INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

The Kaldi OpenKWS System: Improving Low Resource Keyword Search

Jan Trmal[1,2], *Matthew Wiesner*[1], *Vijayaditya Peddinti*[1,2†], *Xiaohui Zhang*[1], *Pegah Ghahremani*[1], *Yiming Wang*[1], *Vimal Manohar*[1,2], *Hainan Xu*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing, Johns Hopkins University, USA
[2]Human Language Technology Center of Excellence, Johns Hopkins University, USA
{trmal,wiesner,vijay.p,xiaohui,pghahre1,ywang169,vmanoha1,hxu31,dpovey1,khudanpur}@jhu.edu

Abstract

The IARPA BABEL program has stimulated worldwide research in keyword search technology for low resource languages, and the NIST OpenKWS evaluations are the de facto benchmark test for such capabilities. The 2016 OpenKWS evaluation featured Georgian speech, and had 10 participants from across the world. This paper describes the Kaldi system developed to assist IARPA in creating a competitive baseline against which participants were evaluated, and to provide a truly open source system to all participants to support their research. This system handily met the BABEL program goals of 0.60 ATWV and 50% WER, achieving 0.70 ATWV and 38% WER with a single ASR system, i.e. *without ASR system combination*. All except one OpenKWS participant used Kaldi components in their submissions, typically in conjunction with system combination. This paper therefore complements all other OpenKWS-based papers.

Index Terms: speech recognition,keyword search, spoken term detection, IARPA Babel, OpenKWS

1. Introduction

This paper discusses a keyword search (KWS) system developed for the IARPA Babel program. The technical goal of the last phase of the program, named Option Period 3 or OP3, was to develop an automatic speech recognition (ASR) and KWS system in a new language given only 40 hours of transcribed speech, no pronunciation lexicon, and some supplemental web-text, that would deliver an ASR word error rate (WER) under 50%, and KWS actual term weighted value (ATWV) over 0.60. (See [1, 2] for a discussion of the ATWV metric.) The program did not constraint the computational resources nor the latency in processing the speech; the only constraint was to develop the ASR and KWS systems in one week after receiving the data, and to submit them to NIST's OpenKWS 2016 benchmark test.

Though typical competitive submissions in OpenKWS tests combine several individual ASR and KWS systems, the BABEL program tasked us with developing a *practical*, yet state-of-the-art, baseline system for use by IARPA to analyze test results. Therefore, the systems we developed respect the OP3 resource constraints, but do not perform ASR system combination. As there are no latency constrains we use bidirectional acoustic models in our ASR system.

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) contract No 2012-12050800010, DARPA LORELEI contract No HR0011-15-2-0024, and NSF Grant No CRI-1513128. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of IARPA, DARPA, DoD/ARL, or the U.S. Government.

[0†]Vijayaditya Peddinti is now with Google, Inc.

2. Novel Features of the Kaldi ASR System

Given the OP3 data and task constraints—having only 40 hours of transcribed telephone speech, no phonetic lexicon, some raw text scraped from web—and the expectation that the KWS system would still handle out of vocabulary (OOV) keywords, we introduced the following novel features to Kaldi.

They are now available through GitHub for developing ASR and KWS systems in other low resource languages

2.1. Grapheme-Based Pronunciation Lexicons

In the absence of a (phonemic) pronunciation lexicon, one must naturally turn to the orthography of the language to create a sub-word representations of words for acoustic modeling. Now, the documentation provided as a part of the OP3 data does contain a mapping from graphemes to phonemes (G2P), but creating a pronunciation lexicon from it is often tedious and complex, especially for inexperienced users unfamiliar the languages and for languages with complex writing systems (G2P mappings).

We therefore investigated automatically inducing sub-word representations directly from the text, without relying on language-specific knowledge. We used the Unicode database [4] to convert the graphemic rendering of a word into a sub-word representation for acoustic modeling: specifically, we use as "phonemes" the standard character names in Normalization Form D (NDF, Canonical Decomposition form). In this form, the characters (i.e. graphemes) are normalized to their canonical form, and multiple combining characters are arranged in a specific order. Our method thus extends the approach of [3] by providing a means to handle certain non-segmental writing systems. It also differs in the mapping derived from the Unicode descriptions as noted below in Section 2.1.1.

Now, writing systems used today may be grouped into four major families: (i) alphabets, (ii) abjads, (iii) syllabaries and abugidas, and (iv) purely logographic writing systems. We handle them as follows.

Alphabets are writing systems in which the graphemes generally represent phonemes, including vowels and consonants. Examples of alphabets present in the Babel corpora are the Latin, Cyrillic, and Georgian (Mkhedruli) scripts. Character names retrieved from Unicode have names such as "GEORGIAN CAPITAL LETTER MAN", and "LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND HOOK ABOVE". After NFD normalization, the latter character, for example, will be mapped into a sequence of three units: "LATIN CAPITAL LETTER O", "COMBINING CIRCUMFLEX ACCENT" and "COMBINING HOOK ABOVE".

This standard form allows us to parse composite

http://dx.doi.org/10.21437/Interspeech.2017-601

graphemes, and generate their estimated "phonetic" representations. It is important to realize that accents (combining characters) have language-specific functions. For example, accents can indicate stress (Catalan, Italian), length (Czech), or tone (Vietnamese) of a phoneme; they may indicate a modified pronunciation; and in some cases they do not affect pronunciation.

Abjads also exhibit related graphemic and phonemic representations, but differ from alphabets in that graphemes in abjads only encode consonants and lengthened vowels. All other vowels are rarely (or not) written. As a result, extracting the correct pronunciation of a word or a sequence of words requires additional semantic and syntactic knowledge.

Currently, the only commonly used abjads are variations of the Arabic and Hebrew scripts. The only abjad present in the Babel corpus is Pashto. An example of such a grapheme is "ARABIC LETTER ALEF + ARABIC MADDAH ABOVE". Composite characters are again decomposed into a standard (base) character, followed by combining characters.

As was the case for alphabets, the combining characters in abjads encode phonetic phenomena including emphasis, lengthening, glottal stops (Arabic), lenition (Hebrew) and pronounced case endings (Arabic). Most importantly, in almost all languages written with abjads (Arabic, Hebrew, Farsi, Pashto), the omitted vowels are occasionally written as diacritic markings, especially in pedagogical and religious contexts.

As in alphabets, we must determine the phonetic importance of diacritics, but the lack of vowels imposes additional challenges; the acoustic model we build for each normalized character will, in reality, need to represent the consonant from which it was derived as well as any surrounding vowel(s).

Syllabaries are writing systems in which each character represents a syllable. Examples of such are Cherokee and Katakana. **Abugidas** are like syllabaries, except that the grapheme representing a syllable can usually be decomposed into a consonant and a vowel part. A single Tamil glyph for example, is represented as a sequence of two code-points: "TAMIL LETTER KA + TAMIL VOWEL SIGN II."

Note that whether a certain script with these traits is represented as a syllabary or an abugida in Unicode is not based exclusively on its linguistic category—the historical organization of the character set, and overall compatibility issues are taken into account during the standardization process.

Amharic, for instance, is an abugida that is effectively treated as a syllabary. i.e. "ETHIOPIC SYLLABLE SO" is an atomic character, and NFD will not decompose it into its linguisticlly *obvious* components "ETHIOPIC LETTER S" and "ETHIOPIC VOWEL SIGN O." We, however, parse the Unicode *description* of the Unicode character and still generate the correct number of acoustic units.

For **purely logographic writing systems** such as the Han script, hieroglyphs or Mayan glyphs, Unicode character descriptions do not provide enough phonetic information. In such cases, the approach we developed *does not apply*, and alternative solutions must be devised.

2.1.1. Unicode-driven Induction of Pronunciation Lexicons

Observe from the discussion above that for each of the 4 types of writing systems, parsing the Unicode descriptions of the NFD decomposition of characters transforms the problem of finding the phoneme-like pronunciation of a word into the problem of deciding which combining characters are phonetically relevant.

We ignore for now the issue of orthographic digraphs. Firstly, the Babel languages do not exhibit this property. Furthermore, prior work [5] shows that such cases can be partially accounted for by using context-sensitive (triphones) models.

Table 1: *Comparison of ASR and KWS performance (on dev speech, using dev keywords) between CE-BLSTM systems built using grapheme-based versus phoneme-based lexicons.*

Babel OP3	WER		ATWV	
Language	Phonetic	Graphemic	Phonetic	Graphemic
Pashto	46.2 %	47.2 %	0.372	0.338
Guarani	44.9 %	45.8 %	0.487	0.464
Igbo	57.1 %	55.8 %	0.270	0.284
Amharic	43.0 %	43.5 %	0.486	0.475
Mongolian	48.2 %	48.7 %	0.419	0.404
Javanese	53.8 %	53.5 %	0.350	0.325
Dholuo	39.4 %	38.6 %	0.501	0.507

For each combination of base grapheme + combining-characters seen sufficiently often in the acoustic training data, create a separate acoustic-phonetic unit. For all infrequent combinations of grapheme + combining-characters, use the base grapheme as an acoustic unit, and the combining characters as tags. When building phonetic decision-trees for triphone clustering, permit questions about each tag, so as to decide in a data-driven manner whether base characters with different tags merit separate acoustic models.

The intuition behind this approach is that redundant acoustic units are acceptable provided we have sufficient data with which to train their models. Otherwise, we let the phonetic decision-tree building algorithm determine the phonetic relevance of the combining character. Since the definition of "infrequent" is somewhat arbitrary in the HMM/GMM modeling framework, we define the threshold for rare graphemes relative to other graphemes in the language. For our experiments we use the data-driven approach on the rarest 10% of the graphemes in the acoustic training data.

Table 1 compares ASR/KWS systems for the OP3 languages: those that use graphemic lexicons built as described here versus hand-crafted phonetic lexicons. Despite the simplicity of our strategy, which does not address digraphs, diphtongs, or missing vowels in abjads, performance suffers only modestly.

2.2. Deep Neural Network-based ASR Systems

2.2.1. Cross-Entropy trained BLSTM systems

We train a hybrid hidden Markov model, deep neural network (HMM-DNN) based ASR systen in the Kaldi Nnet3 framework ([7, 8]). The DNN is composed of 3 layers of bi-directional long-short-term memory (BLSTM) layers (see [9]), and is trained using the cross-entropy (CE) objective function.

The error gradients w.r.t. to the parameters of the system are computed using context-sensitive chunk back-propagation through time (chunk BPTT) similar to the approach suggested in [10]. Randomly shuffled 20-frame chunks, with context of 40 frames on both the left and right sides, were used to train the network. BPTT was performed for just 20 time steps. To stabilize the BLSTM training, norm-based gradient clipping was used. Furthermore, the error-gradients w.r.t. sigmoid and tanh activations in the BLSTMs were clipped during BPTT if they crossed a threshold.

We used the speed perturbation data augmentation approach of [11] with coefficients 0.9 and 1.1, effectively tripling the amount of the audio data to 120 hours. High-resolution 40 dimensional MFCCs, plus 3 dimensional pitch features, spliced

Table 2: *Performance (WER, ATWV) comparison of different types of acoustic models. Graphemic systems for Babel OP3 languages. Notice the opposite trends for WER and ATWV.*

Language	LF-MMI TDNN		CE BLSTM	
	WER	ATWV	WER	ATWV
Pashto	48.2 %	0.377	47.2 %	0.338
Guarani	47.2 %	0.497	45.8 %	0.464
Igbo	59.4 %	0.288	55.8 %	0.284
Amharic	44.8 %	0.506	43.5 %	0.475
Mongolian	50.7 %	0.448	48.7 %	0.404
Javanese	55.3 %	0.384	53.5 %	0.325
Dholuo	40.9 %	0.545	38.6 %	0.507

over a context of 5 frames ($[t-2, t+2]$) were used as the input to the neural network for every time step.

The i-vectors, when provided as an input to a neural network, have been shown to perform speaker adaptation [12]. Furthermore, they have also been shown to be useful for environment adaptation. Hence, in the proposed system, we use i-vectors as an additional input to the network at the initial layer for speaker and environment adaptation.

In addition to these, for the final Georgian system, *multilingual bottle-neck features* (ML-BNFs) were also appended to each frame. The ML-BNF extractor is described in Section 5.

Indicative performance of this system is shown in Table 2 (cf CE-BLSTM) for the OP3 development languages.

2.2.2. Chain LF-MMI systems

Purely sequence trained TDNNs using lattice-free maximum mutual information (LF-MMI) criterion [13], which usually outperform CE-trained systems, did not do so out of the box. We weren't able to make the training stable enough before the OpenKSW submission deadline, so our submission system was the CE-BLSTM system described above. Subsequently, clean up of the training data was found to be necessary, as was the case in other small-data tasks, as noted in [13]. The data clean up was done using constrained decoding of the training utterances, as described in [14]. This removed around 10% (i.e. 4 hours) of the training data. Indicative performance of this system is shown in Table 2 (cf LF-MMI) for the OP3 languages.

Note from Table 2 that though the WERs of LF-MMI systems are worse than the CE-BLSTM systems, they consistently yielded better ATWVs. The likely reason is that the LF-MMI lattices have greater depth—despite the fact the pruning beams were the same for both systems. Due to stability issues we initially encountered in training the LF-MMI system, we submitted the CE-BLSTM system for OpenKWS. Subsequently, however, it has become clear that the LF-MMI TDNN system fares better: it has several additional advantages over the CE-BLSTM systems including greater training & decoding speed, comparable WER, better ATWV, etc.

3. The Kaldi Keyword Search System

The underlying KWS capability in Kaldi is word-based: word-level ASR lattices are converted into an inverted index via a factor transducer, as proposed in [15]. The KWS pipeline is conceptually based on the Babel program Base-Period (BP) and Babel Program Option Period 1 (OP1) pipeline described in [16], but was completely redesigned for OP3, with the aims of (i) enhancing OOV search, and (ii) improving score normalization.

3.1. Score Normalization: Keyword Specific Thresholding

For score normalization, we use the keyword specific threshold technique (KST) from [17]; see also [18] for a detailed discussion of other normalization techniques. The N_{true}-scale parameter needed for KST is determined experimentally by sweeping the range 1.0 to 2.0, and choosing the one that maximizes ATWV on the dev speech+keywords. The value thus chosen is used on the evaluation speech+keywords.

3.2. Fusing Multiple Methods for OOV Search

The new KWS pipeline uses multiple methods to search the ASR output for OOV keywords, as described below. Each putative keyword hit (token) is assigned a score that is a weighted combination of its scores in the hit-lists of the individual methods. The combination weights are initialized to reflect the maximum term weighted value of the individual hit-list, and then optimized on the dev speech+keywords using Powell's method, as done in [19], to maximize ATWV.

This KWS pipeline is directly applicable for KWS system combination, if one were to build multiple ASR+KWS systems.

4. OOV Keyword Search Strategies

The new Kaldi KWS pipeline combines three approaches for handling OOV keywords.

- **Proxy search:** An OOV keyword is converted into sequences of similar sounding in-vocabulary words, and the word-index is searched. See [20] for details.
- **Phonetic search:** An OOV keyword is converted into phonetic sequences, and a phone-level index, derived from an ASR phone lattice, is searched.
- **Syllable/morpheme search:** An OOV keyword is split into a sequence of syllables or morphemes, and a syllable- or morpheme-level index (respectively) is searched. Syllabic decomposition relies on a hand-crafted lexicon; for obtaining a decomposition of words into morphemes, we used Morphessor 2.0 [21]. In either case, we use the proxy search strategy described above to find putative matches of these subword sequences in the appropriate ASR subword lattices.

There are two ways to obtain subword-level lattices: one is to create the decoding graph in terms of these subword units, and run the decoding directly. The other is to convert the word-level ASR lattices into subword-level lattices using a finite state word-to-subword transducer (cf [22]). Despite the fact that direct decoding generally yields better KWS performance, we used the transduction method, which results in a straightforward pipeline that minimizes multiple ASR decoding passes.

Note that though the original (word-level) lattices contain position-dependent (i.e. word-internal versus word-boundary) tags on phonemes, these boundary markers lose their meaning after conversion to morpheme-level or phoneme-level lattices. They are treated like general tags, and do not circumscribe the keyword match.

5. Multilingual Acoustic Model Training

5.1. Data Selection for Multilingual Training

Keeping the "practicality" objective of the system in mind, we didn't want to pursue the approach of training the multilingual

Table 3: *Improvements in WER from different features.*

AM	Features	WER
BLSTM	MFCC+iVector	45.9 %
BLSTM	MFCC+iVector+pitch	45.0 %
BLSTM	MFCC+iVector+pitch+ML-BNF	44.1 %

extractor from as many languages and as much data as possible, as is sometimes practiced (e.g. [23])[1].

Instead we used a technique similar to those used in [24, 25] that entails selecting languages close to the OpenKWS language, Georgian. Specifically, we used the confusion matrix generated from a language identification system to select the 10 languages most confusable with Georgian. The language identification system is a neural network trained to classify 2-10 second utterances as one of the 25 Babel languages. It was based using a time-delay neural network (described in [6]) followed by a statistical pooling layer. The closest languages were selected by computing the average language class posterior over all frames from a given language.

5.2. Bottleneck-features extractor

We added the FullLP data from 10 additional languages identified as being closest to Georgian—Lithuanian, Mongolian, Turkish, Kazakh, Kurmanji, Pashto, Swahili, Tok Pisin, Igbo and Dholuo—to train a multi-lingual acoustic model. The model is trained within a HMM-TDNN hybrid system where the TDNN shares all layers, other than the final affine layer, among different languages. The final softmax (output) layer is divided into language-specific parts that classify senones from their respective language. The TDNN has 6 layers where the fifth layer has a 42 dimensional bottle-neck.

In each epoch of training, the TDNN is trained using mini-batches of data sampled from all 11 languages. The sampling of the mini-batches is based on the relative frequency of data from these languages

The high-resolution MFCC (dimension of 40), pitch features (dimension of 3, see [26]) and 100-dimensional iVectors are used as input features for this multi-lingual acoustic model. The iVector extractor used for extracting the iVectors was trained using data from all 11 languages.

The outputs of the bottle-neck layer are termed as multi-lingual bottle-neck features (ML-BNFs). The improvements in WER from using these features are detailed in the Table 3.

5.3. Using external text

The web text provided as part of the language resources was used to enhance the language models. The web text was cleaned and used to generate a Georgian word list to the augment the vocabulary from 30000 words to 530000 words. Using this augmented vocabulary a 4-gram LM was trained on the in-domain training data (i.e. training data transcripts). A second 4-gram LM was trained on the web text. These two LMs were then interpolated. The interpolation weight was chosen to minimize the perplexity on the development set.

During the evaluation run, the data was first decoded using the trigram LM with the full 530k vocabulary and the 4-gram interpolated LM was used to rescore the lattices.

[1]One could also argue that a multilingual system trained on many languages would be useful for many other languages and tasks, which would mean that the "ammortized training time" is negligible, and that argument certainly can be valid. For our setup, however, we felt that the BNF extractor training time should be part of the total system statistics.

Table 4: *Influence of adding new words from the web-scraped data and rescoring using an interpolated LM (created from training text and cleaned web-scraped text)*

Decoding LM	Rescoring LM	WER
3-gram (30K vocab)	None	44.1%
3-gram + 500K vocab	None	42.9%
3-gram + 500K vocab	Interpolated 4-gram	42.2%

Table 5: *Final results on the dev set and official results on eval set. ATWV(kwlist) denotes development keyword list, ATWV(kwlist3) denotes the official evaluation keyword list.*

	WER	ATWV(kwlist)	ATWV(kwlist3)
dev set	42.2%	0.619	0.617
eval set	37.5%	–	0.705

5.4. Official OpenKWS Evaluation Results

The official NIST results are reported in the Table 5. They easily surpass the Babel program goals: 50% WER and 0.60 ATWV.

While comparing these results with those of other OpenKWS participants, please keep in mind that these were obtained using a single-pass ASR system, and the search results were obtained by combining three different hit-lists (word+proxy, phonetic, morphemic), all created by finite state transduction from a single set of word-lattices.

6. Conclusions

We presented a simple *yet high performing* KWS system based on single-pass decoding with a BLSTM. Several design decisions are discussed, including their quantitative impact on overall performance of the ASR+KWS pipeline. This pipeline does not need any proprietary systems and is freely available. We hope that public release of the pipeline, along with this paper, will stimulate further research, especially in KWS for low-resource languages. The consensus from the Babel program is that some languages are much harder than others and significant further research is needed to understand the causes.

7. Acknowledgement of Babel Datasets

The following IARPA Babel language packs were used to train and evaluate ASR and KWS systems: *Georgian* release IARPA-babel404b-v1.0a, *Dholuo* release IARPA-babel403b-v1.0b, *Javanese* release IARPA-babel402b-v1.0b, *Mongolian* release IARPA-babel401b-v2.0b, *Amharic* release IARPA-babel307b-v1.0b, *Igbo* release IARPA-babel306b-v2.0c, *Guarani* release IARPA-babel305b-v1.0 and *Pashto* release IARPA-babel104b-v04.bY.

In addition, the following language packs were used for multilingual training: *Lithuanian* release IARPA-babel304b-v1.0b, *Turkish* release IARPA-babel105b-v0.4, *Kazakh* release IARPA-babel302b-v1.0a, *Kurmanji* release IARPA-babel205b-v1.0a, *Swahili* release IARPA-babel202b-v1.0d and *Tok Pisin* release IARPA-babel207b-v1.0e.

8. References

[1] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Proceedings of the ACM SIGIR Conference*, vol. 7, 2007, pp. 51–57.

[2] S. Wegmann, A. Faria, A. Janin, K. Riedhammer, and N. Morgan, "The tao of atwv: Probing the mysteries of keyword search performance," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 192–197.

[3] M. J. F. Gales, K. M. Knill, and A. Ragni, "Unicode-based graphemic systems for limited resource languages," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 5186–5190.

[4] The Unicode Consortium, *The Unicode Standard, Version 9.0.0*. Mountain View, CA: The Unicode Consortium, 2016. [Online]. Available: http://www.unicode.org/versions/Unicode9.0.0/

[5] M. Killer, S. Stüker, and T. Schultz, "Grapheme based speech recognition." in *Proceedings of Interspeech*, 2003.

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

[7] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[8] D. Povey. Nnet3: Neural network toolkit for generic acyclic computation graphs. [Online]. Available: http://kaldi-asr.org/doc/dnn3.html

[9] A. Graves, S. Fernández, and J. Schmidhuber, *Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 799–804. [Online]. Available: http://dx.doi.org/10.1007/11550907_126

[10] K. Chen and Q. Huo, "Training Deep Bidirectional LSTM Acoustic Model for LVCSR by a Context-Sensitive-Chunk BPTT Approach," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 7, pp. 1185–1193, July 2016.

[11] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *INTERSPEECH*, 2015.

[12] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 55–59.

[13] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, Y. Wang, X. Na, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Proceedings of Interspeech*, 2016.

[14] V. Peddinti, V. Manohar, Y. Wang, D. Povey, and S. Khudanpur, "Far-field asr without parallel data," in *Proceedings of Interspeech 2016*, 2016.

[15] D. Can and M. Saraclar, "Lattice Indexing for Spoken Term Detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, Nov 2011.

[16] J. Trmal, G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow, D. Yarowsky, and F. Metze, "A keyword search system using open source software," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2014, pp. 530–535.

[17] 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 *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[18] Y. Wang and F. Metze, "An in-depth comparison of keyword specific thresholding and sum-to-one score normalization," in *Proceedings of Interspeech 2014*, 2014.

[19] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel, and V. B. Le, "Score normalization and system combination for improved keyword spotting," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 210–215.

[20] 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*, Dec 2013, pp. 416–421.

[21] S. Virpioja, P. Smit, S.-A. Grönroos, M. Kurimo *et al.*, "Morfessor 2.0: Python implementation and extensions for Morfessor Baseline," Aalto University, Technical Report, 2013.

[22] H. Su, J. Hieronymus, Y. He, E. Fosler-Lussier, and S. Wegmann, "Syllable based keyword search: Transducing syllable lattices to word lattices," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2014, pp. 489–494.

[23] T. Alumäe, D. Karakos, W. Hartmann, R. Hsiao, L. Z. L. Nguyen, S. Tsakalidis, and R. Schwartz, "The 2016 bbn georgian telephone speech keyword spotting system," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017.

[24] A. Ragni, M. J. F. Gales, and K. M. Knill, "A language space representation for speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 4634–4638.

[25] J. Cui, B. Kingsbury, B. Ramabhadran, A. Sethy, K. Audhkhasi, X. Cui, E. Kislal, L. Mangu, M. Nussbaum-Thom, M. Picheny, Z. Tske, P. Golik, R. Schlter, H. Ney, M. J. F. Gales, K. M. Knill, A. Ragni, H. Wang, and P. Woodland, "Multilingual representations for low resource speech recognition and keyword search," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec 2015, pp. 259–266.

[26] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2494–2498.