INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Deep Template Matching for Small-footprint and Configurable Keyword Spotting

Peng Zhang, Xueliang Zhang

College of Computer Science, Inner Mongolia University, Hohhot, China
zhangpeng@mail.imu.edu.cn, cszxl@imu.edu.cn

Abstract

Keyword spotting (KWS) is a very important technique for human–machine interaction to detect a trigger phrase and voice commands. In practice, a popular demand for KWS is to conveniently define the keywords by consumers or device vendors. In this paper, we propose a novel template matching approach for KWS based on end-to-end deep learning method, which utilizes an attention mechanism to match the input voice to the keyword templates in high-level feature space. The proposed approach only requires very limited voice samples (at least only one sample) to register a new keyword without any retraining. We conduct experiments on the publicly available Google speech commands dataset. The experimental results demonstrate that our method outperforms baseline methods while allowing for a flexible configuration.

Index Terms: template matching, keyword spotting

1. Introduction

Keyword spotting (KWS) is in high demand nowadays since it enables a simple voice user interface to consumer electronic devices, such as mobile phones, Amazon Echo, and Google Home. For quick responses, KWS systems always run on the device side, which has limited resources. Therefore, electronics devices require KWS systems to be computationally efficient. In addition, if a consumer or device vendor could conveniently change the keywords, it would be a popular feature for KWS systems.

As a classic solution, keyword/filler hidden Markov model (HMM) based systems [1] have been extensively applied in the KWS task. HMMs are trained for both keyword and non-keyword audio segments. Some previously conducted studies replaced the HMM by recurrent neural network models trained with a connectionist temporal classification criterion [2] or by an attention-based model [3]. With the recent advent of deep learning, a deep KWS [4] approach has trained a simple deep neural network (DNN) to predict the frame-level posteriors of sub-keyword targets and fillers. Numerous deep model-based KWS approaches [5, 6, 7, 8] considered KWS as an audio classification problem, in which each keyword is denoted as a class, and an additional filler class is defined for all the other words. Although these approaches demonstrate high performance and low computational cost, they require a large amount of training data for each pre-defined keyword and less extensibility when the keywords are a dialect or even not from a standard language. Another classic technique for KWS is by employing large vocabulary continuous speech recognition (LVCSR) systems to decode the audio into the generation of lattices, and then to search the keywords from there [9, 10]. LVCSR-based KWS systems are often flexible to change keywords based on the user's requirements, but the KWS systems cause large resource consumption.

In this work, our goal is to build a flexible and extensible KWS system that can conveniently change the keywords with low-resource consumption. The template matching method is a suitable one where only a few samples of the keywords are specified by users to match similar patterns in running speech. Dynamic time warping (DTW) [11] is typically utilized to match different lengths of audio segments. However, the DTW-based template matching has an obvious challenge because it does not have trainable parameters. This implies that the data collect from users through an online system cannot be employed to improve performance. A differentiable DTW is proposed in [12]. It can be paired with a flexible-learning architecture, such as DNNs by turning the DTW discrepancy between time-series into a full-fledged loss function.

Recently, the sequence-to-sequence (Seq2Seq) attention-based model has been widely employed in speaker verification [13], machine translation [14] and image caption [15], in which alignments are learned between source and target sequences. This motivates us to apply deep models to learn matching information between template keywords and evaluation. In this paper, we explore a novel template matching method based on end-to-end deep learning models for KWS systems, called *deep template matching* (DTM) KWS. Our proposed DTM KWS system allows for a flexible configuration and is computationally efficient.

The rest of this paper is organized as follows. We describe the DTM KWS system in Section 2. The experimental setup and evaluation results are demonstrated in Section 3. Finally, we present the conclusions in Section 4.

2. System description

As shown in Figure 1(a), the overall architecture of the DTM KWS system comprises three main modules: feature extractor, template matching, and binary classifier.

2.1. Feature Extractor

The feature extractor module is based on a Siamese network [16], which considers N consecutive frames of template, $T = [t(1), ..., t(n), ..., t(N)]$, and M consecutive frames of evaluation, $E = [e(1), ..., e(m), ..., e(M)]$ as a pair of input. The high-level features of the evaluation and template speech are denoted by h_t and h_e, respectively:

$$h_t = Net_t(t; \theta_t), \tag{1}$$

$$h_e = Net_e(e; \theta_e), \tag{2}$$

where θ_t and θ_e denote the parameters of tower models Net_t and Net_e, respectively. We apply a convolutional

This research work is supported by the National Natural Science Foundation of China (No. 61876214).

http://dx.doi.org/10.21437/Interspeech.2020-1761

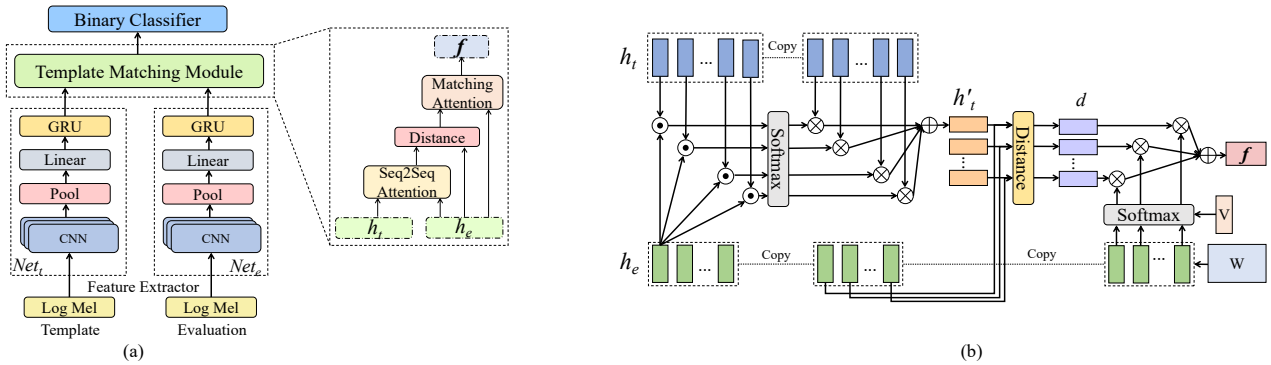

Figure 1: *(a) is a schematic of the architecture. Template Matching Module is shown in the dashed box. (b) shows a schematic diagram of the template matching mechanism.* h_t *stands for the template vector, and* h_e *stands for the evaluation vector. First, the dot product of* h_t *and* h_e *is computed. Then, the corresponding attention scores are computed with a softmax function, and summed to obtain the aligned template vector* $h_t^{'}$*. The distance vectors* d *are obtained by the* h_e *and* $h_t^{'}$*. At last, the evaluation vector* h_e*, which are weighted by the matching attention scores, are fused into the representation* f*.*

recurrent neural network (CRNN) as the feature extractor, which comprises three convolutional neural network (CNN) layers and a gated recurrent unit (GRU) layer. The configurations are presented in Table 1. The input and output sizes of each layer are specified as follows: *feature maps* × *time steps* × *frequency channels* format. The layer hyperparameters are given in different formats: *kernel size*, *strides*, and *output channels*.

Table 1: *Architecture of the CRNN feature extractor. Here T denotes the number of time frames in the spectrogram.*

Layer	Input Size	Hyperparameters	Ouput Size
reshape_1	T × 128	-	1 × T × 128
conv2d_1	1 × T × 128	5 × 5, (1, 1), 16	16 × T × 124
conv2d_2	16 × T × 124	5 × 5, (1, 1), 32	32 × T × 120
conv2d_3	32 × T × 120	5 × 5, (1, 1), 64	64 × T × 116
max_pool	64 × T × 116	5 × 2, (1, 2)	64 × T × 58
reshape_2	64 × T × 58	-	T × 3712
linear	T × 3712	64	T × 64
gru	T × 64	64	T × 64

2.2. Template Matching Method

Instead of the traditional template matching method, e.g., DTW [11], which directly computes the best possible alignment between the evaluation and the available template features by dynamic programming, we employ Seq2Seq attention [13] to first align the two time-series features. Particularly, each frame of the evaluation is aligned to a weighted sum of all frames of the template that can be viewed as a soft alignment. Then, a distant vector, $d(m)$, is calculated to measure the difference between the evaluation, $h_e(m)$, and the aligned template, $h_t^{'}(m)$, on each frame. Finally, we employ a mechanism, called matching attention, on the time-series of the distant vector, $d(m)$, to generate a single fixed-dimensional vector, f. Figure 1(b) depicts a schematic diagram of the template matching mechanism.

In particular, the similarity score of the evaluation vector, h_e, at frame m, and the template vector, h_t, at frame n, is calculated by eq. 3.

$$s(m,n) = h_e^T(m) h_t(n). \tag{3}$$

We normalize attention weight, α_e, by eq. 4.

$$\alpha_e(m,n) = \frac{exp(s(m,n))}{\sum_{n=1}^{N} exp(s(m,n))}. \tag{4}$$

Then, the template vectors, h_t, at each frame are weighted with the normalized alignment weight, $\alpha_e(m,n)$, and summed to obtain the aligned template vector, $h_t^{'}(m)$, at each frame m by eq. 5.

$$h_t^{'}(m) = \sum_{n=1}^{N} \alpha_e(m,n) h_t(n). \tag{5}$$

We calculate the discrepancy between the evaluation vector, $h_e(m)$, and the aligned template, $h_t^{'}(m)$, on each frame using absolute distance:

$$d(m) = \left| h_t^{'}(m) - h_e(m) \right|. \tag{6}$$

Instead of directly feeding the distant vector into the classifier, we use an attention layer to match the period of the keyword in the evaluation to be spotted. We first calculate the scalar score, q_e, for the evaluation vector, h_e, at frame:

$$q_e(m) = V^T tanh(W h_e(m) + b), \tag{7}$$

where W, V and b are parameters to learn. The normalized weight, $\beta_e \in [0, 1]$, can be achieved by applying a softmax function as follows:

$$\beta_e(m) = \frac{exp(q_e(m))}{\sum_{i=1}^{M} exp(q_e(i))}. \tag{8}$$

This has been extensively utilized in numerous existing neural attention models [17, 18]. Finally, we form a single fixed-dimensional vector, f, as the weighted sum of the evaluation vector at all the frames:

$$f = \sum_{m=1}^{M} \beta_e(m) d(m). \tag{9}$$

2.3. Binary Classifier

After obtaining the fixed-dimensional vector, f, through the template matching module, we use a binary classifier with cross-entropy loss to make the final decision for the input pair. The classifier is a fully connected network with one hidden layer that has 128 hidden units. The activation function is a rectified linear unit (ReLU). The output of the classifier is a 2-dimensional vector, z, indicating whether the input pair belongs to the same keyword.

3. Experiments

3.1. Dataset

We conduct our experiments on a public dataset: the Google speech commands dataset [19]. It comprises 105,829 one-second (or less) long audio clips of 35 keywords that are collected from thousands of people. Each clip contains only one keyword. In 35 categories, we employ 24 core words as the training commands. By pairing clips from the same keywords as positive samples and those from different keywords as negative samples, a total number of training sets consisting of 83k positive and 83k negative samples are utilized. For the test set, we randomly select three samples for a keyword as the templates. We build two different test sets of keywords. One set contains 24 keywords that are used during the training of the models, i.e., *trained keywords set*, whereas the other set contains the remaining 11 untrained keywords, i.e., *untrained keywords set*. The aim is to assess the robustness of our system regarding the new keywords, even when spoken by speakers that are not in the *trained keywords set*. For each test keyword, the number of positive pairs is equal to that of the negative pairs. A total of 42k target and imposter samples of test sets are employed.

3.2. Experimental Setup

We utilize 128-band log-mel spectrogram with a 25-ms analysis window and 10ms overlap, and we obtain a dimensionality of 128 frequency bins of 99 contiguous time frames.

Training. We employ stochastic gradient descent (SGD) with a momentum 0.9, weight-decay 1e-5, and initial learning rate of 1e-1, which is multiplied by 0.1 on plateaus. The batch size is 256 and the training procedure is up to 200 epochs with early stopping.

Evaluation. Results are reported in the form of a modified version of receiver operating characteristic (ROC) curves [4], lower curves are better. False alarm and false rejection counts are collected from all the models to compute the false rejection rate at a certain false alarm rate. The curve is obtained by sweeping the decision threshold. Accuracy is another metric of quality, which is simply measured as the fraction of binary classification decisions that are correct.

3.3. Baseline

We compare the proposed method with several baseline KWS approaches.

(1) ***ResNet-15***. We reimplement `ResNet-15` (which is state-of-the-art) [7] to compare the KWS performance on the *trained keywords set*, and with 128-dim log-mel spectrogram as input features.

(2) ***DTW***. A DTW-based KWS [11] is involved in the comparison. It comprises two steps: first, acoustic features are extracted at the frame level; second, DTW is performed to compare the feature matrix of the evaluation and the template. Then, a confidence score is computed from the DTW alignment cost.

(3) ***soft-DTW***. A recently proposed approach, soft-DTW [12], takes the advantage of a smoothed formulation of DTW, which computes the soft minimum of all alignment costs and can be used to train neural networks. We utilize soft-DTW as the loss function to directly optimize our feature extractor. Additionally, inspired by the tuple-based end-to-end loss [18], we redefine the binary classifier based on the soft-DTW loss function:

$$Loss = \delta(e,t)\sigma(c) + (1-\delta(e,t))(1-\sigma(c)), \qquad (10)$$

where $\sigma(x) = 1/(1+e^{-x})$ denotes the standard sigmoid function that normalizes the cost calculated by soft-DTW. $\delta(e,t)$ is equal to 1 if the evaluation and the template are the same keywords; otherwise, it is equal to 0. The end-to-end loss function encourages a smaller alignment cost, c, when $e = t$, and a larger alignment cost, c, when $e \neq t$.

(4) ***ASR***. We perform a decoding search on the *untrained keywords set* to obtain a phoneme-based lattice based on a ASR model. Then, through Viterbi algorithm to matching the phoneme sequences on the lattice. The ASR model is based on Kaldi framework with a lattice-free Maximum Mutual Information [20] model, which trains on 100-hour clean Librispeech corpus [21].

4. Results

4.1. The impact of different distance methods

We investigate the impact of different distance methods in the template matching mechanism on the *untrained keywords set*. As shown in Figure 3, it can be seen that the absolute distance achieves the best performance. Please note that all our subsequent extensions and comparison experiments are based on absolute distance.

4.2. The effectiveness of the template matching mechanism

Table 2 shows the performance comparison of our complete template matching method in ablating certain components on the *untrained keywords set*. We employ the absolute distance that performed the best according to our experiments. We can find that both of the Seq2Seq and the matching attention are important. Without Seq2Seq attention, ACC drops from 92.77% to 90.20%. Without matching attention, ACC drops to 87.90%. When both of them remove, ACC drops to 85.81%.

Table 2: *The effectiveness of template matching mechanism.*

Methods	ACC(%)
W/O Seq2Seq-Atten	90.20
W/O Matching-Atten	87.90
W/O Atten	85.81
Full Model	92.77

4.3. A comparison with other models

Figure 4 shows the performance of the proposed DTM model, `ResNet-15` model and 2 DTW-based KWS models on the *trained keywords*. For the DTW KWS models, one is DTW model which is directly to calculate the alignment score between the acoustic features, another is Soft-DTW model which

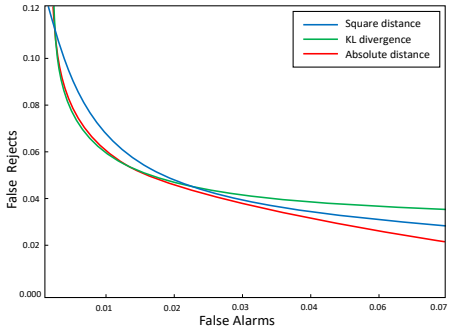

Figure 2: *ROCs for different distance methods with DTM models.*

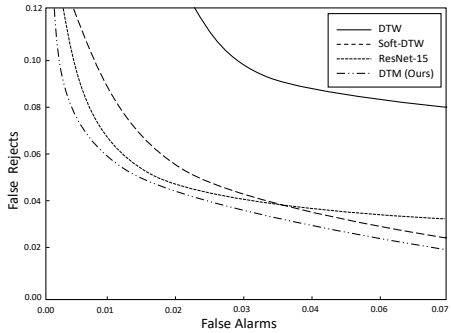

Figure 3: *ROCs for different baseline and DTM models on the trained keywords set.*

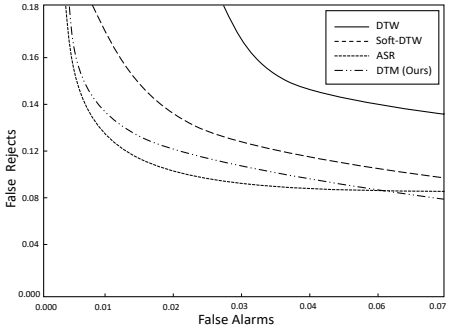

Figure 4: *ROCs for different baseline and DTM models on the untrained keywords set.*

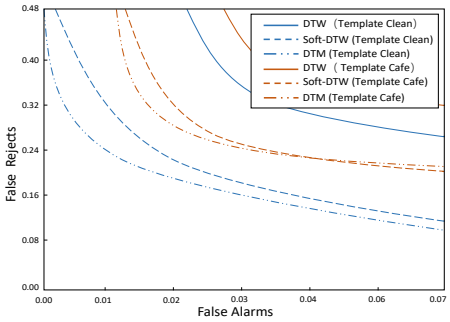

Figure 5: *ROCs for different baseline and DTM models on the noisy template.*

is obtaining alignment score from CRNN as described in Section 2.1. In Figure 4, our proposed DTM model improves significantly over both DTW-based models and `ResNet-15` model on the *trained keywords*. Additionally, the results demonstrate that the differentiable DTW with learning architecture obtains a relative improvement.

As shown in Figure 5, we compare the performance of the proposed DTM model, ASR model and two DTW-based KWS models on the *untrained keywords set*. We observe that our proposed DTM model is comparable with the ASR model. Differently than the ASR model, our proposed DTM model has not been specifically trained on any other dataset. We also note that the soft-DTW model outperforms the DTW model, it indicates that the differentiable DTW with learning architecture also has a generalization ability for *untrained keywords set*.

The accuracies of these models and the number of model parameters as shown in Table 3. The ASR model has best performance on the *untrained keywords set*, but the amount of parameters restricts it from being able to be directly deployed on small devices. We can observe that our proposed DTM model has good performance in both *trained keywords* and *untrained keywords set*, and which is greatly simpler and lighter which would make it more feasible to deploy on a small device.

Table 3: *Test accuracy of each model and footprint in terms of number of parameters.*

Test set	Model	Params	ACC(%)
trained keywords	DTW	-	61.17
	Soft-DTW	190K	94.90
	ResNet-15 [7]	238K	95.80
	DTM(Ours)	190K	**97.91**
untrained keywords set	DTW	-	60.36
	Soft-DTW	190K	85.52
	ASR	9.9M	**94.42**
	DTM(Ours)	190K	92.77

4.4. The impact of noisy template

In Figure 6, we show the robustness of our proposed DTM model when adding different noise source at template and evaluation time on the *untrained keywords set*. To simulate this, we select babble noise and cafe noise. We add 10dB of babble noise to the clean evaluation data, and also add a different 10 dB cafe noise to the clean template data. The performance degrades dramatically for all the systems. We also observe that in a noisy environment, the proposed DTM model is still better than DTW-based models.

5. Conclusions

This study propose an approach of template matching for a KWS system using end-to-end deep learning with Seq2Seq attention mechanism. Experimental results show that our proposed method outperforms DTW-based models and more efficient than an ASR-based system, and no language model and no computationally expensive graph beam searches are required. In this approach, the KWS system is more flexible that only a few keyword samples are specified by the user to match similar patterns in the running speech. Based on the deep template matching method, our future study will explore a more light KWS system and investigate a more robustness KWS system against environmental noise.

6. References

[1] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden markov modeling for speaker-independent word spotting," in *ICASSP*, 1989, pp. 627–630.

[2] K. Hwang, M. Lee, and W. Sung, "Online keyword spotting with a character-level recurrent neural network," *arXiv preprint arXiv:1512.08903*, 2015.

[3] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *ASRU*, 2017, pp. 474–481.

[4] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *ICASSP*, 2014, pp. 4087–4091.

[5] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015, pp. 1478–1482.

[6] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *arXiv preprint arXiv:1703.05390*, 2017.

[7] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *ICASSP*, 2018, pp. 5484–5488.

[8] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, "A time delay neural network with shared weight self-attention for small-footprint keyword spotting," in *Interspeech*, 2019, pp. 2190–2194.

[9] P. Motlicek, F. Valente, and I. Szoke, "Improving acoustic based keyword spotting using lvcsr lattices," in *ICASSP*, 2012, pp. 4413–4416.

[10] D. Can and M. Saraclar, "Lattice indexing for spoken term detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[11] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[12] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *ICML*, 2017, pp. 894–903.

[13] Y. Zhang, M. Yu, N. Li, C. Yu, J. Cui, and D. Yu, "Seq2seq attentional siamese neural networks for text-dependent speaker verification," in *ICASSP*, 2019, pp. 6131–6135.

[14] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015, pp. 3156–3164.

[16] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV*, 2016, pp. 850–865.

[17] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Interspeech*, 2018, pp. 2037–2041.

[18] F. R. rahman Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," in *ICASSP*, 2018, pp. 5359–5363.

[19] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[20] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi." in *Interspeech*, 2016, pp. 2751–2755.

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP*, 2015, pp. 5206–5210.