Beyond Posteriorgram: Bottleneck Features for Keyword Search

Bolaji Yusuf  
Electrical and Electronics Eng. Dept.  
Bogazici University  
Istanbul, Turkey  
bolaji.yusuf@boun.edu.tr

Batuhan Gundogdu  
Electrical and Electronics Eng. Dept.  
National Defense University, Naval Academy  
Istanbul, Turkey  
mbgundogdu@dho.edu.tr

Murat Saraclar  
Electrical and Electronics Eng. Dept.  
Bogazici University  
Istanbul, Turkey  
murat.saraclar@boun.edu.tr

Abstract—Template matching approaches have been proposed as an alternative to large vocabulary continuous speech recognition (LVCSR) based systems for keyword search. These have been shown to have no performance discrepancy between terms in the training vocabulary and out of vocabulary (OOV) terms. Those methods have often relied on the use of posteriorgram as the features for the search. In this paper, we propose the use of bottleneck features instead of posteriorgram because of their potential for cross-lingual transfer learning. We show the feasibility of bottleneck features for template-matching based keyword search using by learning different representations for the features including a mixture of Gaussians and a representation based on a joint distance metric learning framework.

Index Terms—keyword search, low resource languages, bottleneck features, distance metric learning, dynamic time warping.

I. INTRODUCTION

Keyword search (KWS) is a subset of spoken content retrieval. In KWS, a user provides a text query which is then searched in an audio document. A KWS system is expected to return a ranked list of hypotheses that designate the location of the keyword in the document. The contemporary approach to KWS involves performing text retrieval on the output of an LVCSR system [1]. This generally involves the construction of a lattice-based index [2] or similar structure. An LCVSR system trained on a large amount of data can achieve a low word error rate (WER) and, subsequently, good performance on the search task.

In cases where the amount of available training data is limited, however, there is an overall deterioration in the speech recognition performance which impacts the search performance negatively. The extreme case of this degeneration can be seen on so called OOV terms where at least one of the words in the query term does not appear in the training dataset. As the output of the LVCSR system does not ordinarily contain words out of its vocabulary, such words cannot be found by the system on any document and so the system never returns a hit for OOV queries even when they occur in the document. The fairly obvious observation that the OOV rate increases as the amount of training data decreases necessitates a way of handling such terms. Several solutions have been proposed including the use of sub-word models [3]–[6], lexicon expansion [7], point process models [8], proxy keywords [9], [10] and similarity learning [11].

In this work, we investigate the efficacy of similarity learning based KWS for bottleneck features. In the next section, we examine the bottleneck features as well as the different representations that we learn on the bottleneck features to make them feasible for search.

II. RELATED WORK

A. Bottleneck Features

Bottleneck features are obtained from the activations of a hidden layer of an acoustic model deep neural network (DNN). The bottleneck layer is often orders of magnitude smaller than surrounding layers. This has shown to reduce the performance of the DNN [12]. In [13], a low-rank matrix factorization was proposed which allowed the DNN to maintain its ASR performance despite the introduction of a layer with significantly fewer nodes. Extracting the BN features in this way also allows for richer features since no nonlinearity is applied to the activations. In [14], it was found that putting the bottleneck just before the penultimate layer of the DNN was optimal.

B. Gaussian Posteriorgram

Gaussian posteriorgrams have been used in place of spectral features for search in very low to zero resource scenarios because they have the advantage of being less sensitive to
speaker variability [15]. The Gaussian posteriorgram is a vector whose $i$-th element represents the probability of a speech frame having been generated by the $i$-th Gaussian in an unsupervised Gaussian mixture model (GMM).

In this work, the GMM is used to model the BN feature distribution. This not only allows us to remove some of the variability in the BN features but also to use some well studied distance metrics in literature.

C. Joint Distance Metric Learning

Joint distance metric learning (JDML) was proposed recently as an extension to the distance metric learning (DML) approach to KWS [16]. DML involves the use of a Siamese neural network to learn a distance metric for subsequence dynamic time warping (sDTW) based KWS. At search time, the text query is converted into its phonemic representation using a grapheme-to-phoneme conversion tool [17]. A template for each phone is required which, for DML, is obtained via an ad-hoc procedure of passing the phone-wide averages through the neural network. JDML seeks to learn the each phone’s correct centroid for the learned distance metric by introducing an asymmetry to the network. The asymmetry is in the form of an extra linear layer which is fed with one-hot vectors representing the pertinent phone.

![Figure 2: Subsequence dynamic time warping scheme [18].](image)

III. EXPERIMENTS

In this section we describe the parameters of the experiment; the feature kinds evaluated upon, distance metric type used for search, the data and the metrics used to measure the performance of the system.

A. Dataset

The system is trained and evaluated on recordings of Turkish telephone conversation provided by the IARPA babel program [19]. The training set contains 10-hours of conversational speech on which Viterbi forced alignment is used to obtain frame level alignments. The evaluation is done on a separate 10-hour set using 284 keywords.

1babel105b-v0.4 (dev:kwlist)

B. Feature Type

We used two bottleneck representations for our experiments. The first, which we call raw BN, is the use of the 42-dimensional activations obtained from the BN extractor directly. For the second feature kind, termed Gauss BN (GBN), we trained a GMM with 1024 Gaussians on the BN features and use the Gaussian posteriors as the features for search.

C. Distance Metrics

The search is conducted with the sDTW algorithm. Given a query,

$$Q = [q_1, \ldots, q_N]$$

a document

$$X = [x_1, \ldots, x_M]$$

and a distance function, $d(., .)$, the algorithm returns an optimal alignment path, $\Phi$, between $Q$ and a subsequence of $X$ and a score given by:

$$\text{score} = 1 - \frac{1}{\text{length}(\Phi)} \sum_{(i,j) \in \Phi} d(q_i, x_j)$$

We experimented with four distance functions in the for the sDTW search:

(i) The cosine distance defined as:

$$d_{\text{cos}}(q_i, x_j) = 1 - \frac{q_i^T x_j}{||q_i|| \cdot ||x_j||}$$

The cosine similarity measures the angular distance between two vectors. For search, the template for a phone is obtained from the average of its samples in the training set.

(ii) The log-cosine distance defined as:

$$d_{\text{log}}(q_i, x_j) = \log(\epsilon + \frac{q_i^T x_j}{||q_i|| \cdot ||x_j||})$$

where $\epsilon$ is a small number used to ensure that the operand of the log function is non-zero. The log-cosine distance, used on posterior probability features, measures the probability of two frames being generated by the same class (or Gaussian). As with the cosine distance, a phone’s template is obtained from the average of its samples in the training set. Since the log-cosine can only operate on nonnegative vectors, we only use it on GBNF.

(iii) The JDML distance which is learned on the training set is used for the sDTW. For each query phone, the template can be computed by passing a one-hot vector with a one at the index of the pertinent phone through the lower half of the JDML network and obtaining its activations on the shared layer; the document representation is computed by passing the raw BN through the shared W2 layer and storing the activations. The search can then be conducted on these stored activations using:

$$d_\sigma(q_i, x_j) = \sigma(q_i^T x_j + c)$$
where $\sigma(.)$ is the logistic sigmoid function and $c$ is the bias of term of the JDML network.

(iv) The deeper JDML (DJDML) which is obtained by modifying the JDML network architecture to include two extra layers on the document (top) half of the JDML before the shared layer. The extra layers are a rectified linear (ReLu) layer followed by a sigmoid layer; this was done to transform the raw BN to a space whose centroids are more easily representable by the linear layer on the query side. The search is conducted much in the same way as for the JDML.

**Figure 3: DJDML network. Differences from JDML are shown in red.**

**D. Score Normalization**

The performance of the systems are evaluated using the maximum term weighted value (MTWV) metric. The term weight value (TWV) is a weighted sum of the precision and recall at some, $\theta$.

$$TWV(\theta) = 1 - (p_{MISS} + \beta p_{FA})$$

where $p_{MISS}$ is the probability of a miss and $p_{FA}$ is the probability of a false alarm, $\beta$ is a parameter that controls the relative costs of incurring false alarms and misses. The MTWV is the value of the TWV at the $\theta$ value that maximizes it.

Since the TWV computation requires the use of one global threshold for all keywords, it is important to normalize the scores for each keyword in order to ensure that the dynamic ranges of scores for different keywords are aligned. For this, we use a variant of the b-norm [11] computed at the 90-th percentile. The normalized value for each score, $s$, is computed thus:

$$\hat{s} = \frac{s - m}{\sigma\{s : s \geq m\}}$$

where $m$ is a score that is greater than exactly 90% of the hypotheses for the keyword.

**IV. RESULTS**

**A. Individual Results**

Our initial experiment using cosine distance on the raw BN scores was unsuccessful which showed us that the BN features are not readily linearly separable. To rectify this, we train an unsupervised GMM model which we use to generate Gaussian posteriorgram. We then conduct the search using cosine distance again and obtain a positive MTWV of 0.028. Since the Gaussian posteriorgram, unlike the BN features, contains features which are strictly nonnegative, it is possible to conduct the search this time with the log-cosine distance. This yields an MTWV of 0.1035, a significant improvement over the cosine distance experiment.

As a further experiment, we attempt to apply the recently proposed joint distance learning algorithm to the raw BN features and are able to obtain further improvement to 0.1164 as can be seen on Table I. Since the JDML was initially proposed for phone posteriorgrams, we attempt to modify the BN features to make them more similar to the query side by deepening the JDML network. Using the metric and phone templates learned in this way to conduct the search results in an MTWV of 0.1809 which is a significant improvement in over the original JDML architecture.

In the interest of completeness, we also conduct the metric learning experiments on the Gaussian posteriorgrams to see if it is beneficial to build the JDML on those. However, we are not able to obtain any further improvements over the JDML system built with raw BN. In fact, using JDML metric on Gaussian posteriorgram performed worse than using the log-cosine distance.

**TABLE I: MTWV Obtained From Various System Configurations**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Raw BN</th>
<th>GBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine</td>
<td>0</td>
<td>0.0228</td>
</tr>
<tr>
<td>Log-cosine</td>
<td></td>
<td>0.1035</td>
</tr>
<tr>
<td>JDML</td>
<td>0.1164</td>
<td>0.0961</td>
</tr>
<tr>
<td>DJDML</td>
<td>0.1809</td>
<td>0.1036</td>
</tr>
</tbody>
</table>

**B. Comparison and Fusion with Phone Posteriorgram**

We compare our results with that of a JDML system built on phone posteriorgram (PPG) and although we were unable to match its performance with any of our systems, we find that when we combine PPG system with the BN JDML and
DJDML ones, we are able to improve upon its performance by 2% and 4% respectively.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Raw BN</th>
<th>PPG</th>
<th>Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDML</td>
<td>0.1164</td>
<td>0.2617</td>
<td>0.2679</td>
</tr>
<tr>
<td>DJDML</td>
<td>0.1809</td>
<td>0.2617</td>
<td>0.2732</td>
</tr>
</tbody>
</table>

V. CONCLUSION AND FURTHER WORK

In this paper, we have used bottleneck features for template matching based keyword search. We found that training Gaussian posteriorgrams on top of the bottleneck features is a viable procedure for creating such templates. Furthermore, using a modified JDML network, we were able to build a strong BN feature based system which improves a phone posteriorgram based system on combination. We also learned that building the JDML on top of Gaussian posteriorgram was not very useful, and, considering the extra computational cost of such a procedure, might even be counterproductive.

Since the bottleneck features can be generated in multilingual manner, a natural extension of this work would be to use multilingually trained BN features. This would be particularly useful when the language of interest has too little data to train a DNN. Using features from a BN generator trained on several other languages, a GMM or JDML based system could then be trained on the language of interest.

ACKNOWLEDGMENTS

This study was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under Project 116E076.

REFERENCES