JHU-CLSP SYSTEM DESCRIPTION FOR OPENSAT2017 EVALUATION

Vimal Manohar[1]*, Jan Trmal*[1]*, Jesús Villalba*[1]*, Raghavendra Reddy Pappagary*[1]*,*
Jonas Borgstrom[2]*, Pedro Torres-Carrasquillo*[2]*, Doug Sturim*[2]*,*
Najim Dehak[1]*, Daniel Povey*[1]*, Sanjeev Khudanpur*[1]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA.
[2]MIT Lincoln Laboratory, Lexington, MA, USA
{vmanoha1,yenda,jvillal7}@jhu.edu

ABSTRACT

The JHU-CLSP submission to the OpenSAT2017 speech activity detection evaluation consists of a fusion of several neural network based VADs. These systems combine TDNN and LSTM layers. Two of the VADs were trained on Babel and Fisher data augmented with reverberation, noise and music. Another system was trained using only the OpenSAT VAST development data. A fourth system was trained on TIMIT and CMU Child data also augmented with noise and reverberation.

The JHU system for Pashto SST and KWS is based mostly on a conventional, DNN based LVCSR pipeline. Notable features in our system include the use of a lexicon expansion greatly increasing KWS search performance, a LF-MMI TDNN-LSTM acoustic model KWS pipeline optimized for OOV search.

Index Terms— SAD, VAD, TDNN, LSTM, SST, KWS, LF-MMI

1. SAD SUBMISSION

The JHU-CLSP submission to the OpenSAT2017 speech activity detection evaluation consists of a fusion of several neural network based VADs. The speech activity detection networks contain TDNN layers followed by LSTM or statistics pooling [1] for long-context. The neural network computes the speech/non-speech posterior probabilities. Afterwards, we use those posteriors as emission probabilities of a two-states HMM with duration constraints and obtain the speech activity labels by Viterby decoding.

For the VAST condition, we submitted five systems:

- Primary: majority voting fusion of contrastives 1–4.
- Constrastive 1: TDNN with statistics pooling trained on 10 Babel languages [2] and Fisher English [3] perturbed with various room impulse responses, additive noise and music (JHU set).
- Contrastive 2: TDNN-LSTM trained on the same data as contrastive 1.
- Contrastive 3: TDNN with stats pooling trained on OpenSAT2017 dev set.
- Contrastive 4: TDNN with stats pooling trained on OpenSAT2017 dev set; and TIMIT and CMU Child with noise and reverberation (MITLL set).

For the Babel condition, we submitted one primary system, which is the same as contrastive 1 in the VAST condition.

1.1. Training data

We trained the neural networks on three different datasets. The first one was prepared by the JHU team while the second one was prepared by the MITLL team. The third dataset was the OpenSAT development set.

1.1.1. JHU training set

As training data for the neural network, we use data from 10 different Babel languages that were publicly available - Assamese, Bengali, Cantonese, Georgian, Haitian, Pashto, Swahili, Tagalog, Turkish, Vietnamese, and Fisher English. We select about 10 hours from each corpus. To improve the robustness of the speech activity detector to unseen data, we add various perturbations:

- Reverberation: We use synthetic room impulse responses (RIRs) [4]. For each recording, we create 10 copies each reverberated with a randomly selected RIR.
- Additive noise: Many different kinds of noise recordings, both foreground and background noises, from the MUSAN corpus [5] were added. For each recording from the base corpus, several noise recordings were randomly selected, perturbed using the synthetic RIRs and added at different points in the recording. For 5 copies, we added noise from MUSAN corpus scaled between 20dB and -5dB; for the other 5 copies, we added music from MUSAN corpus scaled between 5dB and -5dB.

- Speed and volume perturbation: We apply speed perturbation [6] by randomly choosing one of the 3 speeds – 0.9, 1.0 and 1.1. We additionally scale the volume randomly by a scale between 0.03125 and 2. This is to help the neural network be less sensitive to the energy of the input signal. This is important for robustness because the input features to the neural networks do not have cepstral mean subtraction or any other normalization.

The datasets that we used don't include speech activity detection labels. To infer those labels we performed force alignment using HMM-GMM systems trained on the respective corpora. For the Babel languages, we used the data in the Full LP condition to train the HMM-GMMs. For the Fisher English, we used a 100 hour subset to train the HMM-GMMs.

We align the manual segments of each corpus with the corresponding transcription using a speaker-adaptively trained HMM-GMM system to create alignments. These are converted into phone labels, which are deterministically mapped to 2 classes – speech and silence.

We separately decode both the manual segments and the regions outside the manual segments using a speaker-independent LDA+MLLT HMM-GMM system to get best path alignments. For outside the manual segments, we keep only frames that are silence. For the manual segments, we keep frames for which the transcription-constrained alignments and the decoded hypothesis are both the same class (silence or speech).

1.1.2. MITLL training set

The multicondition data was simulated by concatenating utterances from the TIMIT and CMU Child corpora. For each simulated file, 10 utterances were chosen from the two corpora, with randomly set intervals and gains. Reverberation was introduced by applying a room impulse response from the Voice Home corpus. Finally, additive noise was introduced from the noise and music subsets of the MUSAN corpus, and mixed at an SNR chosen between 20 dB and -3dB.

1.1.3. OpenSAT development set

We also trained a VAD on the development set in order to have an upper bound of the performance that we could achieve by training and testing on the same dataset.

1.2. Neural network description

We use a time-delay neural network [7] with either LSTMP layers [8] or statistics pooling [1] layers interleaved with TDNN layers to provide a long-context. We use 4 TDNN layers and 2 layers of LSTMP or Statistics pooling. The overall context of the network is set to be around 1s, with around 0.8s of left context and 0.2s of right context. The network is trained with cross-entropy objective to predict the speech/non-speech labels.

1.2.1. Input features

As input features to the neural networks, we use 28-dimensional MFCC features extracted from the bandwidth 330Hz to 3000Hz, which corresponds to the telephone bandwidth. This makes the neural network usable for both telephone speech and microphone speech.

1.2.2. Auxiliary task outputs

The neural network is trained to predict some auxiliary features in addition to speech/non-speech labels. These act as regularizers to increase the robustness of the neural networks. At test time, these outputs are not computed, and only the main output for speech/non-speech detection is evaluated. We use two auxiliary features:

- Ideal-ratio-mask (IRM) – This is the ratio of Mel filterbank coefficients of the clean signal to the sum of Mel filterbank coefficients of clean and noise signals. This is converted to log-domain so that they represent log-probability of the sub-band containing speech (clean) energy.

$$\text{IRM}(f) = \frac{S(f)}{S(f) + N(f)}, \quad f = 1, 2, \ldots, K \qquad (1)$$

  where $S(f)$ and $N(f)$ are the Mel filterbank coefficients of the clean and noise signals respectively for sub-band f.

- Music labels – For the recordings, where we add music during perturbation, we add auxiliary labels that specify music/non-music.

The auxiliary tasks are trained simultaneously with the main speech/non-speech task in a multi-task framework. For predicting music/non-music labels, we use a cross-entropy objective. For predicting sub-band log-IRM features, we use cross-entropy objective.

1.3. Decoding

Given the test data, its MFCC features are propagated through the neural network to get speech/non-speech log-posteriors, which are converted to log-likelihoods by subtracting the log-priors. The log-likelihoods are fed to a HMM Viterby decoder. The HMM has a minimum duration constraint of 10 frames for silence and 30 frames for speech. At the end of the minimum duration, there is a self-loop with probability 0.99 for speech and 0.9 for silence. To bias the decoding towards, speech we allow a transition from the end of silence class to the start of the speech class with a probability of 0.009 and a

Table 2. Total CPU time (s) on VAST eval data

Process	Statistics pooling	LSTM
Nnet propagation	2056	20990
Decoding	254	466
Post-processing	25	17
Total	2335	21473

transition to start of silence class with a probability of 0.0009. We force a transition to the start of speech class at the end of silence class with a probability of 0.009. The remaining probability mass is for transition to end state. The best path from the HMM decoder is post-processed to add a 0.2s padding on the edges of the speech segments, followed by merging small silence regions that are less than 0.3s long into the adjacent speech regions.

1.4. Results

Table 1 shows the results of or systems on the VAST development set.

Table 1. VAD Results on the VAST dev data

System	Avg Cost	Avg P_{miss} (%)	Avg P_{fa} (%)
Primary (Fusion)	0.083	5.06	18.03
Contrast 1 (Stats JHU)	0.088	5.56	18.66
Contrast 2 (LSTM JHU)	0.090	3.22	26.6
Contrast 3 (Stats OpenSAT dev)	0.077	6.82	10.41
Contrast 4 (Stats MITLL + OpenSAT dev)	0.119	12.7	9.42

1.5. Timing

Table 2 shows CPU time for the two neural architectures that we used.

2. KALDI OPENSAT PASHTO SYSTEM

Our system is based mostly on a conventional, DNN based LVCSR pipeline. Notable features in our system include the use of a lexicon expansion greatly increasing KWS search performance, a LF-MMI TDNN-LSTM acoustic model KWS pipeline optimized for OOV search. It's an improved version of system described in [22]

2.1. Acoustic Models

2.1.1. Neural network architecture

The acoustic model is a hybrid hidden markov model deep neural network (HMM-DNN), where the neural network is composed of 3 LSTM layers and 7 TDNN layers (the stacking is [3xTDNN, 1xLSTM, 2xTDNN, 1xLSTM, 2xTDNN]. The neural network is trained using the cross-entropy objective function.

2.1.2. Features

40 dimensional MFCCs along with 3 dimensional pitch features spliced over a context of 5 frames ($[t-2, t+2]$) were used as the input to the neural network for every time step. i-vectors when provided as an input to a neural network have been shown to perform speaker adaptation of the affine transforms [10]. Further 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.

2.1.3. Other details

The training data consists only of the transcribed Pashto speech data. Speed-perturbation [11] based data augmentation technique was used to increase the training data 3-fold. The parallel DNN training technique described in [14], was used to efficiently train the neural network across multiple GPUs. Using this training technique we were able to train with 200 hours of augmented speech data using up to 6 GPUs in 12 hours. The alignments for training the TDNN-LSTM system were generated using a HMM-GMM system trained using the PLP+Pitch features. The training algorithm was using backstitch training [20] in combination with [21].

2.2. Lexicon

2.2.1. Lexicon Expansion

The main idea is to automatically generate millions of distinct lexicon entries whose pronunciations are phonotactically plausible in that language. An OOV (key)word in the test speech will then have a good chance of being decoded as a similar-sounding lexicon entry, obviating the need for a separate phonetic decoding pass or a separate subword index for OOV search. The word-lattices may be searched directly for the OOV keyword, with the proxy-based method of [16] to mitigate differences between the correct spelling (of the keyword) and the spelling generated during this automatic lexicon expansion. Apart from the implementation details, the main idea is quite straightforward. First, generate a language model based on syllables (syllable boundaries were provided in OpenSAT dataset). Then, oversample massively the syllable LM for random sequences, note the probability of each sequence (=word) and keep 1M of the most probable (after removing the sequences(words) already known). After that, combine this probability with a small discount and insert these new words into the lexicon. For a more detailed description of the algorithm, see [15].

2.3. Language Modeling

A trigram language model was trained using the in-domain text (i.e.training data transcripts). This trigram LM was used to generate the lattices.

2.3.1. Pronunciation probabilities

In our system, pronunciation probabilities can be used in the lexicon to distinguish multiple pronunciations of the same word. Before system training, since we have no prior knowledge of the word pronunciations, we assign the same probability mass to each pronunciation of the same word. We use this initial lexicon to train an early stage triphone system. We align the training data with this early stage system, and collect "word pronunciation" pair counts, with which we estimate the probability of the pronunciation given a certain word $p(\mathrm{pron}|\mathrm{word})$, and update the lexicon. Later stages training of our system is done with this updated lexicon.

2.3.2. Word position dependent silence probabilities

We also model the probability of silence before and after each pronunciation explicitly. With the same early stage triphone system used above, we align the training data and collect the counts for silence before and after a certain pronunciation. We then estimate the quantities $p(\mathrm{sil}-\mathrm{after}|\mathrm{pron})$ and $p(\mathrm{sil}-\mathrm{before}|\mathrm{pron})$ for each "word pronunciation" pair in our lexicon, and encode those quantities into our lexicon finite state transducer (FST) by adding additional silence states. The updated lexicon FST is then used for training later stages of our system. The probabilities are re-estimated at several stages in our system. More details about the enhanced lexicon FST can be found in [13].

2.4. Decoding

A HMM-DNN acoustic model trained using LDA+MLLT features was used to perform speech segmentation using a phone level decoding graph. The segments identified as speech were then decoded using the trigram word level decoding graph. The rescored lattices were provided as an input to the KWS system.

2.5. Keyword Search

The keyword search pipeline is primarily word-based. The word-level lattices are converted into an inverted index as described in [14]. The pipeline is conceptually based on the Babel program Base-Period (BP) and Babel Program Option-Period-1 (OP1) pipeline described in [15], but was completely redesigned. The OOV search performance was one of the primary objectives.

2.5.1. Handling the OOVs

As the OOVs are not part of the lattice, they are not part of the index either. To allow search for the OOV terms (i.e. terms whose at least one constituent word is OOV), we developed a technique called Proxy-keyword search [16]. The method finds the phonetically closest sequence of the IV (in-vocabulary) words. For graphemic systems, the generation of the pronunciation is trivial, for "real" phonetic systems, we use Sequitur [17].

Table 3. STT results

dataset	WER	Sub	Ins	Del
dev10h.pem	38.0	24.5	9.6	3.9
dev10h.seg	38.5	23.7	10.9	3.9

Table 4. KWS results

Keyword List	ATWV	MTWV	OTWV	STWV
kwlist1	0.6346	0.6353	0.7298	0.8995
kwlist2	0.6267	0.6274	0.7436	0.9165
kwlist3	0.5662	0.5662	0.6680	0.8497
kwlist4	0.6131	0.6134	0.7265	0.9069

2.5.2. Score normalization and calibration

For score normalization, we use the Keyword specific threshold technique (KST) from [19]. For the dev set, the N_{true} constant is determined experimentally, by trying several values between 1 and 2 and choosing the one that maximizes ATWV. For the eval set, we use the devset's N_{true} value.

2.5.3. Results fusion

The primary (and the only) OpenSAT submission didn't include fusion. The reason is that so far we didn't find any improvement of combining subword-level search with the word-level (+proxies) search. We didn't fuse multiple ASR systems (multiple acoustic models) either, as we are primarily interested in "practical" systems.

2.6. Results

2.6.1. STT results on dev speech

The PEM is the segmentation provided in the Babel Pashto training pack. The SEG segmentation is the OpenSAT SAD segmentation (that was used to segment the eval data).

2.6.2. KWS results on dev speech

2.7. Hardware

A variable number of 16-core (Intel(R) Xeon(R) CPU E5-2680) machines was used. The 16-core cpus are not that important, as we used SGE to distribute the workload – any reasonable up-to-date cpus with say 4 cores would be sufficient, as we do not really parallelize on core level but on task level. The amount of per-core memory was 2 GB. The training of

Table 5. STT-KWS Resources

Ingestion Elapsed Time (hh:mm:ss)	108:34:28
Ingestion Total CPU Time (hh:mm:ss)	1396:46:08
Ingestion Total GPU Time (hh:mm:ss)	605:21:47
Ingestion Maximum CPU Memory (gbytes)	16G
Ingestion Maximum GPU Memory (gbytes)	8G
Search Elapsed Time (hh:mm:ss)	37:35
Search Total CPU Time (hh:mm:ss)	54:07:14
Search Total GPU Time (hh:mm:ss)	00:00:00
Search Maximum CPU Memory (gbytes)	16G
Search Maximum GPU Memory (gbytes)	16G

FullLP system was done using 64 cores (4 nodes). GPUs were used for training neural networks. Each training session consumes up to 6 GPUs, as parallel training was used. The detailed timing info will be provided in the next section. The typical size of a complete system (including lattices) is around 200 GB. Indexing and search was done on 198 cpus.

2.8. Timing

NB: these are only best-effort estimates, as we run many other experiments during training of the system so without running the final recipe again it's hard to isolate the individual time factors.

2.9. Conclusion

This is a single pass TDNN+LSTM system achieving 38 % WER on the dev set. No other external data was used during training of the system.

3. REFERENCES

[1] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic modelling from the signal domain using cnns". in INTERSPEECH, 2016, pp. 34343438.

[2] M. Harper, "IARPA Babel Program". 2014.

[3] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: A resource for the next generations of speech-to-text". in LREC, 2004, vol. 4, pp. 6971.

[4] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition", in Acoustics, speech and signal processing (icassp), 2017 ieee international conference on, 2017, pp. 52205224.

[5] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus", arXiv preprint arXiv:1510.08484, 2015.

[6] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition". in INTERSPEECH, 2015, pp. 35863589.

[7] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts". in INTERSPEECH, 2015, pp. 32143218.

[8] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling", in Fifteenth annual conference of the international speech communication association, 2014.

[9] Chen, Kai, and Qiang 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 24.7 (2016): 1185-1193.

[10] Saon, George, et al. "Speaker adaptation of neural network acoustic models using i-vectors." ASRU. 2013.

[11] Ko, Tom, et al. "Audio augmentation for speech recognition." Proceedings of INTERSPEECH. 2015.

[12] Daniel Povey, Xiaohui Zhang, Sanjeev Khudanpur, "Parallel training of DNNs with Natural gradient and Parameter Averaging", , AISTATS 2015.

[13] Chen, Guoguo, et al. "Pronunciation and silence probability modeling for ASR." Proceedings of INTERSPEECH. 2015.

[14] D. Can and M. Saraclar, "Lattice Indexing for Spoken Term Detection", in IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 8, pp. 2338-2347, Nov. 2011. doi: 10.1109/TASL.2011.2134087

[15] J. Trmal et al., "A keyword search system using open source software," Spoken Language Technology Workshop (SLT), 2014 IEEE, South Lake Tahoe, NV, 2014, pp. 530-535. doi: 10.1109/SLT.2014.7078630

[16] G. Chen, O. Yilmaz, J. Trmal, D. Povey and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task", Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, Olomouc, 2013, pp. 416-421. doi: 10.1109/ASRU.2013.6707766

[17] Maximilian Bisani, Hermann Ney, "Joint-sequence models for grapheme-to-phoneme conversion", Speech Communication, Volume 50, Issue 5, May 2008, Pages 434-451, ISSN 0167-6393, http://dx.doi.org/10.1016/j.specom.2008.01.002.

[18] Sami Virpioja, Peter Smit, Stig-Arne Gr?nroos, and Mikko Kurimo. “Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline”. Aalto University publication series SCIENCE + TECHNOLOGY, 25/2013. Aalto University, Helsinki, 2013. ISBN 978-952-60-5501-5.

[19] D. R. H. Miller, et al., “Rapid and accurate spoken term detection”, in Proc. of InterSpeech, pp. 314-317, 2007.

[20] Yiming Wang et al., “Backstitch: Counteracting Finite-sample Bias via Negative Steps”, Interspeech 2017 (accepted)

[21] Gaofeng Cheng et al, “An exploration of dropout with LSTMs”, Interspeech 2017 (accepted)

[22] Jan Trmal et al. , “The Kaldi OpenKWS System: Improving Low Resource Keyword Search”, Interspeech 2017 (accepted)