A Comparison of Multiple Methods for Rescoring Keyword Search Lists for Low Resource Languages

Victor Soto¹, Lidia Mangu², Andrew Rosenberg³, Julia Hirschberg⁴

¹Department of Computer Science, Columbia University, New York, NY, USA
²IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
³Department of Computer Science, CUNY Graduate Center, New York, NY, USA
⁴IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
v.soto@cs.columbia.edu, mangu@us.ibm.com, andrew@cs.qc.cuny.edu, julia@cs.columbia.edu

Abstract

We review the performance of a new two-stage cascaded machine learning approach for rescoring keyword search output for low resource languages. In the first stage Confusion Networks (CNs) are rescoring for improved Automatic Speech Recognition (ASR) by reranking the arcs of each confusion bin. In the second stage we generate keyword search hypotheses from the rescored ASR output and rescore them using logistic regression classifiers to detect true hits and false alarms. We compare the performance of our system with state of the art rescoring techniques, including probability of false alarm normalization, exponential normalization, rank-normalized posterior scores and sum-to-one normalization and show promising results. Experimental validation is performed using the Term Weighted Value (TWV) metric on four corpora from the IARPA-Babel program for keyword search on low resource languages, including Assamese, Bengali, Lao and Zulu. Index Terms: keyword search, rescoring, normalization, low-resource languages.

1. Introduction

Keyword search in low resource languages is of much current interest to the information retrieval and analysis communities. The IARPA Babel program [1] focuses on keyword search in spoken low resource languages, given limited amounts of development time and data, where keyword search is defined as “find all of the occurrences of a ‘keyword’, a sequence of one or more words, in a corpus of un-segmented speech data, transcribed in a language’s original orthography.”[2] In this paper we present results of a two-stage cascaded approach to improving keyword search results. In Section 2 we describe the corpora we use. In Section 3 we describe prior research on rescoring. In Section 4 we describe our new approach, current experiments and their results. In Section 5 we discuss future directions.

2. The Corpora

The corpora provided by IARPA consist of conversational and scripted telephone speech, transcriptions for a fraction of the dataset, and a pronunciation lexicon. Each corpus contains conversations of two speakers recorded on separate channels for 10 minutes. The program defines two conditions of use: the full language pack, where all data resources provided can be used for training, and the limited language pack (LLP), where only a 10 hour subset of transcribed audio and a reduced pronunciation lexicon can be used for training, along with the remaining untranscribed data. All experiments presented here were performed on the LLPs of IARPA- {babel102b-v0.5a, babel103b-v0.4b, babel205b-v3.1a,babel206b-v0.1e} (Assamese, Bengali, Lao and Zulu respectively). The output of a keyword search run is a Posting List (PL), a list of the detected hits containing the following information: the beginning and duration of the hit, a posterior score, and a decision by the system as to whether the detection is correct. Decisions must conform to a global keyword-independent threshold $\theta$, so that all hits under the threshold will be set to “NO” and all hits over the threshold will be set to “YES”. The performance of the KWS system is measured using the term-weighted value function (TWV), defined as one minus the weighted sum of probability of missed occurrence and probability of false alarms (eq. 1). The maximum TWV is then defined as the maximum TWV over all possible thresholds $\theta$ (eq. 2).

$$TWV(\theta) = 1 - (P_{miss}(\theta) + \beta P_{FA}(\theta))$$ (1)

$$MTWV = \max_{\theta} TWV(\theta)$$ (2)

In TWV every keyword has the same weight for computing the probability of false alarms and missed detections, regardless its number of true occurrences in the corpus. The probability of false alarms and missed detection are then computed following

$$P_{miss}(\theta) = \frac{1}{K} \sum_{kw=1}^{K} N_M(kw, \theta) \div N_T(kw)$$ (3)

$$P_{FA}(\theta) = \frac{1}{K} \sum_{kw=1}^{K} N_F(kw, \theta) \div (T_{sp} - N_T(kw))$$ (4)

where $N_T(kw)$ is the number of true occurrences of keyword $kw$ in the corpus, $N_M(kw, \theta)$ is the number of false alarms with scores higher than $\theta$, $N_M(kw, \theta)$ is the number of true detections whose score is lower than $\theta$ plus the number of undetected occurrences, $T_{sp}$ is the duration in seconds of the corpus and $\beta$ is a constant set to 0.999.

We use the IBM Speaker-Adapted DNN (SA DNN) system for recognition; this uses a deep neural network acoustic model with a Stochastic Feature Mapping in a speaker adaptive feature
space [3]. From the output lattices of the ASR, we obtain Confusion Networks (CN) [4], a compact linear representation of the lattices designed to optimize word error rate. CNs are created by clustering lattice edges into an ordered series of “bins” based on time similarity. CNs often improve ASR performance over lattices [4] and are convenient for rescoring since they allow us to use dynamic programming to compute the alignment and edit distance between the reference string and the CN.

3. State of the Art

3.1. Rescoring Techniques

There are a number of current approaches to keyword score normalization for rescoring purposes.

**Exponential Normalization:** an optimal keyword-specific threshold for ATWV can be computed following equation 5 [5]. To translate this thresholding strategy into a global threshold setting, Exponential Normalization compresses the posterior scores, $s_{kw,i}$, such that scores are attenuated if they are below its keyword-specific threshold $\theta_i(kw)$, and boosted otherwise (Eq. 6), where the global threshold can be set to any arbitrary number in $(0, 1)$.

$$\theta_i(kw) = \frac{N_T(kw)}{T_{sp}/\beta + (\beta - 1)N_T(kw)/\beta}$$

$$s'_{kw,i} = \frac{s_{kw,i}}{\log(\theta_i)/\log(\theta_i(kw))}$$

**pFA Normalization:** The global threshold used in the program evaluation requires that typical posterior scores of true hits are comparable to each other. This may not be true since factors like the query length, the duration and the word frequency in the training data have a direct impact on the score spectrum of each keyword. Moreover, true occurrences of a keyword may not appear in the development data. To avoid rescoring using true hits of the data, the authors in [6] propose to normalize scores using the distribution of False Alarms on development data in the following way: every false alarm hit score is sorted in decreasing order and their probability of false alarm is computed as the rank of the hit in the sorted list divided by the total number of hits in the corpus. When a new hit is detected, one looks up the false alarm mapping (using linear interpolation) to assign its pFA. Finally the hit is rescored with $1 - pFA$.

**Reranked Posterior Scores (RPost):** pFA normalization gives good TWV improvements when the list of false alarms in the development data is long enough. When it is not, [7] proposes to use a global map that will assign each hit the average posterior value of its rank. Specifically, the pFA map is used to assign each hit a rank and then the average of the posteriors of every keyword is returned. Reranked posterior scores work well in detection tasks with global thresholds because they expand the posterior score space of a specific keyword to a global set of scores independent of the keyword.

**Sum-To-One Normalization (STO):** in STO, the hit score is divided by the sum of all scores of the same keyword. This normalization thus boosts scores for low scoring keywords, minimizing the number of missed true detections [8], and reduces scores of the lowest scoring hits of each keyword, potentially reducing false alarms. It can also have the undesired effect of reducing scores of keywords with a high number of detections. To avoid this, a variant of STO called β-STO leaves all hits with scores higher than $\beta$ untouched, and applies STO to the rest.

**Machine Learning (ML):** another popular approach to rescoring PLs has been using machine learning classifiers. The authors in [7] build a dataset with mostly probabilistic scores and use the Powell Method to learn a linear classifier that maximizes MTWV. Their results improved Exponential Normalization, False Alarm normalization and STO by 1 and 2% absolute points. In previous work we proposed a two-stage cascaded approach [9] with some success and improved MTWV with respect STO by about 0.5% absolute gain. The method presented here greatly improves over our previous effort due to more feature engineering, a better IV/OOV morphological pipeline (presented in Section 3.2), and a new strategy to maximize MTWV on linear classifiers (presented in Section 3.3).

3.2. Standard and Morphological Pipeline for KWS

An important part of the Babel program and a challenge for all KWS systems is the ability to detect out-of-vocabulary tokens, a scenario that will be very common in an actual low resource setting. To improve the KWS performance on OOV queries, the IBM-led Babel group follows two different strategies, depending on whether the language is analytical (very low ratio of morphemes to words) or synthetic (words are commonly formed by affixing morpheme to roots).

**Standard Pipeline:** for languages like Lao, which is a tonal, analytical language and has no inflectional morphology, OOV queries are mapped to IV queries using a phone confusability model in a WFST framework [10]. Each OOV search is replaced with an IV search, in which the IV hit is penalized according to this phone confusion model. IV queries with no hits or whose hits have very low scores are treated like OOV queries, i.e. we apply the same query expansion using the phone confusability transducer.

**Morphological Pipeline:** in synthetic languages like Amharic, Bengali and Zulu, words can be segmented into roots and morphemes, which can be recombined to form new words. Taking advantage of this, [11] proposes to perform unsupervised morpheme segmentation on raw transcripts and then create a finite state transducer that maps words to morphemes. Thus, if an OOV query is submitted, the new token will be segmented in morphemes and will be searched in a morph-based CN. These morph CNs are created from morph lattices in which the Language Model scores are removed, and only the acoustic scores are kept and used to generate posterior probabilities. In this new pipeline two sets of CNs are kept, the word confusion network, for IV queries, and the morph confusion network, for OOV queries and IV queries with no hits or low scoring hits.

3.3. Optimizing TWV metric

Learning prediction models in this keyword search setting is difficult for two reasons, both derived from the use of the TWV metric. First, the datasets are very unbalanced, with an average of 95% false alarm hits against just 5% correct hits in our corpora. This is a natural consequence of inflicting smaller penalties to false alarms than to missed occurrences. Secondly, the penalties are a function of the number of true occurrences of the keyword in the corpus, and vary from keyword to keyword. To address the first point in [9], we used different cost penalties on correct hits and false alarm hits, with some success. In this paper, and following the spirit of the TWV learning framework published in [12], we apply keyword-dependent cost penalties inspired by the TWV metric, following equations $C_{\text{Miss}}(kw) = 1/N_{\text{true}}(kw)$ and $C_{\text{FA}}(kw) = \beta/(T_{\text{speech}} - N_{\text{true}}(kw))$. 

$$\left| \text{TWV} \right| = 1/N_{\text{true}}(kw) - \beta/(T_{\text{speech}} - N_{\text{true}}(kw))$$
4. Two-stage Machine Learning Approach

Our rescoring pipeline has two main stages: rescoring the CNs and transcriptions so as to minimize the Levenshtein distance between both. Each arc is then labeled as correct or incorrect and a feature vector is computed for each arc. We extract a variety of features at the arc, bin, segment, and conversation level, including:

Lexical Features: The percentile of the word frequency in the transcriptions, whether the arc is labeled as a silence, an epsilon, or a non-speech tag, the number of syllables of the token, and the syllable index of its primary and secondary stress.

Phonetic Features: count of phones (and clicks for Zulu) of each word in its lexicon entry, as well as flags for rare and foreign phonemes, as described in the language pack provided by IARPA. Also included are four binary features indicating whether the word begins/ends in an unvoiced consonant or glottal stop.

Syntactic Proxy Features: We include the model M class [13] to which a token belongs. Model M creates a class-based n-gram language model in which each word belongs to a single class and the prediction of each word depends on the class information such that similar words in the same context belong to the same class.

Probabilistic Features: the probability of the arc being correct, the probability of the arc being correct given the other tokens in the confusion bins, and the probability of the arc being correct given the set of phones in its pronunciation lexicon entry. We also compute rank-normalized probabilities of false alarm, following [6], and global re-ranked posterior scores as described in [7].

Structural Features: This set of features is extracted directly from the CNs. At the arc level, they include the posterior score, the arc rank and the ratio between the arc rank and the confusion bin size. At the bin level we include the confusion bin size, the bin number at the segment and conversation level, the distance in seconds and bins to the previous and next silence and to the beginning and end of the segment and conversation. We also include the number of prior appearances of the token in the segment and conversation.

We apply feature selection to reduce the size of our training set, to alleviate computational requirements and to improve the performance of our classifiers. In this work, we report feature selection results using Quadratic Programming Feature Selection (QPFS) [14], which maximizes relevance to the arc labels while minimizing redundancy among the subset of selected features. The ranking of most relevant features as computed by QPFS varies greatly among languages. Top ranked features in- tersections so as to minimize the Levenshtein distance between both. Each arc is then labeled as correct or incorrect and a feature vector is computed for each arc. We extract a variety of features at the arc, bin, segment, and conversation level, including:

Lexical Features: The percentile of the word frequency in the transcriptions, whether the arc is labeled as a silence, an epsilon, or a non-speech tag, the number of syllables of the token, and the syllable index of its primary and secondary stress.

Phonetic Features: count of phones (and clicks for Zulu) of each word in its lexicon entry, as well as flags for rare and foreign phonemes, as described in the language pack provided by IARPA. Also included are four binary features indicating whether the word begins/ends in an unvoiced consonant or glottal stop.

Syntactic Proxy Features: We include the model M class [13] to which a token belongs. Model M creates a class-based n-gram language model in which each word belongs to a single class and the prediction of each word depends on the class information such that similar words in the same context belong to the same class.

Probabilistic Features: the probability of the arc being correct, the probability of the arc being correct given the other tokens in the confusion bins, and the probability of the arc being correct given the set of phones in its pronunciation lexicon entry. We also compute rank-normalized probabilities of false alarm, following [6], and global re-ranked posterior scores as described in [7].

Structural Features: This set of features is extracted directly from the CNs. At the arc level, they include the posterior score, the arc rank and the ratio between the arc rank and the confusion bin size. At the bin level we include the confusion bin size, the bin number at the segment and conversation level, the distance in seconds and bins to the previous and next silence and to the beginning and end of the segment and conversation. We also include the number of prior appearances of the token in the segment and conversation.

We apply feature selection to reduce the size of our training set, to alleviate computational requirements and to improve the performance of our classifiers. In this work, we report feature selection results using Quadratic Programming Feature Selection (QPFS) [14], which maximizes relevance to the arc labels while minimizing redundancy among the subset of selected features. The ranking of most relevant features as computed by QPFS varies greatly among languages. Top ranked features include the arc posterior score, the probability of the token being correctly recognized in the training partition, a non-speech tag flag, the number of arcs in the confusion bin, the rank of the arc in the confusion bin, the model M class of the arc token, the reranked posterior score as computed in Section 3.1, and the number of appearances of the token in CN until that point. Other highly ranked features include flags for the presence of glottal stops for Asamese, Bengali and Zulu. Top ranked language-dependent features are mainly phonetic features, i. e. number of given phone in token. The size of the subset of features that returned best TER improvements for Asamese, Bengali, Lao and Zulu is 20, 65, 45 and 65 respectively.

Since the problem of rescoring confusion networks is inherently a problem of reranking the arcs of each confusion bin, we train and test svm-ranking classifiers [15] using the library SVM-light [16]. Specifically the training consists of pair-wise comparisons of all arcs in the same confusion bin, and the highest-scoring arc is predicted as correct.

4.2. Rescoring Keyword Search Hits

In the second stage of our rescoring procedure, we train logistic regression classifiers for detecting false alarms and correct hits, using the LIBLINEAR library [17] with the instance-based weights from Section 3.3. Logistic regression is especially convenient for this task because its output is a posterior probability of the instance being correct or a false alarm, with a threshold imposed by the logistic function in 0.5. For this stage we extract a total of 90 features from the posting lists entries and from their matched confusion bins. Among other features, we include acoustic, duration, CN and rescoring features:

- CN features: we match the posting list entries to the rescored confusion networks and compute the number of CN bins, length of the keyword in tokens and their ratio and inverse ratio. From the observation that multitoken keywords usually match to a sequence of CN bins whose length is much higher than the number of tokens of the keyword due to the presence of epsilons in almost every bin, we compute features that measure the presence of epsilons in the current hit. These include: the ratio between the number of epsilon arcs and confusion bins, the sum of original scores of the epsilon arcs divided by the sum of rescored posteriors of epsilon arcs divided by the number of confusion bins. Three analogous features are computed dividing by the length of the keyword.

- Posterior features: Taking the matching arcs from the CNs, we aggregate both the original p and the rescoring posterior scores r − p using the maximum, minimum, arithmetic average, geometric average and product functions. We also compute the probability of a hit being correct pCorrn(kw, bin(p)) or false alarm pFAk(kw, bin(p)) by binning hits from the same keyword in a histogram with twenty equal spaced bins.

- Rescoring features: We include the scores output by the rescoring techniques reviewed in Section 3, that is, the exponential normalization score and its keyword-specific threshold, rank-normalized probability of false alarms, reranked posterior scores, STO scores and STO scores with threshold β = 0.8.

- Acoustic features: We extract the pitch contour of the posting list entry timespan and compute its median, mean, standard deviation, maximum and minimum, the number of unvoiced cycles in the segment and its percentage, the harmonics-to-noise ratio (in dB) and noise-to-harmonics ratio, and the autocorrelation of the pitch contour. We also extract the pulses and include the number of pulses, the number of periods and their mean and standard deviation, along with the number of voice breaks and their percentage. Finally we include jitter values (local, local in seconds, its relative average perturbation (RAP) and its 5-point period perturbation quotient) and shimmer values (local, local in dB, and its 3-, 5-, and 11-amplitude perturbation quotient).

- Duration features: We compute the average duration of true hits d(kw) and all hits (including false alarms) d(kw) for every keyword. From these aggregates we compute a total of twelve duration features that include: ratio, inverse ratio, differ-
Table 1: Most prominent features for PL reranking (Limited LP) according to QFPS.

<table>
<thead>
<tr>
<th>Assamese</th>
<th>Bengali</th>
<th>Lao</th>
<th>Zulu</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod(r-p)</td>
<td>prod(r-p)</td>
<td>prod(r-p)</td>
<td>prod(r-p)</td>
</tr>
<tr>
<td>pCorrr (kw, p)</td>
<td>pCorrr (kw, p)</td>
<td>pCorrr (kw, p)</td>
<td>pCorrr (kw, p)</td>
</tr>
<tr>
<td>0.8-STO</td>
<td>0.8-STO</td>
<td>0.8-STO</td>
<td>STO</td>
</tr>
<tr>
<td>prod(p)</td>
<td>Reranked Post</td>
<td>Reranked Post</td>
<td>Reranked Post</td>
</tr>
<tr>
<td>Cost_f.A</td>
<td>#eps/#bins</td>
<td>#eps/#bins</td>
<td>#eps/#bins</td>
</tr>
<tr>
<td>GAvg(sign(r-p))</td>
<td>Cost_f.A</td>
<td>Cost_f.A</td>
<td>AutoCorr</td>
</tr>
<tr>
<td>Reranked Post</td>
<td>prod(p)</td>
<td>OOV</td>
<td>prod(r-p)</td>
</tr>
<tr>
<td>prod(r-p)</td>
<td>GAvg(r)</td>
<td>KST-Exp</td>
<td>prod(r-p)</td>
</tr>
<tr>
<td>prod(avg(sign(r-p)))</td>
<td>prod(avg(sign(r-p)))</td>
<td>(d(kw) - d)</td>
<td>len(keyword)</td>
</tr>
</tbody>
</table>

Table 1: Most prominent features for PL reranking (Limited LP) according to QFPS.

ence, absolute value of difference and its square between the hit and $d_i(kw)$ and $d(kw)$.

**TWF features:** We include in the feature vector the penalties imposed by the TW metric when the hit is incorrectly classified as False Alarm and incorrectly classified as a Correct hit from Section 3.3. It is critical to note here that both features are included, not just the one corresponding to the true label, which would introduce a bias into the learning process.

We use QFPS once more for feature selection. Table 1 shows the top 10 best features for the prediction task. Features that are most prominent include aggregations of the reranked posteriors, 0.8-STO scores, the penalty cost of FA for that posterior, aggregations of the sigmoid function of the reranked posteriors, the reranked posterior score, the fraction of epsilon arcs in the matched confusion bins, duration features, and the histogram probability of the hit being correct, and an OOV flag.

### 4.3. Results

In this section we discuss the results obtained at each stage of the cascaded strategy. Table 2 shows TER for baseline and rescored Confusion Networks for Assamese, Bengali, Lao and Zulu. We obtain absolute TER gains ranging from 0.8% in Assamese to 3% in Zulu by reducing the percentage of insertions ($\Delta I \in (1.0, 2.6)$) and aggressively reducing substitutions ($\Delta S \in (3.8, 11.2)$) in the 1-best transcriptions, at the cost of increasing the percentage of deletions.

Table 2: Baseline and Rescored Confusion Networks TER results.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Rescored CN</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assamese</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Bengali</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Lao</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Zulu</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 3: MTWV for Assamese, Bengali, Lao and Zulu LPPs. Bold font indicates best system performance and * indicates best system performance outside our ML approach.

The four tables show the same trend for the complete set of keywords and small variations when we examine IV and OOV keywords separately. The cascaded Machine Learning approach is the best rescoring algorithm for every language, keyword list, and keyword type, except for rescoring OOV queries from the dev keyword list in Bengali. Our method improves over every other rescoring technique, with relative gains ranging from 5.02 to 36.98% with respect to the next best rescoring method and the complete keyword lists. The improvements are especially important for the OOV subsets where the gains reach 42.4% relative improvement. With respect to the other rescoring methods, the Reranked posterior scores is usually the best performing method and Exponential Normalization is the weakest. The remainder of the ranking is STO < pFA < β-STO.

### 5. Conclusions

In this paper, we describe a two-pass rescoring approach for improving Keyword Search performance in low-resource languages. This approach consistently generates performance that is between 5.02% and 36.98% higher than the current state-of-the-art score normalization approaches. A secondary result of this rescoring is improvements to ASR performance, 0.7% to 3.0% absolute reductions to TER.
6. References


