SCORE NORMALIZATION AND SYSTEM COMBINATION FOR IMPROVED KEYWORD SPOTTING

Damianos Karakos[1] *Richard Schwartz*[1] *Stavros Tsakalidis*[1] *Le Zhang*[1] *Shivesh Ranjan*[1]
Tim Ng[1] *Roger Hsiao*[1] *Guruprasad Saikumar*[1] *Ivan Bulyko*[1] *Long Nguyen*[1] *John Makhoul*[1]
Frantisek Grezl[2] *Mirko Hannemann*[2] *Martin Karafiat*[2] *Igor Szoke*[2] *Karel Vesely*[2]
Lori Lamel[3] *Viet-Bac Le*[4]

[1] Raytheon BBN Technologies, Cambridge, MA, USA
[2] SpeechFIT, Brno University of Technology, Brno, Czech Republic
[3] CNRS-LIMSI, France
[4] Vocapia Research, France

email: [1] {dkarakos,schwartz,stavros,lzhang,sranjan,tng,whsiao,gsaikuma,ln,makhoul}@bbn.com
[2] {grezl,ihannema,karafiat,szoke,iveselyk}@fit.vutbr.cz
[3] lamel@limsi.fr [4] levb@vocapia.com

ABSTRACT

We present two techniques that are shown to yield improved Keyword Spotting (KWS) performance when using the ATWV/MTWV performance measures: (i) score normalization, where the scores of different keywords become commensurate with each other and they more closely correspond to the probability of being correct than raw posteriors; and (ii) system combination, where the detections of multiple systems are merged together, and their scores are interpolated with weights which are optimized using MTWV as the maximization criterion. Both score normalization and system combination approaches show that significant gains in ATWV/MTWV can be obtained, sometimes on the order of 8-10 points (absolute), in five different languages. A variant of these methods resulted in the highest performance for the official surprise language evaluation for the IARPA-funded Babel project in April 2013.

Index Terms— keyword search, score normalization, system combination, indexing and search

1. INTRODUCTION

The task of finding words or phrases in audio is related to, but still quite different from that of speech recognition, where a verbatim transcript is desired. However, while word-error-rate has been viewed as a very appropriate measure for speech recognition performance, a number of alternative measures have been proposed for keyword spotting. In this paper, we consider the Actual Term Weighted Value (ATWV) [1] and the Maximum Term Weighted Value (MTWV) as the measures of interest, mainly because they are the official measures for the IARPA-funded Babel project. ATWV is a weighted average of the miss and false alarm probabilities over the collection of keywords, and, as defined by the Babel project, it requires that all of the keywords have commensurate scores, a *single* decision threshold is used for all keywords.

The traditional method for computing word confidence uses the measured posterior probability (e.g., from a word lattice) combined with several other features of the word. However, we find that the confidences for a particular keyword tend to be biased higher or lower even though we use other features. If we had a large number of samples of each keyword in a tuning set, we could learn a separate confidence model for each keyword. But, in this data, few keywords have more than 1 or 2 samples in the tuning set, and half have none at all. So we need methods for normalizing the posterior scores that do not require a large number of samples of each keyword.

In this paper we demonstrate that one can calibrate the scores so that all keywords have scores that are comparable to each other. Building on prior work [2], which showed the benefits of mapping the raw scores to probability of false alarm, we present a score normalization technique based on a learning framework that aims to estimate the probability of correctness of each keyword. Ranking the keyword detections based on this new score yields significant improvements in the ATWV/MTWV performance across several languages. Note that our proposed technique is very different from the one proposed in [3], which modifes the scores based on an estimate of the actual log-likelihood ratio, as well as previous techniques (e.g., [4, 5]).

We also present results with the combination of different systems (or different modalities of the same system) which improve ATWV. We also show that the best performance is obtained by normalizing the system outputs *before* doing system combination. The intuitive explanation for this observation is that the system-specific biases (which are not just a simple shift of the scores, but the result of a rather complicated process) act as nuisance parameters in the downstream objective of combining the sets of hits and re-ordering them in order to maximize ATWV/MTWV. Thus, removing (or reducing) these extraneous biases at an earlier stage makes it easier to focus on the final optimization problem. (Of course, a more complicated approach would be to perform score normalization and system combination *jointly*; we leave this for future research.)

The paper is organized as follows: Section 2 contains the task definition. In Section 3 we list a number of techniques that can be used for improving keyword spotting. Section 4 describes the KWS system we implemented, that makes use of score normalization and machine learning. Section 5 presents the methodology for combining outputs from different systems (or different modalities of the same system). Section 6 presents the experimental framework and KWS results on 5 languages (Cantonese, Pashto, Tagalog, Turkish, Vietnamese). Section 7 contains conclusions.

2. TASK DEFINITION

The spoken term detection system has four components: a speech-to-text engine, an indexer, a detector, and a module that re-ranks detections according to various features and assigns a score to each. The speech-to-text engine processes audio files and outputs word lattices and single-best phonetic transcripts. The system uses word lattices instead of single-best transcripts in order to avoid the problem of missing keyword occurrences that are not in the single best transcription. It estimates the posterior probability of each detection's correctness directly from the lattices (or confusion networks). The indexer creates an index containing a precomputed list of candidate detection records (a.k.a. hits) for each word in the speech-to-text lexicon. The index also contains the phonetic transcripts to accommodate out-of-vocabulary search terms. The detector processes a list of search terms (which can be single words or multi-word strings), generating a sorted, scored list of detection records for each term. Next, the re-ranker re-assigns scores based on a learned model, using various features as input. Finally, a decision function assigns a threshold and all detections with scores above the threshold receive a Yes decision.

Accuracy is judged relative to a time-marked reference transcript. A system detection is considered correct if a corresponding exact orthographic match of the term appears in the reference transcript within 0.5 seconds of the asserted time.

System accuracy on a given collection of query terms is measured by the Actual Term-Weighted Value (ATWV) metric, defined in [1] as

$$\text{ATWV} = 1 - \frac{1}{K}\sum_{w=1}^{K}\left(\frac{\#\textit{miss}(w)}{\#\textit{ref}(w)} + \beta\,\frac{\#\textit{fa}(w)}{T - \#\textit{ref}(w)}\right) \tag{1}$$

where K is the total number of keywords, $\#\textit{miss}(w)$ is the number of true tokens of keyword w that are not detected, $\#\textit{fa}(w)$ is the number of false detections of w, $\#\textit{ref}(w)$ is the number of reference tokens of w, T is the total number of trials (e.g., seconds in the audio), and β is a constant, set at 999.9[1].

Note that ATWV is a function of the threshold used in deciding whether a detection exists or not. The Maximum Term-Weighted Value (MTWV) is then defined as the maximum ATWV over all decision thresholds.

3. TECHNIQUES FOR SCORE NORMALIZATION

Here we summarize some normalization techniques that are most relevant to our setting.

Similar to [2], we assume that the keywords are given in advance of any training/decoding of speech. This allows us to perform a more thorough job of detecting the keywords, by (i) adding the keywords to the decoding dictionary and learning language models that give a higher probability to n-grams that contain keywords; (ii) tailoring the pruning beams of the decoder so that it does not miss many occurrences of keywords (whitelisting [2]). The techniques discussed here work equally well in the more traditional setting where the keywords are not known until after decoding has been performed. The models and parameters for normalization and system combination are learned from a transcribed development set.

3.1. Keyword-specific Thresholding and Exponential Normalization (KST)

In [6], a formula is presented for computing a decision (Yes/No) for each detection. This "decider" function declares that a detection for keyword w is present if its posterior score is above threshold

$$\textit{thr}(w) = \frac{N_{\text{true}}}{T/\beta + \frac{\beta-1}{\beta}N_{\text{true}}} \tag{2}$$

where N_{true} is an estimate of how many true tokens of keyword w exist in the audio. In the absence of true transcripts, N_{true} can be approximated by the expected count for that keyword, with an additional correction (regression) learned from training data. Note that this allows one to deal with untranscribed audio seamlessly; the threshold can be decided in an unsupervised fashion, as long as the expected counts are reasonably estimated.

If the scores are true posterior probabilities, and assuming that N_{true} is known[2] this decision rule can be shown to be optimal[3] when the ATWV metric is used. Basically, while the cost of false alarms is the same for all keywords, the cost of misses is lower for keywords with more true occurrences.

The keyword-specific thresholds are used in this paper for normalizing the scores across all keywords in such a way that the decision threshold becomes a constant $\textit{thr} \in (0,1)$. Specifically, the formula for transforming the score s of a keyword w has the following exponential form [7]

$$s' = s^{\left(\frac{\log(\textit{thr})}{\log(\textit{thr}(w))}\right)}. \tag{3}$$

Setting $\textit{thr} = 1/e \approx 0.3679$ simplifies the formula to

$$s' = s^{\left(-\frac{1}{\log(\textit{thr}(w))}\right)}, \tag{4}$$

which is used in the rest of the paper.

The motivation behind this formula is that scores of keywords with generally low scores (low $\textit{thr}(w)$) get a boost, if $\textit{thr}(w)$ is below the global threshold $\textit{thr}$. Conversely, keywords with generally higher scores get attenuated, so that, in the end, all keywords have the same global threshold. The non-linearity of the formula is useful in making the scores more distinguishable in the regime of interest.

3.2. Rank Normalization and Mapping Back to Posteriors

In [2], it was shown that significant gains in performance can be achieved by transforming the raw posteriors through the so-called "pFA mapping". This entails computing the false alarm rate (pFA) corresponding to each raw score, on a per-keyword basis, and then using 1-pFA as the new (normalized) score. The false alarm rate is computed by sorting all scores (corresponding to the false alarms of a keyword, obtained through an alignment of the detections and the true transcripts on a development set) and then assigning a normalized rank to each hit, starting with the largest posterior values.

[1]This was set equal to $\frac{C}{V}(\Pr_{kw}^{-1} - 1)$, where $C = 0.1$ is the cost of a false detection, $V = 1$ is the value of a correct detection, and $\Pr_{kw}$ is the prior probability of a keyword, which is fixed at 10^{-4}.

[2]In the case where N_{true} is random and cannot be reliably estimated, one can try to find the decision rule that minimizes the expected Bayes risk.

[3]Theoretical optimality is established when all distributions are known in advance. When there are biases in the data, even using the true number of references does not necessarily lead to optimum performance; a modification of the decision threshold that corresponds to these biases may be preferable.

Note that this does not require having any true tokens of the keywords in the development set. This allows to learn keyword-specific mappings[4]. In the case where there are very few false alarms for a keyword (less than 5) a "global" map (described below) is used instead.

In order to use formula (2), one has to have scores which resemble posteriors. To do that, we define a global map as the *average* posterior value at each rank. Then, after a keyword-specific map transforms a posterior to a rank, the rank is mapped back to a posterior-resembling value using this averaged map. That is, for keyword w, the mapping function is

$$f_w(x) = \frac{1}{K} \sum_{v=1}^{K} R_v^{-1}(R_w(x)) \tag{5}$$

where K is the number of keywords, $R_k(x)$ is the keyword specific map for keyword k (maps a posterior to a rank) and $R_k^{-1}(r)$ is its inverse function (maps a rank to a posterior). For instance, if a hit in a keyword-specific list is ranked 5th, then its normalized score is set equal to the average of the posteriors over all hits which are ranked 5th in their respective keyword-specific lists. Note that these maps are used with linear interpolation in both directions.

3.3. Sum-to-One Normalization (STO)

A technique for score normalization, sum-to-one, appeared in [8]. It was motivated by a similar technique from Information Retrieval (IR) [9]. Simply put, given a query/keyword w with scores $s_{w,1}, \ldots, s_{w,n}$, the normalized scores become

$$s'_{w,i} = \frac{s_{w,i}}{\sum_{j=1}^{n} s_{w,j}}, \tag{6}$$

Similar to the exponential normalization above, this is motivated by the desire to boost the scores of keywords with generally low scores (low denominator) and to give lower cost to misses for keywords with more true tokens.

Note that the global decision threshold has to be adjusted according to the ratio of the Dev and Test audio durations. This happens because the denominator in (6) grows linearly with the duration of the audio, while the numerator is bounded above by 1. This is less of a problem in the KST method, as N_{true} and T are linearly related, and therefore the coefficient of proportionality cancels out.

4. SCORE NORMALIZATION USING MACHINE LEARNING

In this section we outline the main steps that we follow to perform score normalization using a machine learning framework that uses several features.

4.1. Score Normalization Procedure

The score normalization procedure is performed through a series of steps:

1. The original scores of the detections go through a number of transformations, such as: rank-normalization, mapping-back to posteriors, "probability of correct" normalization $p_{corr}()$, and through some non-linear functions such as $\log()$, $()^{1/2}$, $()^2$, sigmoid, using equation (2), etc. The $p_{corr}()$ mapping is estimated by sorting all hits by score, defining bins, and then computing the probability that a random detection in the bin is correct, i.e.

$$p_{corr}(\text{bin}) = \frac{\text{\# true positives in bin}}{\text{size of bin}}.$$

   The mapping from score to p_{corr} is smoothed using linear interpolation during lookup.
2. For each detection, the transformed scores, together with various additional features, e.g., keyword training count, keyword length, conversation-aggregated scores, are concatenated together into a feature vector $\boldsymbol{f}$.
3. This vector, together with a target variable denoting whether the detection is a "true positive" or a "false alarm", is given as input to Powell's method [10], which learns a linear model using MTWV as the maximizing criterion. Powell's method learns a weight vector $\boldsymbol{w}$ for these features. The score assigned to a feature vector is then given by the inner product $s(\boldsymbol{f}) = \langle \boldsymbol{w}, \boldsymbol{f} \rangle$. This is subsequently converted to a number in $[0, 1]$ using another $p_{corr}()$ mapping.
4. (Optional) The p_{corr}-mapped values are given next as input to the threshold determination and normalization module of Section 3.1, and a per-keyword detection threshold *thr*(w) is then determined. This thus makes all keywords consistent with each other, as they all have exactly the same decision threshold $1/e$.
5. Further correction/calibration of the global threshold is (optionally) done by selecting a value that maximizes ATWV on some training data.

5. SYSTEM COMBINATION

The goal of system combination is to take as input the detections of several systems, and then come up with a new list of hits which has performance that exceeds that of any individual system. We present a procedure that combines several hit lists together.

System combination exploits the diversity among the different outputs. Even a single decoding run, when appropriately diversified through the generation of different modalities, can offer gains. Obviously, the biggest gains are obtained when the different systems are radically different (e.g., GMM-based and DNN-based HMMs).

5.1. Algorithm for System Combination

The algorithm for doing system combination consists of the following steps:

1. Perform score normalization, including $p_{corr}()$ mapping, of the scores of the different systems.
2. Rank the systems in terms of MTWV performance, and merge their hits incrementally (two at a time) into a new list. The merging is done so that if there is an overlap of at least 10% of the smallest of the two intervals corresponding to the two hits, the new hit has start/end times of the hit of the better of the two systems. If the overlap is smaller (or, there is no overlap) the two hits are added into the new list without modification. The above merging creates a feature vector per merged hit, with the scores of the systems being the elements of the vector. When there is no score for a system, a score of zero is assigned.
3. An initial set of weights is chosen, based on which the vector scores are linearly combined. The weights are given by the formula

$$w(s) = 2^{\text{MTWV}(s) - \text{MTWV}_{\text{best}}}, \tag{7}$$

[4]When transforming the training (tuning) data, a leave-one-out methodology is used, in order to avoid exact integer mapped values; this is helpful in the subsequent learning.

where $\text{MTWV}_{\text{best}}$ is the MTWV of the best system among those combined. Note that the exponent is expressed in terms of percent scores (e.g., 45, rather than 0.45).

4. The initial weights are further optimized using Powell's method using MTWV as the optimization criterion.
5. After the hits are re-ranked using the optimized weights, the new (weighted) scores are further conditioned using another $p_{corr}()$ mapping (trained on the Dev data), so that the resulting scores resemble probabilities (and can be used as input in another stage of system combination).

6. EXPERIMENTS

Here we list all experiments we have run on a variety of languages and conditions.

6.1. Audio corpora and keyword sets

The audio corpora and keyword sets that we considered in our research were provided by the IARPA Babel program (FullLP releases). The languages were Cantonese (release babel101b-v0.4c), Pashto (release babel104b-v0.4bY), Tagalog (release babel105b-v0.4), Turkish (release babel106b-v0.2g) and Vietnamese (release babel107b-v0.7). The condition we report in this paper is the so-called Automatically Adapted condition, where the keywords are supposed to be known in advance of the decoding of the audio, and their knowledge is used to improve the search. This should be contrasted with the more well-known form of keyword search where the audio is pre-indexed without knowledge of the keywords/queries. We also used these same methods for the more traditional Pre-Indexed condition and found similar results. In addition, we have experimented with a two-pass approach in which we use the Pre-Indexed approach to find likely keywords and then rescore only those hits with high scores.

The training data for each language were of the order of 100 hours, and the data on which we report performance are: (i) Dev set of each language, about 10 hours each, (ii) Test set of each language, with durations 5 hours (Cantonese, Pashto, Tagalog and Turkish) and 15 hours (Vietnamese). The test sets were supplied by NIST as "unsequestered" parts of the official evaluation sets used in the March/April 2013 Babel evaluations. The keyword sets on which we report results are the official lists provided by NIST for the evaluations; their sizes are 3762 for Cantonese, 3842 for Pashto, 3171 for Turkish, 3805 for Tagalog, and 4065 for Vietnamese.

6.2. System Descriptions

(i) BBN GMM System
The BBN Byblos system uses Hidden Markov Models (HMMs), with State-Clustered-Tied Mixture (SCTM) crossword quinphone models. The parameters for these models are estimated using the Minimum Phone Error (MPE) objective criterion. The acoustic features are based on a 6-layer stacked bottleneck neural network architecture [11].

Recognition is performed using the BBN two-pass decoder. The forward pass uses a State Tied Mixture (STM) model, and an approximate bigram LM to produce word-ending scores. The backward pass then uses the word-ending scores and associated scores from the forward pass to perform a detailed search using within-word state-clustered tied-mixture (SCTM) quinphone acoustic models and a trigram language LM to produce a lattice. Finally, lattice rescoring using a state clustered cross-word quinphone model is done.

(ii) BBN Deep Neural Network System
The BBN DNN system has a topology of 4 hidden layers, each with 2000 hidden units. The network uses the same set of clustered states from the HMM/GMM system for the output layer, and is trained using state-alignments derived from an MPE-trained HMM/GMM. Discriminative pre-training is used to initialize the network weights [12]. This entails starting with one randomly initialized hidden layer, and then pretraining it using back-propagation with the minimum cross-entropy criterion. The softmax output layer is subsequently replaced with another randomly initialized hidden layer, and the whole procedure is repeated until a desired number of hidden layers are trained. GPU machines are used to speed up the training of the DNN system.

For recognition, HMM decoding is first done with a speaker-independent model. This is followed by speaker-adaptive decoding and the DNN decoder is applied in the backward pass to produce lattice outputs, which are then fed into the keyword search system. The observation probability in the HMM is set equal to the ratio of DNN state posterior to its prior, as in [12].

(iii) Different Modalities of BBN Systems
Word lattices are converted to word and phone based confusion networks, as in [13, 14]. Additionally, character-based confusion networks are generated for Cantonese, and syllable-based confusion networks are generated for Vietnamese. The whole-word raw scores are the actual arc posteriors (both for lattices and confusion networks), whereas the phone based results are computed via the product and geometric mean of the scores of the individual phones that constitute the pronunciation of the keyword. Each one of these tokenizations produces a different set of hits that are normalized and combined together using the algorithms mentioned earlier.

(iv) BUT GMM System
The BUT STK HMM system uses cross-word tied-state triphones (approx. 2000 tied states). Feature extraction is done based on the concatenation of three feature streams: (i) PLP-HLDA (39 dimensions): HLDA transformation of a feature vector consisting of mean-and-variance-normalized cepstral coefficients, including delta, double and triple delta coefficients. (ii) Stacked Bottleneck Neural Network (30 dimensions) [15], which is a hierarchical composition of two Neural Networks. (iii) F0 with delta and double delta coefficients (3 dimensions); the implementation of F0 and probability of voicing estimation follows [16]. The concatenated feature vector (72 dimensions) is further processed by two different RDT transforms [17], used in a first-pass and a speaker-adapted decoding, respectively.

(v) BUT DNN System
An initial HMM/GMM system was used to produce SAT features, triphone alignments and the triphone-clustering tree. After splicing together 11 frames of fMLLR features (440 dims), an unsupervised pre-training of a DNN layer-wise is done by stacking 6 RBMs, using the contrastive-divergence optimization. The dimension of the hidden layers is 2048. The topology of the network was not tuned, but re-used a standard recipe tested on Switchboard.

After pre-training, the final layer of the DNN is added, which has $\approx$4.5k outputs (same as the number of GMM PDFs (tied states)), and the whole network is trained optimizing frame-level cross-entropy using mini-batch Stochastic Gradient Descent. This is followed by realignment, and a second iteration of cross-entropy training.

Finally, the DNN is re-trained optimizing the sMBR criterion, which is done by Stochastic Gradient Descent with per-utterance updates [18]. A development set is used to check which sMBR iteration gives the best WER.

The KWS is done based on exact match in the lattices, using the word representation of the terms. The posterior of a term is computed using a forward/backward procedure. In case of overlapping detections of the same term, the posteriors are summed together.

(vi) LIMSI-Vocapia Systems
The LIMSI-Vocapia speech-to-text (STT) systems for all languages make use of BBN voice activity detection and BUT features. The acoustic models are tied-state, left-to-right 3-state HMMs with Gaussian mixture observation densities (typically 32 components). The triphone-based phone models are word-independent, but position-dependent. The states are tied by means of a decision tree. For the Cantonese, Pashto, Tagalog and Vietnamese languages a 2-pass decoding was used with system combination (cross adaptation or Rover) of systems with different acoustic units (phone, initial-final, graphemic). A single-pass decoding was used for Turkish. Tone is explicitly represented in the phone sets for both the Cantonese and Vietnamese languages. Case sensitive language models were used for the Tagalog and Turkish languages.

The KWS systems are based on the multi-hypotheses produced by a consensus network (CN) decoding, using a character-based CN decoding for both Cantonese and Vietnamese. The KWS hits are ranked using a geometric mean score, prior to further normalization and calibration by BBN's system.

6.3. Score Normalization Results

Table 1 contains normalization results on the five languages for the BBN GMM system mentioned earlier (best tokenization per language). The column headings correspond to language abbreviations. Each of the methods discussed in Sections 3 and 4, corresponds to a different row. The row "Raw" shows the results obtained without any normalization whatsoever (raw posteriors). The row "ML" corresponds to the machine learning approach mentioned in Section 4.

As is clear from these results, all of the normalization methods improve significantly over the raw posteriors. In comparing the methods, (i) the sum-to-1 method is generally the weakest, (ii) the machine learning method is generally the best, (iii) the KST-norm method is almost as good as the best method (which is a bit surprising, given that it is very simple to implement and does not involve learning).

6.4. System Combination Results

We performed combination experiments with two different BBN systems (the GMM and DNN systems) mentioned above, to show the interplay between normalization and combination. We also present results with a wider set of systems used in the BABELON team in the March/April 2013 Babel project evaluations.

(i) Results with the BBN GMM and DNN systems
As mentioned above, we used several tokenizations/modalities for the two systems (GMM and DNN) under consideration. In all, we created at least 10 different sets of hits per language (with Cantonese having the most, as, because of its ideographic nature, it allows to use character-based matching).

	Ca	Pa	Tu	Ta	Vi
Raw	43.9%	38.1%	43.5%	45.0%	46.1%
STO	52.5%	45.2%	52.6%	52.1%	58.0%
Rank	52.7%	45.7%	53.4%	52.5%	58.9%
KST	53.1%	45.9%	53.7%	52.9%	58.9%
ML	**54.5%**	**46.4%**	**54.4%**	**53.3%**	**59.3%**

(a) MTWV Results on the Dev data for the HMM system.

	Ca	Pa	Tu	Ta	Vi
Raw	46.8%	35.9%	48.1%	43.5%	41.8%
STO	57.1%	42.7%	57.5%	53.3%	51.9%
Rank	55.8%	42.0%	57.6%	53.2%	52.3%
KST	57.3%	44.1%	**58.7%**	54.6%	52.9%
ML	**57.9%**	**44.3%**	58.5%	**55.0%**	**53.0%**

(b) ATWV Results on the Test data for the HMM system.

Table 1. Score normalization results. The best result in each column is shown in bold.

We compare two different pipelines for normalization and combination. (i) Pipeline 1 first performs combination of the modalities of the systems based on their raw scores. The resulting combined hit list is subsequently normalized with the machine learning method mentioned earlier. (ii) Pipeline 2 first performs normalization of the different modalities, and then combines the normalized lists of hits (and performs one more normalization at the end).

Table 2 shows the results obtained with the two pipelines (rows "P1", "P2"). We also implemented methods CombSUM, CombMNZ and WCombMNZ from [8], and show the results obtained with the best of the three methods, as determined by the MTWV on the Dev data (row "P2-c").

As can be easily seen, the procedure that first normalizes the different outputs and subsequently combines the resulting hits lists (pipeline 2) gives better Dev and Eval scores in all languages, even by as much as 2.9 points compared to pipeline 1. This can be explained by the fact that normalization outputs scores which are more commensurate across systems and can thus be combined more efficiently. Interestingly, in many cases the performance of pipeline 1 does not even reach the normalized performance of the best system (compare the "P1" rows with the corresponding best results of Table 1).
(ii) Results with the BABELON systems
Table 3 shows official results obtained with the BABELON systems used in the NIST Evaluation in March and April of 2013 (row "Best" shows the best system among those combined, and "P2" shows the system combination with the P2 pipeline). As mentioned earlier, the normalization/combination techniques presented benefit equally well the automatically-adapted ("-AA" rows) and the pre-indexed ("-PI" rows) conditions; however, the final ATWV is (as expected) better in the automatically-adapted condition. As can be easily seen, system combination achieves several points gain on top of the best normalized single system. The significant diversity between the different systems contributes greatly to the large gains.

7. CONCLUDING REMARKS

We have shown that a machine-learning framework that utilizes many features can result in improved KWS performance when com-

	Ca	Pa	Tu	Ta	Vi
P1	54.1%	46.4%	54.2%	54.0%	60.4%
P2	**57.1%**	**47.8%**	**55.6%**	**54.7%**	**62.9%**
P2-c	55.7%	47.0%	54.9%	53.8%	59.9%

(a) MTWV Results on the Dev data.

	Ca	Pa	Tu	Ta	Vi
P1	58.6%	43.5%	57.5%	54.9%	51.2%
P2	**59.6%**	**43.8%**	**58.4%**	55.5%	54.0%
P2-c	59.1%	43.3%	58.1%	**55.7%**	**55.1%**

(b) ATWV Results on the Test data.

Table 2. System combination results. P1 and P2 stand for Pipeline 1 and 2, respectively. The best result in each column is shown in bold.

	Ca	Pa	Tu	Ta	Vi
Best-PI	56.0%	43.1%	57.6%	50.8%	52.7%
P2-PI	**61.3%**	**46.9%**	**62.0%**	**57.5%**	**62.5%**
Best-AA	61.4%	45.6%	63.7%	57.0%	55.6%
P2-AA	**63.2%**	**49.2%**	**65.4%**	**60.4%**	**64.7%**

Table 3. BABELON system combination results using pipeline P2.

pared to using raw posteriors, or even using the "decider" formula (2)[5]. Our focus has been on improving ATWV/MTWV, and most of our innovations are designed specifically for these measures. We showed gains in both measures for the five languages used in the first year of the Babel program, namely, Cantonese, Pashto, Tagalog, Turkish and Vietnamese. Furthermore, a system combination framework that merges the detections from multiple systems and then uses Powell's method to learn how to weight the different scores results in significant ATWV/MTWV gains.

8. ACKNOWLEDGMENTS

We would like to thank all members of the BBN Speech and Language group for useful discussions, and especially those who work on the Babel project and contributed decoder outputs that were used in the score normalization results of this paper. We would also like to acknowledge the help and contribution of other partners of the BABELON team on the IARPA-funded Babel project. Furthermore, the comments of the anonymous reviewers are greatfully acknowledged.

This work 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.

[5]We have observed larger gains with the machine learning method, when applied on detection outputs whose scores are more approximate posteriors; we plan to report such results in a future publication.

9. REFERENCES

[1] "Openkws13 keyword search evaluation plan."

[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. of Interspeech*, Portland, OR, Sep 2012.

[3] G. Doddington, "The role of score calibration in speaker recognition," in *Proc. of Interspeech*, Portland, OR, Sep 2012.

[4] H. Jin, R. M. Schwartz, S. Sista, and F. Walls, "Topic tracking for radio, TV broadcast and newswire," in *Proc. of EUROSPEECH 1999*, Budapest, Hungary, Sep 1999.

[5] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. of SIGIR'07*, Amsterdam, The Netherlands, July 2007.

[6] 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 *Proc. of InterSpeech*, 2007.

[7] BBN Presentation at IARPA Babel PI Meeting, Sep 2012.

[8] J. Mamou, J. Cui, X. Cui, M. J. F. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, R. Schlüter, A. Sethy, and P. C. Woodland, "System combination and score normalization for spoken term detection," in *Proc. of ICASSP*, May 2013.

[9] S. Wu, *Data Fusion in Information Retrieval*. Springer Verlag, 2012.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The art of Scientific Computing*. Cambridge University Press, 2007.

[11] M. Karafiat, F. Grezl, M. Hannemann, K. Vesely, and H. Cernocky, "BUT Babel system for spontaneous Cantonese," in *Proc. of Interspeech*, Lyon, France, Aug 2013.

[12] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. of Interspeech*, Florence, Italy, Aug 2011.

[13] I. Bulyko, O. Kimball, M.-H. Siu, J. Herrero, and D. Blum, "Detection of unseen words in conversational Mandarin," in *Proceedings of ICASSP*, Kyoto, Japan, Mar 2012.

[14] I. Bulyko, J. Herrero, C. Mihelich, and O. Kimball, "Subword speech recognition for detection of unseen words," in *Proc. of Interspeech*, Portland, OR, Sep 2012.

[15] F. Grezl, M. Karafiat, and L. Burget, "Investigation into bottleneck features for meeting speech," in *Proc. Interspeech 2009*, Brighton, UK, 2009.

[16] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech Coding and Synthesis*, W. B. Kleijn and K. Paliwal, Eds. New York: Elsevier, 1995.

[17] B. Zhang, S. Matsoukas, and R. Schwartz, "Recent progress on the discriminative region-dependent transform for speech feature extraction," in *Proc. of Interspeech 2006*, Pittsburgh, PA, USA, Sep 2006, pp. 2977–2980.

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