End-to-End Open Vocabulary Keyword Search

Bolaji Yusuf[1,2], *Alican Gok*[1], *Batuhan Gundogdu*[1,3], *Murat Saraclar*[1]

[1]Bogazici University, Department of Electrical and Electronics Engineering, Turkey
[2]Brno University of Technology, Faculty of Information Technology, Speech@FIT, Czechia
[3]University of Chicago, Department of Radiology, IL, USA

{bolaji.yusuf,alican.gok,batuhan.gundogdu,murat.saraclar}@boun.edu.tr

Abstract

Recently, neural approaches to spoken content retrieval have become popular. However, they tend to be restricted in their vocabulary or in their ability to deal with imbalanced test settings. These restrictions limit their applicability in keyword search, where the set of queries is not known beforehand, and where the system should return not just whether an utterance contains a query but the exact location of any such occurrences. In this work, we propose a model directly optimized for keyword search. The model takes a query and an utterance as input and returns a sequence of probabilities for each frame of the utterance of the query having occurred in that frame. Experiments show that the proposed model not only outperforms similar end-to-end models on a task where the ratio of positive and negative trials is artificially balanced, but it is also able to deal with the far more challenging task of keyword search with its inherent imbalance. Furthermore, using our system to rescore the outputs an LVCSR-based keyword search system leads to significant improvements on the latter.

Index Terms: keyword search, spoken term detection

1. Introduction

Many technologies have emerged to facilitate browsing the vast amounts of spoken content that have become commonplace. One of these is keyword search (KWS), in which a user-specified keyword (also referred to as query or term) is searched within an audio archive (or document), and the locations, if any, are returned along with confidence scores.

The traditional approach to KWS entails using an LVCSR system to transcribe the archive, and then conducting text retrieval on the LVCSR output [1]. This approach has been successfully applied for retrieval on broadcast news [2, 3], video lectures [4] and web videos [5].

A limitation of the LVCSR approach is the inability to retrieve words that are not seen at training time. Retrieving such out-of-vocabulary (OOV) terms requires extra engineering. The most common approach to open-vocabulary search is to use subword units instead of words as the ASR output, thereby exchanging lattice compactness for coverage [6, 7, 8]. Another approach is to use phone-confusion statistics to convert an OOV term into a similar sounding in-vocabulary (IV) one and then searching for this *proxy* term instead [9, 10]. More recently, dynamic time warping (DTW)-based methods which completely eschew LVCSR have been shown to outperform both [11, 12].

Recently, several neural architectures have been proposed for hot-word spotting. In [13], a neural network is used to predict keywords from spectral features. CNN-based systems have also been applied for supervised [14] and weakly-supervised [15] keyword spotting. In [16], a system based on rank-1 encoder-decoder networks was proposed for end-to-end streaming keyword spotting. These systems have the limitation of requiring a pre-specified set of keywords.

End-to-end networks that bypass LVCSR while also being able to deal with open-vocabulary applications have also been studied. In [17], a pair of auto-encoders are pre-trained to encode the query and document, and a feed-forward network is used to predict whether the query encoding occurs in the document encoding. Other similar works use acoustic word embeddings for metric-based keyword search [18, 19]. In [20], the authors improve on [17] by improving the pre-training and also by leveraging temporal information available in the form of forced-alignments. While these methods have been shown to perform well for well-balanced tasks such as classifying whether two embeddings belong to the same term or whether an utterance contains a query, they have limited applicability for actual keyword search, which is extremely imbalanced in that most locations do *not* contain the keyword; for every positive trial, there are thousands of negative ones.

In this paper, we propose an end-to-end keyword search model that takes a query as input and returns frame-level probabilities of the query occurring in the document. Unlike other neural KWS architectures, we do not need to construct positive-negative samples for testing but can directly search on whole utterances. We show that our model outperforms other end-to-end KWS systems in segment-level classification, performs well in realistic KWS settings, and is able to improve the performance of a competitive LVCSR-based system by rescoring.

2. Model

We formulate keyword search as the task of classifying whether a keyword occurs at any given location in the document. This formulation – as opposed to, say, a regression task predicting the boundaries of a hypothesized hit – is very well suited to being used in combination with other keyword search systems since it allows us to get a keyword search result for any possible subsequence of the document.

Given a query phrase $\mathbf{q}$ and an utterance represented as a sequence of frames $\mathbf{X} = \left(\mathbf{x}_1, \ldots, \mathbf{x}_N\right)$, we seek the sequence $\mathbf{y}(\mathbf{q}, \mathbf{X}) = (y_1, \ldots, y_N) \in \{0, 1\}^N$ such that:

$$y_n = \begin{cases} 1, & \text{if } \mathbf{q} \text{ occurs at frame } n \text{ of } \mathbf{X} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Note that although we elect not to write it explicitly to reduce clutter, each y_n is a function of $\mathbf{q}$ and $\mathbf{X}$. We can then train a neural network with parameters $\boldsymbol{\theta}$ to obtain:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{q}} \sum_{\mathbf{X}} \sum_{n} \log p_{\boldsymbol{\theta}}(y_n | \mathbf{q}, \mathbf{X}), \tag{2}$$

where the summations are over all the phrases, utterances and time frames in the training set.

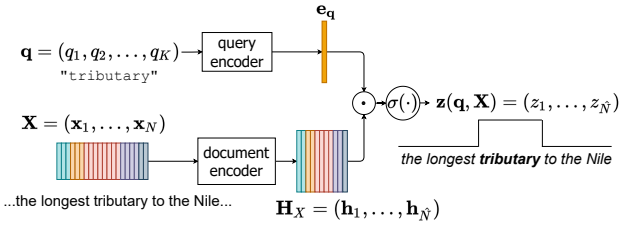

Figure 1: *Overview of the proposed model.*

2.1. Model definition

Our keyword search model, depicted in Figure 1, comprises a recurrent document encoder and a recurrent query encoder. We conduct the search via a matrix-vector multiplication of the encoder outputs which results in a vector of logits for each time frame. The logistic sigmoid is then used to compute the posterior probabilities $p_{\boldsymbol{\theta}}(y_n|\ldots)$ which we post-process to detect the locations of each keyword.

2.1.1. Query encoder

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

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

where $\mathbf{W}_1$ and $\mathbf{b}_1$ are the weight and bias of an affine transform that changes the dimensionality of the query representation to ensure it matches the output of the document encoder.

We experimented with using the output of the GRU at final frame ($\mathbf{v}_K$) instead of the summation in (3), but found that the summation performed better. We also experimented with having a unidirectional query encoder but we found the bidirectional encoder to be better.

2.1.2. Document encoder

The input to the document encoder is the sequence of speech features $\mathbf{X}$ of length N. First, $\mathbf{X}$ is passed through a stack of BiLSTM layers which output $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{\hat{N}})$ of length $\hat{N}$. The final encoder output $\mathbf{H_X} \in \mathbb{R}^{\hat{N} \times D}$ is then:

$$\mathbf{H_X} = \mathbf{W}_2\mathbf{U} + \mathbf{b}_2, \tag{4}$$

where $\mathbf{W}_2$ and $\mathbf{b}_2$ constitute an affine transformation similar to that at the output of the query encoder, and the addition is done by broadcasting $\mathbf{b}_2$ across the temporal axis.

We down-sample the hidden representations between some of the BiLSTM layers (so that $\hat{N} = \lfloor \frac{N}{s} \rfloor$ for some s). This decreases the computational cost of the search, and we found empirically that it improves the search accuracy.

2.1.3. Search function

The search output is computed by taking a matrix-vector product of the encoder outputs, followed by a logistic sigmoid to compute the desired vector of per-frame posterior probabilities $\mathbf{z}(\mathbf{q}, \mathbf{X}) = (z_1, \ldots, z_{\hat{N}}) \in (0, 1)^{\hat{N}}$ thus:

$$\mathbf{z}(\mathbf{q}, \mathbf{X}) = \sigma(\mathbf{H_X}\mathbf{e_q}). \tag{5}$$

Note that the document and query representations only interact through this product and are otherwise independent. Therefore, we can amortize the cost of search by pre-computing the document representation. For each query, we can then compute its representation and take the product.

2.2. Model training

Our training procedure involves optimizing a modified form of (2). The first two summations (2) are over all phrases and utterances in the training set. For a corpus of $\mathcal{U}$ utterances with $\mathcal{W}$ words each, there are $\mathcal{O}(\mathcal{U}\mathcal{W}^2)$ elements in the double summation. Therefore, we make the following modifications to reduce the training cost:

1. We limit training phrases to unigrams, bigrams and trigrams.
2. At each training step, we sample a batch $\mathcal{Q}$ of such phrases for the first summation instead of the whole set.
3. For each training phrase $\mathbf{q} \in \mathcal{Q}$, we sample $\mathcal{X}_\mathbf{q}$, a set of $\mathcal{M} \ll \mathcal{U}$ utterances, at least one of which contains $\mathbf{q}$.

When sampling, we consider different examples of the same phrase to be different tokens, so that more frequently occurring phrases are naturally sampled more often. The utterances $\mathcal{X}_\mathbf{q}$ are re-sampled at each training step.

For each query-utterance training pair $(\mathbf{q}, \mathbf{X})$, we define a loss function between the sigmoid outputs $\mathbf{z}(\mathbf{q}, \mathbf{X})$ and the (down-sampled) labels $\mathbf{y}(\mathbf{q}, \mathbf{X})$:

$$J_s(\mathbf{q}, \mathbf{X}) = -\sum_{n=1}^{\hat{N}} \Big(\mathbb{1}_{z_n > 1-\phi} \cdot (1 - y_n) \log(1 - z_n) + \mathbb{1}_{z_n < \phi} \cdot \lambda \cdot y_n \log z_n \Big). \tag{6}$$

λ is a hyper-parameter that controls the relative weight of frames labeled 1 to frames labeled 0. ϕ controls the sensitivity of the loss function to easily classified frames: frames labeled 1 with sigmoid outputs already above ϕ and frames labeled 0 with sigmoid outputs already below $1 - \phi$ do not contribute to the loss function as they are considered good-enough. This prevents the model from learning to better classify frames that are already well classified at the expense of learning to classify difficult frames. Observe that if we set $\lambda = 1$ and $\phi = 1$, then the loss function returns to the binary cross-entropy function.

2.3. Post-processing for search

Having obtained the vector of probabilities from (5), we post-process them to perform various keyword search tasks.

2.3.1. Segment classification

This task is a simplified form of keyword search where it is enough to classify whether or not a speech segment contains the given query. To do this, we simply compute a segment-wide score: $p = \max_n z_n$, and say that the segment contains the query if p is larger than some threshold, which is tuned on a development set. The rationale for this method is that it is enough for a query to occur at any frame of a segment for it to have occurred in that segment. Therefore, we need only care about the frame with the highest probability.

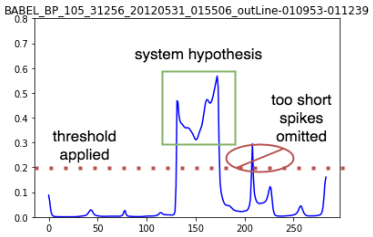

Figure 2: *System hypothesis calculation from network outputs.*

2.3.2. Keyword search

The task here is to return a location within the utterance hypothesized to contain the query along with a confidence score. The procedure, illustrated in Figure 2, is as follows:

1. We set the probabilities (z_n) below some threshold to zero. This threshold is tuned on the development set.
2. We pick the "islands" of non-zero elements as our system hypotheses. The confidence score is computed as the median probability of each interval. We also experimented with the mean and max operations but found median to be better.

The thresholding is a necessary first step because the sigmoid outputs strictly positive values. Additionally, we found that pruning out intervals which are shorter than 20ms $\times$ #letters for each query led to slightly improved performance.

2.3.3. Keyword search rescoring

In this task, we seek to improve the performance of another keyword search system by fusion of scores. Our approach entails rescoring the hypothesis list of a baseline system. Our proposed model is particularly well suited to this task since it outputs scores for each frame, and so we can readily obtain a score for any location with which we are presented. For a given query-document pair, if the baseline hypothesizes a start-end-score triplet $(n_1, n_2, \tilde{p})$, we return a new triplet (n_1, n_2, p) at the same location with score given by a weighted sum of the original score and the average of the network's outputs in that interval, $p = \gamma\tilde{p} + \texttt{mean}(\mathbf{z}[n_1 : n_2])$, where γ is a weighting hyper-parameter which we tune on the development set. We experimented with using the median and max instead of the mean but found no significant improvements.

3. Experiments

3.1. Experiment setup

3.1.1. Data

We experiment on the limited language pack (LLP) data from the IARPA Babel program [21]. For each language, this comprises a 10-hour training set used to train a speaker-adapted HMM-GMM model with which we obtain word level training alignments, a 10-hour development set to tune model hyper-parameters, a 5-hour "evalpart1" set for evaluation, and keyword lists whose distributions are shown in Table 1.

We consider two kinds of acoustic features as input to the document encoder: 80-dimensional filter-banks and 42-dimensional multilingual bottleneck features (BNF), each with frames of length 10ms. The BNF extractor is trained with 19 languages' LLPs (totaling about 190 hours) including Assamese and Bengali but not Pashto, Turkish and Zulu.

Table 1: *Number of dev and eval queries in each language.*

Language	Assamese	Bengali	Pashto	Turkish	Zulu
Dev-IV	1403	1295	1465	219	1194
Dev-OOV	597	705	600	88	806
Eval-IV	5286	4957	3233	1955	2199
Eval-OOV	2087	2368	970	1216	1111

3.1.2. Network configuration

The document encoder consists of 6 BiLSTM layers each with 512 dimensional output in each direction. Each BiLSTM layer (including the first) is preceded by a batch-normalization layer and followed by a dropout layer with dropout probability of 0.4. In addition, the outputs of the first and fourth BiLSTM layers are each down-sampled by a factor of 2 (i.e. $\hat{N} = \lfloor \frac{N}{4} \rfloor$). The final affine projection has an output dimension of $D = 400$.

The query encoder is composed of a 32-dimensional embedding layer, two 256-output-dimensional BiGRU layers each preceded by a batch-normalization layer and a final affine layer with 400-dimensional output.

We train with Adam using an initial learning rate of 2×10^{-4}. We use a batch size of $|\mathcal{Q}| = 64$ and $|\mathcal{X}|_{\mathbf{q}} = 4$ utterances per training phrase. The training hyper-parameters in (6) are empirically chosen as $\lambda = 5$ and $\phi = 0.7$. We select a random 10% of the training utterances for validation. We halve the learning rate whenever the validation loss stagnates for 4 epochs and halt training when it stagnates for 10 epochs.

3.2. Classification task

Our first experiment involves classifying whether a given segment of speech contains a keyword. This task serves as a benchmark for comparing our approach with recent end-to-end keyword search sytems [17, 20]. We use the evaluation setup from [20]. Each utterance is divided into one-second segments with 50% overlap. Any segment that contains even a portion of the keyword is considered a positive test for that keyword. For example, a query which occurs between the 0.4-1.2 second marks of some utterance will yield three positive tests (the ranges 0-1, 0.5-1.5 and 1-2). For each such positive test, a random negative test (3 in total for the given example) is generated from other locations (possibly other utterances) which do not contain the query. Each test returns a score which should be high for positive and low for negative tests. The performance is measured in terms of accuracy and area under the curve (AUC).

We take the systems from [17] and [20] as baselines. In both cases, we directly take the results reported in [20]. For our system, we use the procedure described in Section 2.3.1 with threshold determined on the dev-set.

Table 2 shows the results obtained. Across all languages, our system gives significant improvements in both accuracy and AUC over the baselines. We get still further improvements by switching from filter-banks to BNF. We report P-B results for Assamese and Bengali for completeness, but keep in mind that the BNF extractor training includes those languages.

3.3. Keyword search

In this set of experiments, the task is to return the exact locations of each query within utterances along with the confidence scores. Unlike the balanced classification task, here the problem is made more complex by the presence of more negative trials than positive ones. For instance, the Pashto dev-set contains 14451 examples and 72 million negative trials keeping the rate

Table 2: *Accuracy and AUC on the eval set. B1 and B2 refer to the baseline systems from [17] and [20] respectively. P-F and P-B refer to our system using filter-banks and bottleneck features respectively.*

Language	System	Accuracy IV	Accuracy OOV	AUC IV	AUC OOV
Assamese	B1	0.604	0.604	0.638	0.632
	B2	0.626	0.619	0.724	0.724
	P-F	**0.713**	**0.677**	**0.797**	**0.778**
	P-B	**0.824**	**0.811**	**0.909**	**0.896**
Bengali	B1	0.589	0.579	0.658	0.653
	B2	0.639	0.637	0.743	0.742
	P-F	**0.744**	**0.696**	**0.840**	**0.769**
	P-B	**0.836**	**0.784**	**0.912**	**0.871**
Pashto	B1	0.608	0.594	0.653	0.634
	B2	0.674	0.665	0.797	0.786
	P-F	**0.857**	**0.830**	**0.927**	**0.909**
	P-B	**0.881**	**0.855**	**0.946**	**0.925**

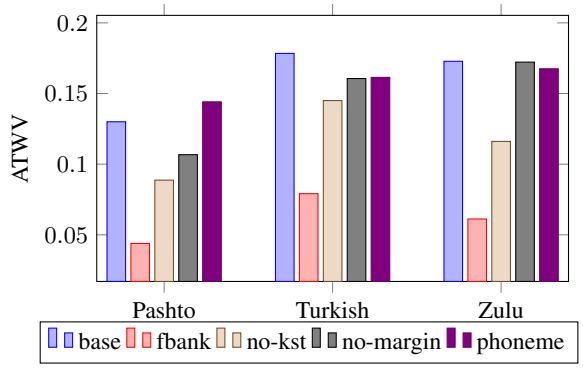

Figure 3: *Eval-set ATWV of various ablations. Every other bar is obtained by changing one component from the* `base` *system.*

of one trial/second, which makes metrics like accuracy unsuitable. Therefore, we use the term weighted value (TWV) metric proposed by NIST [22] for KWS evaluations [21, 23, 24, 25]. The TWV for a set of queries $\mathcal{Q}$ and a threshold θ is computed as follows:

$$TWV(\theta, \mathcal{Q}) = 1 - \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} (P_{miss}(q,\theta) + \beta P_{FA}(q,\theta)), \quad (7)$$

where $P_{miss}(q,\theta)$ and $P_{FA}(q,\theta))$ are the probabilities of misses and false alarms respectively, and β controls the relative costs of false alarms and misses. As in the NIST STD evaluations [23], we set $\beta = 999.9$. On the dev-set, we report the maximum TWV (MTWV) which is the TWV at the threshold that maximizes it. This threshold is fixed and used to compute the actual TWV (ATWV) for the eval-set. Since TWV requires a single threshold, it is necessary to normalize the scores across keywords. We adopt keyword specific thresholding (KST) from [26] for score normalization.

As stated before, Assamese and Bengali are included in the BNF extractor training, so we elect not to report results for those. Instead we use Turkish and Zulu, which, along with Pashto, are not included in the BNF training.

Figure 3 shows the performance of our system when used as a standalone KWS system by following the procedure described in Section 2.3.2 with KST normalization (`base`). For context, our re-implementation of [17] (using BNF) achieved MTWV of 0 and negative ATWV. The figure also shows the impact of changing various components of our system. First, we find that, as expected in this low-resource setting, filter-banks (`fbank`) perform worse than BNF. We also observe deterioration in performance when we omit score normalization (`no-kst`).

Next, we measured the impact of removing the margin term from our loss function by setting $\phi = 1$ in (6) which results in a weighted binary cross-entropy objective (`no-margin`). For Turkish and Pashto, we find that the margin term helps significantly, whereas for Zulu, the impact is milder.

Furthermore, we considered the impact of using phonemic instead of graphemic query representation (`phoneme`). When using phonemes, we use the lexicon available with the Babel data for training and for IV terms, and we train a G2P model [27] to obtain pronunciations for OOV terms. We find graphemes to be superior for Turkish and Zulu. This can be explained by the fact that when using phonemes, we only consider one pronunciation for each word, so the model learns a limited pronunciation model; whereas when we use graphemes, the model is allowed to implicitly learn pronunciation variability. However, for Pashto, phonemes work better. We hypothesize that this is because the graphemic representation uses the non-diacritized Pashto alphabet which omits vowels.

Finally, we utilize our system (referred to as `base` in Figure 3) to rescore the output of an LVCSR-based KWS system (multilingually pre-trained and finetuned on each target language, with words for IV and subwords for OOV search). We rescore using the procedure described in Section 2.3.3 followed by KST normalization. The results in Table 3 show that we are able to significantly improve on an already competitive system.

Table 3: *Results of rescoring LVCSR output. L denotes the LVCSR baseline. L $\oplus$ P denotes the result of rescoring.*

Language	System	Dev MTWV	Eval ATWV
Pashto	L	0.2754	0.3190
	L $\oplus$ P	**0.3068**	**0.3464**
Turkish	L	0.4892	0.4051
	L $\oplus$ P	**0.5080**	**0.4419**
Zulu	L	0.3716	0.3381
	L $\oplus$ P	**0.3917**	**0.3620**

4. Conclusion

In this work, we have proposed a model that is directly optimized for keyword search. By predicting frame-wise likelihoods of keyword occurrences, the proposed model facilitates fine-grained keyword search and readily accommodates fusion with other systems.

We showed that our method works well not only for utterance classification used as a proxy for search but also for the more difficult "needle-in-a-haystack" search problem - an encouraging result for the prospects of end-to-end approaches to keyword search. Moreover, we have shown that an application of our technique can already significantly improve LVCSR-based keyword search performance through rescoring.

5. Acknowledgements

The work was supported by European Union's Horizon 2020 project No. 870930 - WELCOME, Boğaziçi University Research Fund under the Grant Number 16903, and the Turkish Directorate of Strategy and Budget under the TAM Project number 2007K12-873.

6. References

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

[2] D. Wang, J. Frankel, J. Tejedor, and S. King, "A comparison of phone and grapheme-based spoken term detection," in *ICASSP*, 2008, pp. 4969–4972.

[3] S. Parlak and M. Saraçlar, "Performance analysis and improvement of Turkish broadcast news retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 731–741, 2012.

[4] A. Garcia and H. Gish, "Keyword spotting of arbitrary words using minimal speech resources," in *ICASSP*, vol. 1, 2006, pp. 949–952.

[5] S.-w. Lee, K. Tanaka, and Y. Itoh, "Combining multiple subword representations for open-vocabulary spoken document retrieval," in *ICASSP*, vol. 1, 2005, pp. 505–508.

[6] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 615–622.

[7] D. Karakos and R. Schwartz, "Subword and phonetic search for detecting out-of-vocabulary keywords," in *Interspeech*, 2014, pp. 2469–2473.

[8] I. Szöke, M. Fapšo, and L. Burget, "Hybrid word-subword decoding for spoken term detection," in *The 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2008, pp. 42–48.

[9] M. Saraçlar, A. Sethy, B. Ramabhadran, L. Mangu, J. Cui, X. Cui, B. Kingsbury, and J. Mamou, "An empirical study of confusion modeling in keyword search for low resource languages," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 464–469.

[10] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for oov keywords in the keyword search task," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 416–421.

[11] B. Gündoğdu, B. Yusuf, and M. Saraçlar, "Joint learning of distance metric and query model for posteriorgram-based keyword search," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1318–1328, 2017.

[12] B. Yusuf, B. Gundogdu, and M. Saraclar, "Low resource keyword search with synthesized crosslingual exemplars," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 7, pp. 1126–1135, 2019.

[13] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *ICASSP*. IEEE, 2014, pp. 4087–4091.

[14] Y. Segal, T. S. Fuchs, and J. Keshet, "SpeechYOLO: Detection and Localization of Speech Objects," in *Proc. Interspeech 2019*, 2019, pp. 4210–4214. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1749

[15] D. Palaz, G. Synnaeve, and R. Collobert, "Jointly learning to locate and classify words using convolutional networks," *Interspeech*, pp. 2741–2745, 2016.

[16] R. Alvarez and H.-J. Park, "End-to-end streaming keyword spotting," in *ICASSP*. IEEE, 2019, pp. 6336–6340.

[17] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017.

[18] Y.-C. Chen, S.-F. Huang, H.-y. Lee, Y.-H. Wang, and C.-H. Shen, "Audio word2vec: Sequence-to-sequence autoencoding for unsupervised learning of audio segmentation and representation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 9, pp. 1481–1493, 2019.

[19] H. Kamper, Y. Matusevych, and S. Goldwater, "Improved acoustic word embeddings for zero-resource languages using multilingual transfer," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1107–1118, 2021.

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

[21] M. Harper, "IARPA Babel program," https://www.iarpa.gov/index.php/research-programs/babel, 2014, accessed at March 2021.

[22] J. Fiscus, J. Ajot, and G. Doddington, "The spoken term detection (std) 2006 evaluation plan," *NIST USA*, 2006.

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

[24] "OpenKWS14 keyword search evaluation plan," http://www.nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf, accessed at March 2021.

[25] F. Byers and O. Sadjadi, *2017 Pilot Open Speech Analytic Technologies Evaluation (2017 NIST Pilot OpenSAT): Post Evaluation Summary*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[26] D. R. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Interspeech*, 2007, pp. 314–317.

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