Multitask Learning Based Muti-Examples Keywords Spotting in Low Resource Condition

Jianbin Yang, Jian Kang, Wei-Qiang Zhang, Jia Liu,
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
Email: {yjb15, kangj13}@mails.tsinghua.edu.cn, wqzhang@tsinghua.edu.cn

Abstract—Keywords Spotting (KWS) is a critical task in speech recognition, aiming to spot the pre-selected keywords out of a continuous speech. In a typical low resource condition, we can only obtain dozens of examples of each keyword. How to make full use of multi-examples information to build an effective keyword spotting system is the present challenge since traditional keyword spotting technologies are not suitable. In this paper, we propose a multi-examples keywords spotting system, which gains a significant performance improvement by applying multitask learning technologies to extract feature and build model. A post-processing method is also used to reduce the false alarm rate.

Keywords—Keywords: keywords spotting, filler model, multitask learning, post-processing

I. INTRODUCTION

As an important artificial intelligence technology to process a huge amount of speech data, keywords spotting (KWS) plays an important role in information retrieval, electronic devices awaken and national security. After decades of researching, KWS based on large vocabulary continuous speech recognition (LVCSR) [1][2], templates based KWS [3][4][5][6] and KWS based on Hidden Markov Model and Filler Model (HMM-Filler) [7] [8] [9] [10]became three mainstream methods of keyword spotting. For KWS based on LVCSR, researchers use continuous speech recognition method to recognize the speech data as text, then locate the keywords using a text searching algorithm. The traditional speech recognition engine is based on Gaussian mixture model - Hidden Markov model (GMM-HMM) framework. With the rise of artificial neural networks, a variety of networks replace the traditional acoustic model, such as deep neural network(DNN) [11], convolutional neural network (CNN) [12] and recurrent neural networks (RNN) [13], they all have significant performance improvement over GMM. The LVCSR method could achieve an excellent performance when sufficient training resources are supplied. On the contrary, if training resources are limited or the pronunciation dictionary is unknown, this method cannot be used since it is impossible to train a speech recognition engine.

For the very low resource condition, templates based keywords spotting algorithm is developed. For given examples and testing speech, the acoustic features are extracted, then researchers match example features (as templates) with testing features using dynamic time warping algorithm (DTW). To deal with the situation in which a keyword has more than one examples, there are usually two approaches to cope with this situation. The first one is averaging the examples to a single example, by which a faster computing speed can be gotten, but some information of multiple examples will be lost. The second one is matching every example and fusing the results, by which the information of all multiple examples can be used, but the calculation grows rapidly as the number of examples increases. [4] reported a new method to generate templates, which trims the outputs of an RNN hidden layer to fixed length data segments as features, so that the matching among templates and testing features can be simply completed using dot product since the length of templates and testing features are fixed. Similar to the two methods above, this method also faces the problem that the information of multiple examples cannot be fully used.

As for the KWS based on Hidden Markov Model-Filler Model [7], researchers model each keyword as a Hidden Markov states chain, while all the other non-keywords share one HMM as a filler model. These HMMs could be monophone or tri-phone, which can be trained from an existing large dataset. During testing, the acoustic score of each acoustic state for each frame of testing data is computed, then the Viterbi decoding algorithm [14] is used to locate the keywords and compute their confidence scores. The speed of decoding is closely related to the topology of HMMs.

In a real-world low resource condition, dozens of examples of each keyword can be obtained, yet it is impossible to build a speech recognition system since the dataset is too small and linguistics information is unknown. Considering the downside of traditional methods, we develop a filler-based multi-examples KWS system using DNN in the multitask learning framework. The paper is organized as follows. The KWS system using DNN is introduced in chapter II, the multitask learning and post-processing methods are described in chapter III. Our experiment results are shown in chapter IV, and in chapter V we give our conclusion.

II. KWS SYSTEM USING DNN

We build a KWS-DNN system which can be used for keywords spotting in the low resource condition similar to [8]. We model each keyword as an HMM, while all the other non-keywords are modeled as a filler model. Simultaneously, we train an HMM for silence independently. A DNN is trained to compute the posterior probabilities of each state in those HMMs, and then the Viterbi algorithm is used to find the optimal states sequence to locate the keywords. Each part of the system is described in detail as follows.
A. Keyword Model

In this paper, we model keywords with two types of models of different granularity. The first one is whole-word modeling, in which each keyword is modeled as a unit corresponding to the output of neural network directly. The second one is sub-word modeling, in which each keyword is modeled as a Hidden Markov states chain whose states are simply connected from left to right. Assume a keyword has four hidden states, the topology of its HMM is shown in figure 1. State 1 and 6 represent the virtual start and end, which aim to connect other HMMs conveniently, and there are no feature vectors corresponding to them. Since the data we obtain is limited and the keywords list is fixed, neither mono-phone nor tri-phone model will be used. The topology of HMMs are hand designed and the average length of keyword examples determines the number of hidden states.

B. Filler Model and Silence Model

As for filler model, we believe its pronunciation are different from those of keywords. Considering their randomness, we model filler as a single state self-circulation HMM or a multiple states free-jumping HMM. Figure 2 shows a free-jumping filler model with two hidden states. Besides, considering the significant difference of spectrum between silence and filler, we train a skip-back HMM for silence independently. The silence segments can be determined by applying the voice activity detection (VAD) based on sub-band spectral entropy [15].

C. Feature Extraction

In a speech recognition task, there are usually two kinds of features that can be chosen. One is hand-designed basic acoustic features, such as filter bank feature (Fbank), perceptual linear prediction feature (PLP) and Mel Frequency Cepstral Coefficient feature (MFCC). The other is data-driven high level features, such as bottleneck features and LSTM features. We can generate data-driven features by applying a series of nonlinear transformations to the hand-designed features. For instance, bottleneck feature is the output of bottleneck layer in a previously trained DNN [16], and LSTM feature is the output of a hidden layer in an already trained RNN, trimmed into a fixed length data segment [4]. Compared to the hand-designed features, these data-driven features have a stronger description ability for the nonlinear characteristics of speech. Meanwhile, since they are derived from networks trained by a large amount of data, we can use them to augment data in the low resources condition.

D. Model Training

We estimate the posterior probabilities using a DNN. The outputs of the DNN correspond to the probabilities of keywords in whole-word modeling, while in sub-word modeling they correspond to the probabilities of sub-word states. Filler and silence are modeled in the same way in whole-word modeling and sub-word modeling, as it is shown in figure 3. During training, we use sigmoid function as the activation function and cross-entropy as the cost function, and apply backpropagation algorithm to update the weights.

E. Decoding

To improve the efficiency of testing, we only process the speech parts determined by applying VAD to testing data. Then we input the bottleneck features extracted from testing speech to a trained DNN to get posterior probabilities of states. Since we want to obtain the most likely states sequence, and the parameters of HMMS and observations are already known, the Viterbi decoding algorithm is applied. Compared to the complex decoding network in LVCSR, our decoding network with loop structure is much simpler. The decoding network is shown in figure 4.

Each result of decoding is constituted with four parts:
- Keyword
- Start time
- Duration
- Confidence score

A result may be a false alarm if its confidence score is relatively low, and too much false alarm can have a terrible influence on system performance. In the next chapter, we will introduce a post-processing method to reduce the false alarm.

III. MULTITASK LEARNING AND POST PROCESSING

Multitask learning (MTL) [17] [18] is a kind of machine learning technique which is designed to improve the generalization ability of models. The key to the successful application of MTL is that the tasks should be relevant but not necessarily similar. It means that these tasks can share a part of the representation of the features on a certain abstraction level. If
the tasks are indeed similar, multitask learning algorithm helps to transfer knowledge between tasks by increasing the amount of training data. If the tasks are related but not similar, learning them together can limit the functional space of each task, increasing the generalization ability of each task. Multitask learning algorithm is more effective when the training set is smaller than the model size, so it is suitable for our task in low resource condition.

A. MTL–Bottleneck Feature

Since all layers of neural network contain information that classifies the phonemes of a specific language, and different languages may share similar phonemes, it is possible to use neural network trained for a language to extract bottleneck feature of another language [19]. In the low-resource condition, using bottleneck feature extracted from multilingual DNN is an effective method.

In this paper, we use a multitask learning approach to extract bottleneck features. The architecture of this approach contains different softmax output layers for different languages, as shown in figure 5. During the training process, different tasks share the hidden layer units.

B. MTL–Model Initialization

Using the thought of transfer learning [20], we use a DNN trained on multilingual dataset as a seed to initialize KWS-DNN. After that, a randomly initialized output layer is added, as shown in figure 6. Finally, the KWS-DNN is tuned by backpropagation algorithm using the small keywords dataset until convergence.

C. MTL–Model Training

From above we have already known that multitask learning algorithm can be used to extract features. Now we focus on applying multitask learning algorithm to train multi-targets neural network models. In addition to the main task, we can also set up some auxiliary tasks to assist the model training. In this paper, we set the recognition of the sub-word state as the main task, and design two auxiliary tasks. One is the recognition of whole-word, the other is the recognition of the contexts of sub-word states. The contexts of a state mean its preceding and succeeding states in time.

D. Post Processing

[21] pointed out that in the KWS system based on continuous speech recognition, it is effective to improve the system performance by rescoring the confidence scores of detection results. In this paper, we use the keywords examples in training set to perform post-processing method as follows:

For a detected result of a keyword, calculate the average DTW distance $d_{\text{average}}$ between the result and all training
### TABLE I
**F1 MEASURES OF Fbank SYSTEM AND MTL-BOTTLENECK FEATURE**

<table>
<thead>
<tr>
<th></th>
<th>Baseline(Fbank)</th>
<th>MTL-bottleneck</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.258605</td>
<td>0.577778</td>
</tr>
</tbody>
</table>

### TABLE II
**F1 MEASURES BOTTLENECK SYSTEM WITH SEED, MTL, AND POST PROCESSING**

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottleneck(1)</td>
<td>0.577778</td>
</tr>
<tr>
<td>Bottleneck+seed(2)</td>
<td>0.593625</td>
</tr>
<tr>
<td>Bottleneck+seed+MTL(whole-word)(3)</td>
<td>0.597858</td>
</tr>
<tr>
<td>Bottleneck+seed+MTL(contexts)(4)</td>
<td>0.598765</td>
</tr>
<tr>
<td>Bottleneck+seed+MTL(whole-word)+DTW(5)</td>
<td>0.599388</td>
</tr>
<tr>
<td>Bottleneck+seed+MTL(contexts)+DTW(6)</td>
<td>0.604273</td>
</tr>
</tbody>
</table>

### IV. DATASET AND EXPERIMENT

#### A. Dataset

The dataset we use in this article is about 40 hours Chinese speech collected and labeled by the Electronic Engineering Department of Tsinghua University (THUEE), the speakers are from different regions, ages, and genders. Since our dataset is recorded in reality and is quite small, it is ideal for simulating the actual low-resource condition. We select about 10 hours speech data randomly as the testing set, and choose 30 words randomly as keywords, each of them has 30 to 60 examples.

#### B. F1 Measure

We use the F1 measure to evaluate the system performance. F1 is calculated based on the following formula:

\[
F_1 = \frac{2PR}{P + R}
\]

Where \( P \) represents the correct rate, which is the ratio of correct results number and total results number; \( R \) represents the recall rate, which is the ratio of correct results number and existing keywords number in the testing set.

#### C. Results of Experiments

We first focus on the MTL-bottleneck feature. In baseline system we extract the 120-dimensional Fbank feature, which contains 40-dimensional filter bank feature and its first and second derivatives. The bottleneck features are extracted by a bottleneck-DNN, the bottleneck-DNN is trained using 1000 hours Chinese and 300 hours English, and the structure of hidden layers of bottleneck-DNN is 1024-1024-1024-1024-1024-256. We use the KALDI toolkit [22] to build KWS-DNN, which has two hidden layers and each hidden layer has 512 neurons. The two systems are totally same except the input features. TABLE I shows the F1 measures of baseline and MTL-bottleneck system.

Obviously, the MTL-bottleneck system has a dramatic increase in performance over the baseline system because the bottleneck features that transfer information from large dataset significantly improved the low-resource condition.

We also apply MTL method to initialize the model. Unlike previous KWS-DNN model, which were initialized by a generative pre-training, we now initialize KWS-DNN with a seed model trained on a large dataset containing 700 hours of Chinese and 700 hours of English. Furthermore, we use multitask learning method to train the model and design two auxiliary tasks. One is whole-word recognition, the other is sub-word contexts recognition. Finally, we use the DTW post-processing method to rescore the decoding results. All experimental results are shown in TABLE II.

With the initialization, MTL and post-processing methods added, detection performance gradually increased. As for the choice of auxiliary tasks, sub-word contexts recognition (4), (6) are better than the whole-word recognition (3), (5). In the subsequent experiment, the auxiliary task is fixed as the sub-word contexts recognition.

As described in II.B, the filler model directly affects the performance of keyword spotting since it absorbs all the non-keywords. Except the basic single-state filler, we also use a multi-state filler to model non-keywords. Figure 7 shows the performance of several KWS systems in TABLE II with different number of filler states. Two states are better than single state in each system. However, the performance of the system decreases significantly if the filler model is too complicated.

Now, To evaluate our method, we use LVCSR to spot keywords in the same testing set. The structure of DNN is 1320-1024-1024-1024-1024-1024-8393, which is trained
using 1000 hours Chinese speech. The comparison of our best system and LVCSR system is shown in TABLE III. The result of LVCSR system is far worse than our best system since in the low resource condition, the dataset we use to train ASR-DNN does not match the testing data.

V. CONCLUSION

In this paper, we focus on multi-examples keyword spotting in low resource condition, applying multitask learning techniques in both feature extracting and model building. In addition, we improve detection performance using a DTW-based post-processing method. With applying MTL methods and post-processing method, we get a 34.67% absolute increase in F1 measure over the baseline system. In the aspect of filler modeling, the multi-states modeling is used to replace the original single-state modeling, and the performance of each system is improved unanimously. Furthermore, the result of our method is far better than that of LVCSR system, indicating the effectiveness of our method in low resource condition.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant No. 61370034.

REFERENCES


### TABLE III

<table>
<thead>
<tr>
<th>LVCSR system</th>
<th>Our best system</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.348074</td>
</tr>
</tbody>
</table>

{table: Measures of LVCSR System and Our Best System}