# EXPLOITING DIVERSITY FOR SPOKEN TERM DETECTION

*Lidia Mangu, Hagen Soltau, Hong-Kwang Kuo, Brian Kingsbury and George Saon*

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA

## ABSTRACT

The paper describes a state-of-the-art spoken term detection system in which significant improvements are obtained by diversifying the ASR engines used for indexing and combining the search results. First, we describe the design factors that, when varied, produce complementary STD systems and show that the performance of the combined system is 3 times better than the best individual component. Next, we describe different strategies for system combination and show that significant improvements can be achieved by normalizing the combined scores. We propose a classifier-based system combination strategy which outperforms a highly optimized baseline. The system described in this paper had the highest accuracy in the 2012 DARPA RATS evaluation.

***Index Terms***— spoken term detection, keyword spotting, audio indexing, system combination

## 1. INTRODUCTION

Finding a spoken or written `term` in a collection of audio recordings is a fundamental problem in automatic speech processing. By `term` (or `keyword`) we mean a word or a sequence of words. Research in spoken term detection (STD) has been substantially advanced by competitive evaluations. The first evaluation was the NIST STD 2006 evaluation [1], in which participants built systems for English, Arabic and Chinese. Recently, there is renewed interest in evaluating spoken term detection systems in a variety of languages and audio conditions. In the DARPA RATS (Robust Automatic Transcription of speech) program, existing Arabic Levantine telephone conversations are retransmitted through 8 different communication channels with different degrees of noise and channel distortion. One of the four RATS tasks is keyword search on this highly degraded speech. In this paper we describe the system we deployed in the first RATS evaluation, which was held in February 2012. We also present improvements obtained after the evaluation by replacing a linear combination with a classifier-based combination strategy. There are three main contributions in this paper. (1) To our knowledge, this is the first use of diverse ASR systems for improving STD performance. We deliberately design diverse and complementary ASR components (i.e., front ends, acoustic models, etc.), while the level of diversification in prior work is limited to combining STD systems which use word and sub-word models [2, 3, 4, 5, 6]. (2) We describe a score normalization technique which results in significant STD improvements. (3) We present a novel classifier-based combination method for merging STD results. In this paper we use a pre-indexed system in which the audio to be searched is indexed without prior knowledge of the query terms. This approach is beneficial when large amounts of audio are to be searched interactively. The same index is used for both in-vocabulary (IV) and out-of-vocabulary (OOV) queries; the only difference between IV and OOV searches is the degree of query expansion.

In Section 2 we describe the evaluation data and metrics. In Section 3 we describe our WFST-based indexing and search system, as well as various system combination strategies. The ASR systems used for indexing are described in Section 4. In Section 5 we describe the system used in the 2012 DARPA RATS evaluation, as well as a number of alternate architectures. We conclude in Section 6.

## 2. DATA AND METRICS

All the training and test sets for the DARPA RATS program are provided by the LDC (Linguistic Data Consortium) [7, 8]. Much of the clean speech data is existing Fisher Levantine data that was repurposed for the RATS program. To introduce signal degradation, the clean speech was transmitted through eight different radio channels, labeled A–H and corresponding to different transmitter/receiver pairs, and then recorded. The clean data contained about 150 h of audio, but only about 65 h was labeled as speech. In the end, for acoustic model training, we defined a 251 h noisy speech (N) training set and a 310 h noisy plus clean (N+C) set.

To train the language model, we used only the transcripts corresponding to the 65 h of clean speech (about 500K words) because the transcripts for the other channels are exactly the same.

The probability of miss (pMiss) is defined to be $\frac{\sum_i \#\text{times keyword } i \text{ is missed}}{\sum_i \#\text{occurrences of keyword } i}$. The probability of false alarm (pFA) is defined to be $\frac{\#\text{false alarms}}{\#\text{total words} \times \#\text{keywords}}$. In the Phase 1 DARPA RATS evaluation, the goal is to minimize pFA at an operating point of 30% pMiss; this metric is denoted pFA@30%pMiss.

We will report results on two sets `dev-1` and `dev-2`. `dev-1` is the development data consisting of 219 keywords to search in 2.4 h of audio. `dev-2` is the evaluation test data and consists of 200 words to search in 34.2 h of audio.

## 3. KEYWORD SEARCH SYSTEM OVERVIEW

In this section we describe our overall keyword search framework, which runs many STD systems in parallel and combines the results to produce a ranked list of term occurrences. Each STD component follows the transcribe-and-match strategy in which the audio is processed by an ASR system that produces a word lattice for each audio segment. The lattices are converted to a WFST-based index [9, 10, 11] that is used to find the location and score of a detection. The STD components differ only in their ASR models and keyword pre-processing methods.

### 3.1. Lattice Pre-processing

Prior to indexing and search, the word lattices are converted into phonetic WFSTs in which the input labels are phones, the output labels are the starting times of phones, and the costs are phone negative log posteriors. The resulting utterance WFSTs are used (1) as

the starting point for creating the index, and (2) for retrieving the time marks of the hits during the search phase.

### 3.2. Indexing

The algorithm for converting the utterance WFSTs obtained in the previous step to a WFST index is described in [9]. The steps are as follows: (1) The time information is eliminated from the utterance WFSTs. (2) For each utterance WFST, a new start node $S$ is created and connected to all the original nodes $n$. The weight of the arc from $S$ to $n$ is the posterior probability of reaching node $n$ from the original start node $s$ of the WFST (i.e. shortest distance in log semiring between $s$ and $n$). The input and output labels of the arc from $S$ to $n$ are both $epsilon$. (3) For each utterance WFST, a new end node $E$ is created and all the original nodes $n$ are connected to it. The weight of the arc from $n$ to $E$ is the posterior probability of reaching the end node $e$ of the original WFST from $n$ (i.e. shortest distance in log semiring between $n$ and $e$). The arc from $n$ to $E$ has as input label $epsilon$ and output label the lattice id. The index WFST is the union of all the utterance WFSTs.

### 3.3. Search

In WFST-based keyword search, the query WFSA is constructed using the pronunciation dictionary for IV queries and a letter-to-sound model for OOV queries. Multiple pronunciations are compactly represented in the WFSA. In our Levantine Arabic system, the pronunciations are grapheme based: the pronunciation of a word is its letter sequence. Therefore, the pronunciations for both IV and OOV words can both be generated in the same way, which we represent as letter to sound transducer $L2S$. A fuzzier search, which may improve recall while degrading precision, can be accomplished using query expansion. Specifically, we estimate the probabilities of phone-to-phone confusions and create a confusability model implemented as the phone-to-phone transducer $P2P$. Given a query $q$, the query WFSA $Q$ is obtained via composition:

$$Q = \text{nbest}(q \circ L2S \circ P2P) \quad (1)$$

Varying the number of hypotheses kept after the composition ($NbestP2P$) controls the degree of query expansion, trading off between precision and recall.

#### *3.3.1. Training a Phone Confusability Transducer*

The phone confusability transducer should model the behavior of the ASR system when it is used to index new audio data. Thus, it is important to collect the confusability statistics on data that was not used for ASR training. For this work, we used two 10-hour subsets of the clean (transmitted) training audio, training separate models on each 10-hour subset, and then collecting statistics for each model on the other 10-hour subset. The acoustic model was a deep neural network model taking 13 frames of 40-d PLP+LDA+STC features as input, containing four hidden layers of 512 hyperbolic tangent units each, and estimating posterior probabilities for 144 context-independent HMM state targets. When we collected statistics for the confusability model, we collected counts for each frame in the test data. The reference labels were computed using forced alignment. The hypothesized labels were computed by running speech recognition followed by forced alignment.

#### *3.3.2. Producing the final hit list*

We use a 2-step search [12, 11] in which we first find the lattices containing the query through composition of the query WFST $Q$ with the index and then use the relevant utterance WFSTs from Section 3.1 to obtain the start and end time information for the hits. [10] proposes a 1-pass retrieval which improves the search time at the cost of a larger index, but we decided to use the 2-pass approach.

### 3.4. STD system combination

After producing a list of hits and the associated scores (posterior probabilities) for each STD branch, the results are merged. We explore two different methods: (1) linear combination of the scores followed by a normalization step, which is the method used in the RATS evaluation; and, (2) classifier-based combination, which was developed after the evaluation and found to produce significantly better results.

#### *3.4.1. Basic method*

First, we describe the fusion method used in the RATS evaluation. For each keyword, we take the union of all hits from all systems and then produce a final list using the following procedure: (1) a hit which does not overlap with any other hit is copied to the final list, while (2) a set of overlapping hits corresponding to the same keyword is merged into one hit in the final list which has the time marks of the highest scoring hit and a score that is the sum of the hit scores. After producing the new list of hits, we normalize the scores per keyword: for each keyword we sum all the scores for all the hits and divide each score by this sum. A variant of this normalization scheme (in which the min was shifted to 0) was proposed for IR data fusion in [13]. This is the first time this method is used for STD and shown to produce significant improvements.

#### *3.4.2. Maximum Entropy based system combination*

After the evaluation we implemented classifier-based system combination. The classifier is a conditional, maximum-entropy model. The input to the MaxEnt classifier is the merged hit list with the associated total scores from the previous section. For each hit in this list we consider three types of features: (1) keyword specific features, namely the number of phones and number of words; (2) system specific features, namely the system score and rank of the score among all the hits for a specific keyword; and (3) general features, namely the number of systems voting for a hit, the rank of the duration across all hits, and the total score after combining using the basic method. Except for the system scores and the total score, all the other features have discrete values. We discretize the continuous values features by binning into $k$ equal bins, where $k$ is optimized. Other binning methods had similar performance. Based on these features, the MaxEnt classifier produces a new score for each hit in the merged list. The final score is computed by multiplying the maximum-entropy score with the original total score.

## 4. DIVERSITY FOR INDEX BUILDING

Our approach to keyword search is to produce hit lists using different indexes and combine the results. We achieve index diversity in four ways: (1) acoustic model type, which can either be a Gaussian mixture model (GMM) or deep neural network (DNN); (2) decoding algorithm, which can be either dynamic or static; (3) audio segmentation technique; and (4) training data set, which can either use only

| Channel | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| GMM-U | 54.1 | 57.8 | 56.6 | 63.3 | 70.3 | 60.4 | 60.5 | 64.9 |
| GMM-V | 57.4 | 59.5 | 59.6 | 66.7 | 73.0 | 61.2 | 63.1 | 67.5 |
| DNN | 55.6 | 71.7 | 63.4 | 61.0 | 80.5 | 65.2 | 50.7 | 76.2 |

**Table 1**. Word error rates for GMM-U, GMM-V and DNN acoustic models on `dev-1`.

noisy data (N), or combine the noisy and clean data (N+C). All indexes are produced using the same language model, a 3-gram model with modified Kneser-Ney smoothing [14], and a 37K vocabulary.

### 4.1. Acoustic Models

#### *4.1.1. Gaussian Mixture Models (GMM) U and V*

Similar to our GALE system [15], we built two GMM acoustic models: a conventional unvowelized model (GMM-U) and a Buckwalter vowelized model (GMM-V). We added the vowelized model as a source of system diversification for keyword search. The frontend features for both models are based on VTL-warped PLP features and a context window of 9 frames. We apply speaker based cepstral mean and variance normalization, followed by an LDA transform to reduce the feature dimensionality to 40. The ML training of the acoustic model is interleaved with estimation of a global semi-tied covariance (STC) transform. FMLLR speaker adaptation is applied both in training and testing, while MLLR regression trees are applied only during run-time. The total number of Gaussian components is $120,000$, distributed over 7000 quinphone context-dependent states. Feature- and model-level discriminative training uses the boosted MMI (bMMI) [16] criterion. The first two rows in Table 1 show the word error rates (WER) on `dev-1` for our discriminatively-trained GMM systems.

#### *4.1.2. Deep Neural Network (DNN) Model*

We developed a deep neural network (DNN) acoustic model for the RATS Levantine Arabic keyword search task. The DNN acoustic model uses a feature processing pipeline similar to that used for the GMM-U and GMM-V acoustic models, with the primary differences being a Mel filterbank that only passes frequencies between 250–3000 Hz and no VTLN. The LDA+STC transform and speaker-dependent FMLLR transforms are inherited from an auxiliary GMM acoustic model that is independent of the GMM-U and GMM-V models. The DNN takes 13 consecutive frames of 40-d PLP+LDA+STC+FMLLR features as input, contains four hidden layers of 512 hyperbolic tangent units each, and estimates posterior probabilities for 4096 quingrapheme context-dependent HMM states. It therefore contains 3.2M trainable parameters.

The DNN model is initialized with random weights, using the normalized pinitialization proposed in [17]. Following initialization, the DNN is trained using stochastic gradient descent with a cross-entropy (CE) loss function. Following cross-entropy training, the DNN is trained in a sequence-discriminative fashion, using the state-level minimum Bayes risk (sMBR) criterion and a distributed implementation [18] of Hessian-free training [19]. The WER of this system is shown in the last row of Table 1.

### 4.2. Decoding Strategies

We used two different decoders and lattice generation strategies. The first decoder (static) is based on a fully precompiled search network that is heavily optimized at "compile" time. The decoder generates lattices using a small LM and then rescores them with the full LM. Lattice generation is done by propagating multiple search tokens corresponding to different LM histories during the forward pass. At merge points, only the top $N$ tokens (or backpointers) are kept.

In the second decoder (dynamic) the language model is applied dynamically. The search network, representing the search vocabulary, is precompiled at the HMM level, with sharing of common prefixes and suffixes. Lattice generation for the dynamic decoder is a conversion of the backpointer table to a lattice. We create *extra* arcs between matching backpointers (those with the same start and end times and cross-word context). Two criteria are used to limit the number of arcs: beam pruning based on the arc posterior and rank pruning that limits the number outgoing arcs per lattice state. This approach produces very rich lattices with minimal overhead (less than 10%) over regular Viterbi search. More decoder details can be found in [20].

### 4.3. Audio Segmentations

The first audio segmentation variant (S1) is a combination of multiple audio segmentations, all of which are based on Viterbi decoding with 3 states corresponding to speech, silence and no-transmission. These segmentations differ in their acoustic models and features. The first approach uses channel-dependent GMMs and neural networks trained on a 40-dimensional LDA feature space obtained by projecting consecutive PLP cepstra within a time window of $\pm 4$ frames. Both GMMs and neural networks are estimated with BMMI using an asymmetric loss that only penalizes false alarms. During segmentation, the scores from the GMMs and the neural networks are log-linearly combined at the frame level. Similar to [21], channel detection is performed by selecting the channel with the highest likelihood after decoding with a set of 8 GMMs trained with maximum likelihood. The second segmentation variant (S2) is based on the same Viterbi decoding, but uses 2 states corresponding to speech and non-speech. Another difference is that the segmentation models are derived from a full-resolution channel- and speaker-independent acoustic model via k-means clustering. $512$ Gaussians are used for the speech model and 12 for the non-speech model. The third segmentation (S3) is nearly identical to S2, save for the tuning strategy.

## 5. EXPERIMENTAL SETUP

In this section we describe two systems: 1) the system we deployed in the RATS 2012 evaluation with some additional post-eval improvements and 2) a simpler system more appropriate for an operational scenario.

### 5.1. Evaluation System

In the evaluation we combined the 5 systems shown in Table 2. The first 4 columns refer to the dimensions of variability described in the previous section. The last column shows the value of $NbestP2P$ that was used for in-vocabulary keyword search. For OOV queries, all systems use $NbestP2P = 10000$. As mentioned in Section 2, the evaluation metric is pFA@30%pMiss. After combining systems using the basic method and thresholding the results to achieve pMiss=30%, the result was pFA=0.07 on `dev-1` and pFA=0.22 on

| System | AM | Segmentation | Decoding | Data | $NbestP2P$ |
|---|---|---|---|---|---|
| Sys1 | GMM-V | S1 | dynamic | N | 50 |
| Sys2 | DNN | S1 | dynamic | N | 1 |
| Sys3 | GMM-V | S2 | static | N | 50 |
| Sys4 | GMM-U | S2 | dynamic | N+C | 1 |
| Sys5 | GMM-U | S3 | static | N+C | 1 |

**Table 2**. The 5 systems used in the RATS evaluation

| dev-1 | A | B | C | E | F | G | H | ALL |
|---|---|---|---|---|---|---|---|---|
| pMiss | 18.8 | 32.4 | 41.9 | 39.6 | 25.3 | 30.3 | 22.8 | 30.0 |
| pFA | 0.07 | 0.07 | 0.06 | 0.08 | 0.07 | 0.06 | 0.07 | **0.07** |
| dev-2 | A | B | C | E | F | G | H | ALL |
| pMiss | 30.0 | 43.6 | 40.9 | 29.3 | 19.0 | 10.9 | 50.6 | 30.0 |
| pFA | 0.32 | 0.29 | 0.18 | 0.31 | 0.19 | 0.14 | 0.18 | **0.22** |

**Table 3**. Per channel performance of the 5-way STD system on dev-1 and dev-2 using the basic combination method.

dev-2. Table 3 gives more details of the performance per channel at this operating point. If we do not normalize the combined scores, the resulting system is significantly worse: pFA=0.18 on dev-1 and pFA=0.45 on dev-2.

| | Sys1 | Sys2 | Sys3 | Sys4 | Sys5 | ALL |
|---|---|---|---|---|---|---|
| dev-1 | 0.38 | 0.28 | 0.34 | 0.24 | 0.26 | 0.07 |
| dev-2 | 0.88 | 0.80 | 0.85 | 0.71 | 0.74 | 0.22 |

**Table 4**. pFa@30%pMiss for the system components and the final combined system.

The lattices used in the 5 systems have been heavily pruned and are ten times smaller than what we usually use for speech recognition. We discovered that this design decision not only improved the performance of the combined system, but also significantly reduced the size of the final index and consequently the retrieval time. But due to the small lattice size, the probability of Miss is increased and none of the systems reaches pMiss=30% even when keeping all the hits. To better quantify the improvement over the single system components, we measured the individual performances without heavy pruning. Table 4 shows that the combined system is 3 times better than the best component on both dev-1 and dev-2.

The MaxEnt system combination requires a training set and a heldout set for optimizing $k$, the number of bins. Given that dev-2 is the evaluation data, we used dev-1 for training. We initially split dev-1 into 70% training and 30% heldout, but after finding the optimal value of $k = 10$, we used all of dev-1 for training the MaxEnt classifier. The new combination method results in pFA@30%pMiss = 0.18 on dev-2 which is 20% better than the baseline evaluation system.

#### *5.1.1. Simpler Architecture*

Having in mind an operational scenario, we investigated simpler alternatives to the evaluation system. Table 5 shows a 2-way combination, in which only the acoustic models and search strategies are different. Due to the fact that a 2-way combination would produce significantly fewer hits, if we want to be able to produce a full ROC curve we can either enlarge the lattices or use a higher $NbestP2P$. We explored both alternatives and found that it is better to increase $NbestP2P$ from both a computational point of view and performance-wise. Large lattices produce big indexes which are time consuming to produce and search. By increasing the number of alternate paths we keep in the query WFSA we insure more hits with no cost for indexing and minimal computational cost for retrieval. Empirically we found $NbestP2P$=1000 to be good for both IV and OOV words. Table 6 shows the comparison between the evaluation system and the simpler one. Consider packaging the whole system in a 12-core machine: it would be better to use the simpler system, which takes 25xRT for indexing, versus 100xRT for the 5-system combination. Note that search is not the bottleneck, and its latency for even the 5-system combination is small.

| System | AM | Segmentation | Decoding | Data | $NbestP2P$ |
|---|---|---|---|---|---|
| Sys1 | GMM-U | S1 | static | N + C | 1000 |
| Sys2 | DNN | S1 | dynamic | N | 1000 |

**Table 5**. Simpler Architecture

| | Eval System | Simplified System |
|---|---|---|
| dev-1 | 0.07 | 0.11 |
| dev-2 | 0.22 | 0.34 |

**Table 6**. pFa@30%pMiss comparison between the evaluation system and the simpler 2-way combination.

## 6. CONCLUSION

In this paper we present a state-of-the-art STD system which benefits substantially from diversifying the indexes and combining the results. The key messages we want to highlight are:

1. Diversifying acoustic models, search strategies, audio segmentations produces very different results for each STD component, generating significant improvements after combination. This approach works on tasks other than RATS: a similar methodology has been applied successfully in the context of the IARPA BABEL program [22] for a Cantonese STD task.
2. The fact that we combine many STD systems allows us to produce much smaller indexes, resulting in a faster retrieval.
3. Combining the results of different STD systems is much more beneficial than combining different acoustic models to produce a single STD output.

The final combined system is 3 times better than the best component, and additional improvements can be obtained by replacing linear combination with classifier-based combination. In the future, we will explore additional features and sample weighting schemes for the classifier-based system combination.

## 7. ACKNOWLEDGMENT

We thank Abhinav Sethy and Bhuvana Ramabhadran for useful discussions. This work was supported by the DARPA RATS Program.[1]

[1]The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views

## 8. REFERENCES


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

[2] B. Zhang, R. Schwartz, S. Tsakalidis, L. Nguyen, and S. Matsoukas, "White listing and score normalization for keyword spotting of noisy speech," in *Proc. Interspeech*, 2012.

[3] I. Bulyko, O. Kimball, M.-H. Siu, J. Herrero, and D. Blum, "Detection of unseen words in conversational mandarin," in *Proc. ICASSP*, 2012.

[4] D. Vergyri, I. Shafran, A. Stolcke, R. Gadde, M. Akbacak, B. Roark, and W. Wang, "The sri/ogi 2006 spoken term detection system," in *Proc. Interspeech*, 2007, pp. 2393–2396.

[5] D. Miller, M. Kleber, C. Kao, O. Kimball, T. Colthurst, S. Lowe, R. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Proc. Interspeech*, 2007.

[6] I. Szoke, L. Burget, J. Cernock, and M. Fapo, "Sub-word modeling of out of vocabulary words in spoken term detection," in *Proc. IEEE Workshop on Spoken Language Technology*, 2008.

[7] M. Maamouri er al., "Ldc2006s29, arabic cts levantine qt training data set 5," in *Linguistic Data Consortium*, 2006.

[8] D. Graff, S. Sessa, S. Strassel, and K. Walker, "Rats data plan," in *Linguistic Data Consortium, Tech. Rep.*, 2011.

[9] M. Mohri C. Allauzen and M. Saraclar, "General indexation of weighted automata - application to spoken utterance retrieval," in *Proc. HLT/NAACL*, 2004, vol. I., pp. 33–40.

[10] D. Can, E. Cooper, A. Sethy, C. White, B. Ramabhadran, and M. Saraclar, "Effect of pronunciation on oov queries for spoken term detection," in *Proc. ICASSP*, 2009, pp. 3957–3960.

[11] C. Parada, A. Sethy, and B. Ramabhadran, "Query-by-example spoken term detection for oov terms," in *Proc. ASRU*, 2009, pp. 404–409.

[12] S. Parlak and M. Saraclar, "Spoken term detection for turkish broadcast news," in *Proc. ICASSP*, 2008.

[13] M. Montague and J.A. Aslam, "Relevance score normalization for metasearch," in *Proceedings of the tenth international conference on Information and knowledge management*, 2001, pp. 427–433.

[14] S. F. Chen and J. T. Goodman, "An empirical study of smoothing techniques for language modeling," Tech. Rep. TR-10-98, Harvard University, 1998.

[15] H. Soltau, G. Saon, B. Kingsbury, H.-K. J. Kuo, L. Mangu, D. Povey, and A. Emami, "Advances in Arabic speech transcription at IBM under the DARPA GALE program," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 5, pp. 884–894, 2009.

[16] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *Proc. ICASSP*, 2008, vol. II., pp. 4057–4060.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.

[18] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *Proc. Interspeech*, 2012.

[19] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. Intl. Conf. on Machine Learning (ICML)*, 2010.

[20] H. Soltau and G. Saon, "Dynamic network decoding revisited," in *Proc. ASRU*, 2009, pp. 276–281.

[21] T. Ng, B. Zhang, L. Nguyen, S. Matsoukas, X. Zhou, N. Mesgarani, K. Vesely, and P. Matejka, "Developing speech activity detection system for the darpa rats program," in *Proc. Interspeech*, 2012.

[22] B. Kingsbury, J. Cui, X. Cui, M. J. F. Gales, K. Knill, J. Mamou, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, R. Schlüter, A. Sethy, and P. C. Woodland, "A high-performance Cantonese keyword search system," in *Proc. ICASSP*, Submitted for publication.



or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Approved for Public Release, Distribution Unlimited.