www.fit.vut.cz/research/publication/12784

[58] S. Thomas, B. Kingsbury, G. Saon, and H.-K. J. Kuo, "Integrating text inputs for training and adapting RNN transducer ASR models," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 8127–8131.

[59] A. Jansen and B. V. Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011, pp. 401–406.

[60] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.