INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Comparison of Multiple System Combination Techniques for Keyword Spotting

William Hartmann, Le Zhang, Kerri Barnes, Roger Hsiao, Stavros Tsakalidis, Richard Schwartz

Raytheon BBN Technologies, Cambridge, MA, USA
{whartman, lzhang, kbarnes, whsiao, stavros, schwartz}@bbn.com

Abstract

System combination is a common approach to improving results for both speech transcription and keyword spotting—especially in the context of low-resourced languages where building multiple complementary models requires less computational effort. Using state-of-the-art CNN and DNN acoustic models, we analyze the performance, cost, and trade-offs of four system combination approaches: feature combination, joint decoding, hitlist combination, and a novel lattice combination method. Previous work has focused solely on accuracy comparisons. We show that joint decoding, lattice combination, and hitlist combination perform comparably, significantly better than feature combination. However, for practical systems, earlier combination reduces computational cost and storage requirements. Results are reported on four languages from the IARPA Babel dataset.
Index Terms: speech recognition, keyword spotting, system combination, joint decoding, lattice combination

1. Introduction

Keyword spotting (KWS) in low resourced languages has been a recent topic of significant interest [1, 2, 3, 4]. Typical systems must struggle with limited data and high word error rates (WER). It is becoming increasingly common to increase performance through the combination of multiple systems [5, 6]. With the advent of deep learning, a large variety of possible models can be trained and used. While each individual model must be trained—increasing the overall cost of the final system—it is a one time cost that can be reduced through parallel training if the resources are available. Depending on the goal, the increased performance may be worth the additional training cost. Obviously the additional training time is not the only cost when using multiple models; additional models also increase decoding and keyword search time.

Little consideration has been paid to the trade-off between system performance and computational requirements in previous work. Most work focuses only on the final performance metrics when comparing combination techniques [7]. Joint decoding, similar to multi-stream ASR [8], combines multiple systems during decoding and lattice generation [9]. This comes with an increase in decoding time, but it should not affect any subsequent processing. Hitlist combination is another common approach [2, 6, 10]. It combines results only after both decoding and keyword search have been performed by each system separately. A comparison of just system performance ignores the significant differences in computational effort and storage required—both are important in system deployment.

We also investigate two additional combination strategies. Feature combination takes different types of features and concatenates them together. This is commonly done with different types of cepstral, pitch, and bottleneck features [11, 12]. After decoding, lattices can also be combined to generate a single lattice. To our knowledge, this is the first time lattice combination has been investigated in the context of KWS. Earlier work jointly searched multiple lattices in order to minimize bayes risk for ASR, but the individual lattices were never combined [13]. Lattice combination still requires multiple decodings, but eliminates the need for additional storage.

Previous studies have focused only on combining two systems, either a tandem and hybrid system, or two GMM systems [7, 13]. We expand upon this by looking at three state-of-the-art acoustic models—a CNN and two DNNs trained on different sets of bottleneck features. Both two system and three system combinations are considered.

2. Combination Approaches

We explore several combination approaches, ranging from early fusion to late fusion. In addition to the differences in performance, each approach varies in flexibility and computational cost. It is important to acknowledge the trade-offs each approach entails. If systems are not designed with system combination in mind, they will be difficult to combine using certain approaches. Typically, the earlier the fusion, the more constraints are placed on the systems.

2.1. Cross Adaptation

Since our DNN acoustic models use speaker adapted bottleneck features, a first decoding pass is required to compute the fMLLR transform. This is typically done with a GMM model. As we are already exploring system combination with CNN and DNN acoustic models, there is no additional cost to performing decoding with the CNN. The only requirement is the targets for the CNN must be the same as the GMM states used during the fMLLR transform estimation. Since the CNN is more accurate than the GMM, it may improve the accuracy of the fMLLR estimation and, ultimately, the final WER.

2.2. Feature Combination

We have trained two DNNs on different types of features. The first are bottleneck features generated from a DNN, and the second are bottleneck features trained from a CNN. These features

This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: 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 expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

http://dx.doi.org/10.21437/Interspeech.2016-1381

can be concatenated for an early fusion of the systems. The two types of features still need to be generated separately. The models will be slightly larger since the initial feature size is doubled. If GMMs used for fMLLR computation share the same set of acoustic states, the decoding time should not be affected much as a single first-pass decoding can be used. Prior to combining the bottleneck features, two sets of fMLLR transforms are computed with two GMM acoustic models. Only after SAT has been applied are the features combined. It would also be possible to combine the features before GMM training so that only one SAT model is required, but we chose to use the models already built instead. Initial results show a decrease in performance due to the larger feature vector used during GMM training.

2.3. Joint Decoding

Multiple models can be used in parallel for joint decoding. The predictions from each model are averaged. Each system could potentially be weighted based on some characteristic, but we use an equal weight for each system in this work. Since a forward pass of multiple networks is required, the decoding time increases. Only one graph and lattice are maintained during decoding, so the increase in decoding time is less than decoding each system separately. The main restriction is each model must have the same targets. Specifically, we use a frame-synchronous approach. At each frame, the log posteriors from each system are averaged together. While it would also be logical to take the average of the posteriors prior to applying the log, we others [13]—have obtained better performance by averaging the log posteriors. If the targets were different, it could still be possible to perform joint decoding if maps between the two clusterings are provided [14], but this is beyond the scope of this work.

2.4. Lattice Combination

Previous work has used multiple lattices to generate the one-best hypothesis [13], but we are unaware of previous work combining multiple lattices for KWS. Prior to combining the lattices, the posteriors in each lattice are scaled by their respective combination weight. Omitting this step can cause issues with downstream processing. Attempting to interleave the lattices would be a difficult process. Instead we attach the start and end nodes of each lattice together. Once they have been attached, we convert the lattice to a consensus network (Cnet) [15]. This process implicitly handles the interleaving of the arcs. An alternative is to generate Cnets for each system, and then combine those Cnets. Preliminary experiments gave similar results, but required more computational effort.

Lattice combination is an expensive operation that takes nearly as long as decoding. A significant portion of this computation comes from reading and writing the files to disk; the required computation could be reduced through optimization. It still saves some computation as keyword search does not need to be repeated, and requires no additional storage. The main requirement is that all systems need to use the same pronunciation lexicon. This requirement is for keyword search. We use a fuzzy phonetic search to improve results that can only use one lexicon for all the words.

2.5. Hitlist Combination

Hitlist combination is the most computationally expensive approach to system combination. Given multiple hitlists, the scores for the individual hits are combined to maximize their score on a tune set—while the combination weights could in principle be estimated for each individual keyword, we only generate a weight for the set of IV keywords and OOV keywords. While the actual combination is quick, the extra computation comes from performing a separate decoding and keyword search for each system. Multiple lattices or indexes must be stored, significantly increasing the storage cost. This increased cost also comes with increased flexibility. No restrictions are placed on state targets or lexicons; each system could even use a different phone set. Unlike other approaches to system combination, hitlist combination lends itself to a relatively straightforward approach to learning combination weights that adds little additional overhead [2].

3. Experimental Setup

We use the Sage ASR toolkit [16]. Sage is BBN's newly developed STT platform that integrates technologies from multiple sources, each of which has a particular strength. In Sage, we combine proprietary sources, such as BBN's Byblos [17], with open source toolkits, such as Kaldi [18], CNTK [19] and Tensorflow [20]. For example, DNN can be trained using Byblos, Kaldi nnet1 [21] or nnet2, CNN using Kaldi or Caffé [22], and LSTM using Kaldi as well as CNTK. Sage also includes software supporting keyword search from Byblos [23, 24]. The integration of these technologies is achieved by creating wrapper modules around major functional blocks that can be easily connected or interchanged. In addition, Sage software has been designed to make it easy for a group of researchers to use the system, to transfer experiments from one person to another, to keep track of partial runs, etc. Sage also includes a cross-toolkit FST recognizer that supports models built using the various component technologies.

The baseline DNN system uses a bottleneck MLP with a bottleneck layer of 40 nodes. These features are used for speaker-adapted training (SAT), and the final features are the fMLLR transformed bottleneck features. Thirteen frames of the fMLLR features are stacked together as input to the sequence trained DNN acoustic model. Each of the six hidden layers has 2048 nodes, and the output layer has approximately 4500 tied-state targets. Two additional acoustic models are trained. The first is a DNN using bottleneck features derived from a CNN. It has the same structure as the bottleneck MLP, except two convolutional layers are prepended. The second acoustic model is a CNN—two convolutional layers followed by four fully connected layers—using the filter bank features directly.

All experiments are performed on data from the IARPA Babel project [25]. We selected four development languages from the final year of the program: Amharic (IARPA-babel307b-v1.0b), Guarani (IARPA-babel305b-v1.0c), Igbo (IARPA-babel306b-v2.0c), and Pashto (IARPA-babel104b-v0.bY). For each language, the full language pack (FLP) is used, containing approximately 40 hours of transcribed audio—the audio is conversational telephone speech collected in a variety of conditions. Lexicons are derived using simple G2P rules [26]. Trigram language models are built only from the transcriptions. This data can be further augmented with data collected from the web [27]. Decoding is performed on an additional 10 hours of development data, and keyword search uses a set of approximately 2000 keywords for each language. Both whole word and phonetic search are used for keyword spotting [2].

Actual term-weighted value (ATWV) is the primary measure of interest for the IARPA Babel program. ATWV was also used in the NIST 2006 Spoken Term Detection evaluation [28]. In this performance metric, all keywords are equally weighted.

Language	First Pass Model	WER	ATWV
Amharic	GMM	44.2	0.582
Amharic	CNN	43.6	0.583
Guarani	GMM	45.9	0.564
Guarani	CNN	45.7	0.571
Igbo	GMM	55.7	0.335
Igbo	CNN	55.5	0.339
Pashto	GMM	48.2	0.411
Pashto	CNN	48.1	0.411

Table 1: Comparison of using GMM and CNN models for first pass decoding in a DNN system.

Language	Baseline	Joint	Lattice	Hitlist
Amharic	0.583	0.599	0.600	0.607
Guarani	0.571	0.584	0.579	0.587
Igbo	0.339	0.353	0.357	0.354
Pashto	0.411	0.428	0.430	0.432

Table 2: Comparison of system combination approaches using CNN and DNN acoustic models. Joint refers to joint decoding. Lattice refers to lattice combination. Hitlist refers to hitlist combination. Baseline refers to best single system performance.

Missing a single occurrence of a rare word can affect the final score as much as missing a more common word dozens of times. Wegmann et al. have a more detailed discussion of ATWV in relation to the IARPA Babel program [6]. While ATWV numbers are commonly reported for in-vocabulary and out-of-vocabulary (OOV) keywords separately, our focus is on overall performance. The number of OOV keywords in the FLP task is relatively small, so we do not spend the extra computational effort on techniques designed to specifically detect them [4, 29, 30, 31, 32].

4. Experimental Results

4.1. Cross Adaptation

Table 1 shows a performance comparison for our baseline DNN system with either a GMM or CNN for first pass decoding. Overall gains from using the CNN are small for all languages, but it never produces a degradation in performance. These gains come from the alignments produced for fMLLR feature computation—the CNN typically has a 5% lower WER compared to the GMM. The largest computational requirement is the training of the CNN itself, however, this can be done in parallel to the DNN model. Since the goal of this work is to explore system combination, the training of the CNN is required anyway. While the performance may not justify using the model for first pass decoding, it also decreases the overall decoding time by an average of 15%. In our standard setup, the GMM requires two passes to generate the fMLLR features; the CNN only requires one. Given the decrease in decoding time and the associated gains in performance, all further experiments will use the CNN model for first pass decoding. Note that when discussing timing information in the following sections, we consider the DNN system using a GMM for first pass decoding as a baseline. All numbers will be relative to that system.

4.2. CNN + DNN System Combination

In addition to being used for first pass decoding, the CNN system can be incorporated in three additional ways as described in Section 2. A comparison of ATWV performance for each type is shown in Table 2. All combination types give about 1.5 to 2 points absolute improvement in ATWV. Hitlist combination tends to give the best overall performance, but the other two types are close. Note that the joint decoding approach gives similar performance and still requires approximately 5% less decoding time than the baseline system. It is able to beat the baseline system because the performance gain from using the CNN in the first pass is more than the cost of decoding with two models. Lattice combination and hitlist combination both require double the decoding time and hitlist combination also doubles the keyword search time and storage requirements.

Language	Baseline	Feature	Joint	Lattice	Hitlist
Amharic	0.583	0.592	0.603	0.606	0.607
Guarani	0.571	0.560	0.582	0.588	0.585
Igbo	0.339	0.351	0.365	0.364	0.365
Pashto	0.411	0.427	0.431	0.437	0.436

Table 3: Comparison of system combination approaches using DNN acoustic models with CNN-BN and DNN-BN features. Feature refers to feature combination.

4.3. CNN-BN + DNN-BN Feature Combination

We can incorporate the CNN in one additional way. Instead of using the CNN as an acoustic model, it can be used to train bottleneck features. This produces a second set of features in addition to the standard bottleneck features trained from a DNN. We label the system using the CNN bottleneck features as CNN-BN and the standard system using bottleneck features from a DNN as DNN—though both systems ultimately use a DNN as the acoustic model. Systems using these features can be combined in the same way as in the previous section, but the features themselves can also be combined as input to a single model. Results for the four approaches are presented in Table 3.

Performance for lattice combination and hitlist combination are similar. Joint decoding is slightly behind and feature combination is even further behind. Feature combination provides a gain over the baseline—except for Guarani—but is far behind the results of the other techniques. It is a more efficient system to train and use, the decrease in computational cost is unlikely to be worth the decrease in performance though.

The overall pattern of results is similar to the previous CNN+DNN results. The extra bottleneck feature computation is more expensive than using filterbank features, so joint decoding uses all of the gains from the CNN as a first pass decoding and takes approximately the same amount of time as the baseline system. Note that since we are still using the CNN for first pass decoding, these results actually require a total of three systems to be trained. In the next section we take full advantage of all three models by combining them all.

4.4. Combining All Systems

We can also combine all three systems. Since the systems can always be combined in two stages—first combining two systems with one approach, and then combining with the final system using another—there are a large number of possible combination strategies. For simplicity, we only consider combination in a single step using the same approach for all three systems. Results are shown in Table 4.

With three systems, the overall pattern of results is similar. Joint decoding ranges from 0.4 to 1.2 points worse than hitlist combination, and lattice combination is never more than

Language	Baseline	Joint	Lattice	Hitlist
Amharic	0.583	0.606	0.615	0.618
Guarani	0.571	0.590	0.594	0.594
Igbo	0.339	0.366	0.367	0.372
Pashto	0.411	0.440	0.445	0.444

Table 4: Comparison of system combination approaches using all three systems.

0.5 points worse. It is expected that hitlist combination would further outperform the other types of system combination as the number of systems increased. This is not the case in these results. Even with three systems, joint decoding only requires 15% more decoding time than the baseline system. It is clearly an efficient method to increase performance at decode time, but there is a limit. Moving from one system to two gives a greater gain than moving from two to three; additional systems would unlikely see further significant gains.

5. Discussion

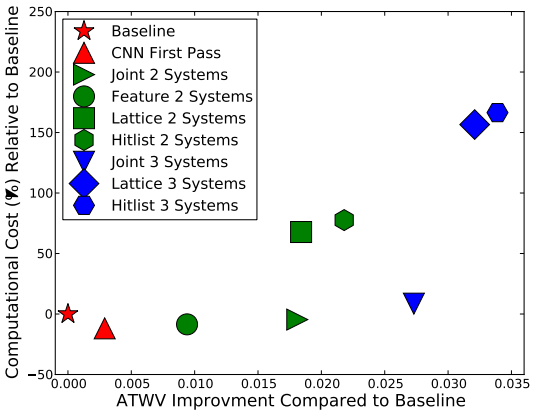

Figure 1: Relative computational cost relative to baseline versus ATWV improvement.

In the previous sections, we briefly discussed the decoding cost as the number of systems increased. Figure 1 makes the trade-off between performance and computational time explicit. These values are averaged over all four languages for each approach. Computational cost is the sum of decoding, keyword search (approximately half the cost of decoding), and any additional cost related to the combination approach. This cost is in terms of total clock time. In practice, these operations are parallelized, but this does not affect the relative differences. All results are relative to the baseline DNN systems using a GMM for first pass decoding reported in Table 1. Note that for the two system results—except for feature combination—only the CNN+DNN combination is considered. Since it was using the CNN for first pass decoding, the CNN-BN+DNN-BN results (c.f. Section 4.3) actually used three systems in total.

The figure makes it clear how little joint decoding and feature combination affect the overall computational time compared to hitlist combination and lattice combination. Joint decoding provides the majority of the performance gain, with negligible cost. If an additional gain is required, it can be obtained through lattice combination. Hitlist combination sometimes brings further gains, but requires each lattice (or its related index) to be stored on disk for keyword search.

When combining three systems, hitlist combination has a slight edge over lattice combination. Intuitively, the gap between hitlist combination and other techniques should only grow as the number of systems increases. However, this is not the case in our results. Part of the reason may be due to the low OOV rate. In previous years of the IARPA Babel program, when the amount of training data was less, system combination was especially important because it increased the total number of hits for OOV keywords. Comparing the recall rates for the different combination experiments shows little variation.

Hitlist combination has another implicit advantage; the weights used during combination can be easily trained even for large numbers of systems. For joint decoding and lattice combination, we give each system an equal weight. Preliminary experiments testing other weights had little effect on final performance. There is no simple way to learn ideal weights. The obvious approach would be a grid search over possible settings, but this would require many passes of decoding and keyword search, eliminating the computational advantage these techniques hold over hitlist combination. As the number of systems grows, this search approach becomes even more impractical. In future work, we plan to investigate more efficient methods of learning the weights for joint decoding and lattice combination in order to further close the gap with hitlist combination.

As mentioned previously, each approach also comes with limitations or restrictions. Joint decoding requires all systems to have the same state targets—though, as previously mentioned—this restriction could be lifted if there is a map between to two sets of targets. Lattice combination requires all systems use the same lexicon. In order to perform this study, we built all systems so as to satisfy the requirements of all approaches. This potentially limited the benefits of lattice and hitlist combination. Previous work has shown large gains by combining hitlists from whole-word decodes and hitlists from subword decodes [33, 29]. Joint decoding cannot function in that scenario.

6. Conclusions

We have shown that using a CNN acoustic model for first pass decoding can give small gains over the commonly used GMM model. The CNN can also be used in joint decoding to further improve performance with little additional computational cost. Hitlist combination is generally assumed to be the best performing approach to system combination for keyword spotting, but it is also the most expensive due to the multiple decodings and searches it requires. When combining two systems, we have demonstrated joint decoding performs nearly as well as hitlist combination with a minimal increase in overall decoding time. It provides the best trade-off between performance and cost. Lattice combination provides similar performance and reduces storage cost compared to hitlist combination.Feature combination is a little behind the other techniques, but only requires a single acoustic model.

Hitlist combination can give a small additional gain, but requires a large amount of computational effort and storage. These effects hold even when moving to three systems. Assuming the systems meet the restrictions required by joint decoding, it would be the preferred method of system combination. Hitlist combination should be reserved for when the systems cannot be combined through joint decoding or lattice combination, where the gains from hitlist combination are potentially greater.

7. References

[1] R. Hsiao, T. Ng, F. Grézl, D. Karakos, S. Tsakalidis, L. Nyugen, and R. Schwartz, "Discriminative semi-supervised training for keyword search in low resource languages," in *ASRU*, 2013, pp. 440–445.

[2] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nyugen, 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 *Proceedings of IEEE ASRU*, 2013.

[3] K. Knill, M. Gales, A. Ragni, and S. Rath, "Language independent and unsupervised acoustic models for speech recognition and keyword spotting," in *Interspeech*, 2014, pp. 20–26.

[4] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proceedings of IEEE ASRU*, 2013, pp. 416–421.

[5] V.-B. Le, L. Lamel, A. Messaoudi, W. Hartmann, J. L. Gauvain, C. Woehrling, J. Despres, and A. Roy, "Developing STT and KWS systems using limited language resources," in *Proceedings of Interspeech*, 2014.

[6] S. Wegmann, A. Faria, A. Janin, K. Riedhammer, and N. Morgan, "The tao of ATWV: Probing the mysteries of keyword search performance," in *Proceedings of IEEE ASRU*, 2013, pp. 192–197.

[7] H. Wang, A. Ragni, M. Gales, K. Knill, P. Woodland, and C. Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," in *Interspeech*, 2015, pp. 3660–3664.

[8] A. Morris, A. Hagen, H. Glotin, and H. Bourlard, "Multi-stream adaptive evidence combination for noise robust ASR," *Speech Communication*, vol. 34, no. 1, pp. 25–40, 2001.

[9] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[10] S. Rath, K. Knill, A. Ragni, and M. Gales, "Combining tandem and hybrid systems for improved speech recognition and keyword spotting on low resource languages," in *Interspeech*, 2014, pp. 835–839.

[11] R. Schluter, L. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in *ICASSP*, vol. 4, 2007, pp. 649–652.

[12] B. Zhang, S. Matsoukas, and R. Schwartz, "Discriminatively trained region-dependent transform for speech recognition," in *ICASSP*, 2006.

[13] P. Swietojanski, A. Ghoshal, and S. Renals, "Revisiting hybrid and GMM-HMM system combination techniques," in *ICASSP*, 2013, pp. 6744–6748.

[14] O. Siohan and D. Rybach, "Multitask learning and system combination for automatic speech recognition," in *ASRU*, 2015, pp. 589–595.

[15] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[16] R. Hsiao, R. Meermeier, T. Ng, Z. Huang, M. Jordan, E. Kan, T. Alumäe, J. Silovsky, W. Hartmann, F. Keith, O. Lang, M. Siu, and O. Kimball, "Sage: The new BBN speech processing platform," in *submission to Interspeech*, 2016.

[17] Y. Chow, M. Dunham, O. Kimball, M. Krasner, G. Kubala, J. Makjoul, P. Price, S. Roucos, and R. Schwartz, "BYBLOS: The BBN Continuous Speech Recognition System," in *ICASSP*, 1987.

[18] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proceedings of IEEE ASRU*, 2011.

[19] A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, R. Hoens, X. Huang, Z. Huang, V. Ivanov, A. Kamenev, P. Kranen, O. Kuchaiev, W. Manousek, A. May, B. Mitra, O. Nano, G. Navarro, A. Orlov, M. Padmilac, H. Parthasarathi, B. Peng, A. Reznichenko, F. Seide, M. L. Seltzer, M. Slaney, A. Stolcke, Y. Wang, H. Wang, K. Yao, D. Yu, Y. Zhang, , and G. Zweig, "An introduction to computational networks and the computational network toolkit," Microsoft, Tech. Rep. MSR-TR-2014-112, August 2014.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[21] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Interspeech*, 2013, pp. 2345–2349.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[23] I. Bulyko, O. Kimball, M. Siu, J. Herrero, and D. Blum, "Detection of unseen words in conversational mandarin," in *Proceedings of ICASSP*, 2012, pp. 5181–5184.

[24] T. Ng, R. Hsiao, L. Zhang, D. Karakos, S. H. Mallidi, M. Karafiát, K. Vesely, I. Szoke, B. Zhang, L. Nyugen, and R. Schwartz, "Progress in the BBN keyword search system for the DARPA RATS program," in *Interspeech*, 2014, pp. 959–962.

[25] M. Harper, "IARPA solicitation IARPA-BAA-11-02," http://www.iarpa.gov/solicitations_babel.html, 2011.

[26] M. Davel, E. Barnard, C. van Heerden, W. Hartmann, D. Karakos, R. Schwartz, and S. Tsakalidis, "Exploring minimal pronunciation modeling for low resource languages," in *Interspeech*, 2015, pp. 538–542.

[27] L. Zhang, D. Karakos, W. Hartmann, R. Hsiao, R. Schwartz, and S. Tsakalidis, "Enhancing low resource keyword spotting with automatically retrieved web documents," in *Interspeech*, 2015, pp. 839–843.

[28] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Proceedings of ACM SIGIR*, 2007, pp. 51–55.

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

[30] W. Hartmann, L. Lamel, and J. L. Gauvain, "Cross-word subword units for low-resource keyword spotting," in *SLTU*, 2014, pp. 112–117.

[31] Y. He, B. Hutchinson, P. Baumann, M. Ostendorf, E. Fosler-Lussier, and J. Pierrehumbert, "Subword-based modeling for handling OOV words in keyword spotting," in *Proceedings of IEEE ICASSP*, 2014.

[32] D. Karakos and R. Schwartz, "Combination of search techniques for improved spotting of OOV keywords." in *Proceedings of ICASSP*, 2015, pp. 5336–5340.

[33] W. Hartmann, V.-B. Le, A. Messaoudi, L. Lamel, and J. L. Gauvain, "Comparing decoding strategies for subword-based keyword spotting in low-resourced languages," in *Proceedings of Interspeech*, 2014.