VERIFYING DEEP KEYWORD SPOTTING DETECTION WITH ACOUSTIC WORD EMBEDDINGS

Yougen Yuan[1⋆], *Zhiqiang Lv*[2], *Shen Huang*[2], *Lei Xie*[1†]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[2]Tencent Research, Beijing, China

ABSTRACT

In this paper, in order to improve keyword spotting (KWS) performance in a live broadcast scenario, we propose to use a template matching method based on acoustic word embeddings (AWE) as the second stage to verify the detection from the Deep KWS system. AWEs are obtained via a deep bidirectional long short-term memory (BLSTM) network trained using limited positive and negative keyword candidates, which aims to encode variable-length keyword candidates into fixed-dimensional vectors with reasonable discriminative ability. Learning AWEs takes a combination of three specifically-designed losses: the triplet and reversed triplet losses try to keep same keyword candidates closer and different keyword candidates farther, while the hinge loss is to set a fixed threshold to distinguish all positive and negative keyword candidates. During keyword verification, calibration scores are used to reduce the bias between different templates for different keyword candidates. Experiments show that adding AWE-based keyword verification to Deep KWS achieves 5.6% relative accuracy improvement; the hinge loss brings additional 5.5% relative gain and the final accuracy climbs to 0.775 by using calibration scores.

Index Terms— Query-by-example, keyword spotting, acoustic word embeddings, hinge loss, calibration scores

1. INTRODUCTION

Keyword spotting (KWS) is the task of detecting predefined keywords in audio signals. In practical industrial applications, as there are billions of online requests on a large volume of real-world live audio streams, we are particularly interested in computation-constrained low-latency KWS solutions. Many previous KWS approaches use a large vocabulary continuous speech recognition (LVCSR) system to decode the audio signals and their efforts mainly focus on efficient keyword indexing and accurate search on the lattices or confusion networks [1, 2, 3, 4, 5, 6, 7]. These approaches require high computational resources so that they are not suitable for low-latency KWS applications. The keyword/filler approach [8, 9, 10, 11] is relatively lightweight, where hidden Markov models (HMM) are trained for keywords and non-keyword fillers respectively and Viterbi decoding is used at runtime, which is still computationally expensive.

In recent years, following the keyword/filler approaches, a more lightweight KWS approach based on deep neural networks (DNN) named *Deep KWS* has been proposed [12, 13, 14, 15, 16], in which a single DNN is trained to predict the posteriors of keyword/sub-keyword units and a simple post-processing module is used to produce confidence scores. Without HMMs involved, the Deep KWS approach has the advantage of low-latency and small-footprint [12]. Thus, it has become a highly competitive solution in practical industrial KWS application. However, when Deep KWS comes to live broadcast scenarios where there are heavy speaker accents and various types of background music and noise, the detection errors will increase rapidly. Hence it is still plenty of space to improve the performance of a Deep KWS system.

In previous HMM-based keyword/filler approaches, a two-stage framework has been proposed for KWS with considerable performance improvement and ignorable computation and latency [17, 18]. The first stage works on audio streams and extracts audio segments hypothesized to contain pre-specified keywords, which is targeted to high recall. Given the hypothesized keyword segments from the first stage, a second-stage classifier can be effectively cascaded to filter out false alarms of hypothesized candidates. Current two-stage KWS methods mainly focus on feature engineering for the second stage classifier, including support vector machine (SVM) [17] and feed-forward DNN [18]. The classifiers mainly target to single wakeup word detection for smart devices on the edge side, while our KWS scenario has to handle online search of multiple keywords in a live broadcast scenario.

In this paper, we borrow the two-stage idea to cascade a light second-stage keyword verification module to the Deep KWS system. Specifically, we propose to use a template matching method based on acoustic word embeddings (AWE)

⋆ The author performed the work as an intern at Tencent Research.

† Corresponding author.

This research work is supported by the 2018 Tencent Rhino-Bird Elite Training Program and the National Natural Science Foundation of China (No.61571363).

as the second stage to verify the detections from Deep KWS. In simple words, if a Deep KWS detection is similar enough to the corresponding keyword templates, we consider it as a correct spotted keyword. AWEs have been successfully used in query-by-example (QbE) spoken term detection [19, 20, 21]. They aim to encode variable-length speech segments into fixed-dimensional compact vectors with reasonable discriminative ability, such that the distance between embeddings of different keyword candidates is larger than those of the same keyword candidates.

In our approach, AWEs are learned by deep bidirectional long short-term memory (BLSTM) networks with three specifically-designed loss functions: a triplet loss, a reversed triplet loss and a hinge loss. The first two losses try to keep same hypothesized detections closer and different hypothesized detections farther. As the two triplet losses just learn the relative distance between hypothesized detections, it is difficult to apply a single threshold for all hypothesized detections. To solve this problem, we introduce a hinge loss [22] to set a proper fixed threshold to distinguish all positive and negative hypothesized detections. The fixed threshold is larger/smaller than the distance between same/different hypothesized detections respectively. During the verification stage, similarity scores can be measured over AWEs between the hypothesized detections and the corresponding keyword templates. As there exists unavoidable bias between different keyword templates for different hypothesized detections, calibration scores are further added to the similarity scores for reducing such bias. Experiments on a live broadcast scenario show that adding AWE-based keyword verification to Deep KWS achieves 6.7% relative improvement in accuracy. The introduction of the hinge loss brings additional 5.5% relative improvement and the final accuracy climbs to 0.775 by using calibration scores.

2. METHOD

2.1. System overview

Fig. 1 shows the overall framework of our two-stage cascaded KWS. We set a Deep KWS system (the steps in black in Fig. 1) as the first stage, and focus on using a second-stage AWE-based keyword verification (the steps in red in Fig. 1) for improving Deep KWS detection. In the training process, a Deep KWS is trained using the transcribed speech, and an AWE-based keyword verification is also trained using the collected positive and negative keyword examples. In the testing process, the trained Deep KWS is firstly used to obtain a list of hypothesized detections with start and end time in the audio streams. Next, instead of producing the final results directly, the hypothesized detections are pushed to the second-stage keyword verification to obtain AWEs. Finally, the corresponding AWEs between the hypothesized detections and the keyword templates are compared to determine the final results.

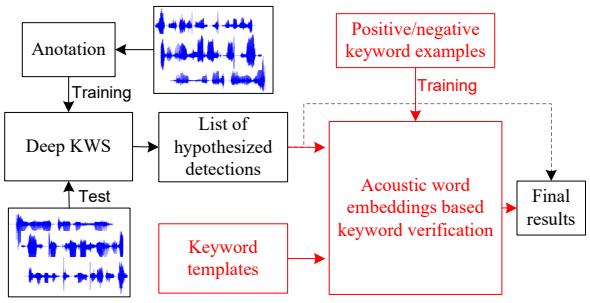

Fig. 1. The overall framework of two-stage cascaded KWS. A second-stage AWE-based keyword verification (the steps in red) is proposed to improve Deep KWS detection.

Before describing AWE-based keyword verification in details, we briefly introduce the system building of Deep KWS. The Deep KWS is originally proposed in [12], which can be divided into three processes, including feature extraction, DNN-based acoustic modeling and posterior handling. First of all, a voice activity detection (VAD) algorithm is performed to remove non-speech frames, and filterbank (Fbank) features are extracted to represent the area of speech frames. Then, a DNN model is trained using the Fbank features. The last layer of DNN has a softmax function which outputs an estimate of the posteriors of each output label. Finally, suppose p_{ik} is the DNN posterior for the i^{th} label and the k^{th} frame x_k, the confidence score at the j^{th} frame is computed as follows:

$$confidence\ score_j = \sqrt[m-1]{\prod_{i=1}^{m-1} \max_{h_{max} \le k \le j} p_{ik}^{smooth}} \quad (1)$$

where m represents the number of total labels. p_{ik}^{smooth} is the smoothed state posterior of p_{ik}, and $h_{max} = \max\{1, j - w_{max} + 1\}$ is the index of the first frame within the fixed-length sliding window.

2.2. Loss functions in AWE-based keyword verification

After Deep KWS as the first stage is triggered, we use a template matching method based on AWEs as the second stage to verify the hypothesized detections. AWEs [23] aim to encode speech segments with variable-length into fixed-dimensional discriminative representations. The key is to effectively aggregate frame-level information into a segment-level compact representation vector, where the distance between vectors of different words is larger than those of the same words.

In the proposed AWE-based keyword verification, a deep BLSTM network is used to do the feature aggregation across time. As shown in Fig. 2, the network takes triplets (x_a, x_p, x_n) as input. We use a pair of the same hypothesized detections as an anchor example x_a and

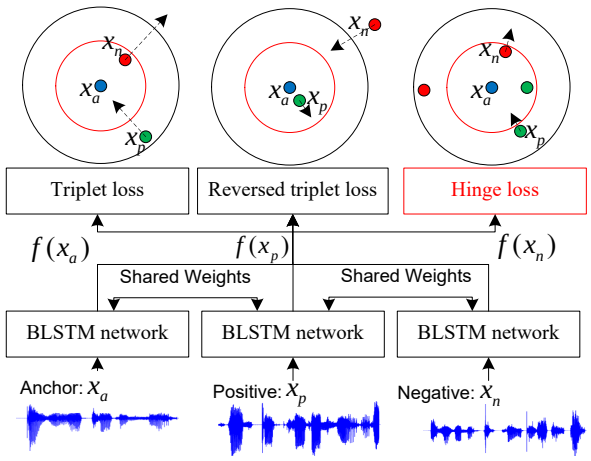

Fig. 2. The diagram of learning AWE-based keyword verification, where the hinge loss (red box) is used to set a fixed threshold to distinguish all positive examples (green dots) and negative examples (red dots).

a positive example x_p. Then, a misidentified keyword candidate detected from the first-stage Deep KWS is selected as our negative example x_n. With the triplets (x_a, x_p, x_n), AWEs $(f(x_a), f(x_p), f(x_n))$ are obtained via forwarding the BLSTM network.

An efficient objective function is very important for the training of neural networks. In order to obtain AWEs with discrimination and robustness for QbE speech search, we train the BLSTM network with three specifically-designed loss items. One loss function is a triplet loss that is adapted from the originally proposed one for face recognition [24]. The adapted triplet loss aims to decrease the cosine distance between the embeddings $(f(x_a), f(x_p))$ and increase the cosine distance between the embeddings $(f(x_a), f(x_n))$, which is defined as follows:

$$TL(x_a, x_p, x_n) = \max\{0, d_+ - d_- + \delta_1\} \quad (2)$$

where δ_1 is a margin constraint that reduces the gap between the cosine distance between same hypothesized detections $d_+ = \cos(f(x_a), f(x_p))$ and the cosine distance between different hypothesized detections $d_- = \cos(f(x_a), f(x_n))$. The cosine distance between two embeddings $(f(x_1), f(x_2))$ can be calculated by:

$$\cos(f(x_1), f(x_2)) = \frac{1 - \frac{f(x_1) * f(x_2)}{\|f(x_1)\|_2 \|f(x_2)\|_2}}{2} \quad (3)$$

In our case, one triplet belongs to the same hypothesized detections from Deep KWS. The main difference is that the anchor and positive examples are true positive while the negative example is a false positive. Therefore, the triplet still has a certain similarity at acoustic level, and the triplet loss is unreasonable to make the similarity between different hypothesized detections small enough. To address this problem, another loss function—reversed triplet loss—is added as a trade-off to limit the distance gap between d_- and d_+ not be too large. The reversed triplet loss is specifically defined as follows:

$$RTL(x_a, x_p, x_n) = \max\{0, d_- - d_+ - \delta_2\} \quad (4)$$

where δ_2 is also a margin constraint.

Besides both triplet losses, an extra hinge loss (the red box in Fig. 2) is added in learning AWE-based keyword verification. The hinge loss aims to set a fixed threshold to distinguish all positive and negative examples [22]. It makes the cosine distance between the embeddings $(f(x_a), f(x_p))$ smaller than the threshold, while the cosine distance between the embeddings $(f(x_a), f(x_n))$ larger than the threshold. Our hinge loss is specifically defined as follows:

$$HL(x_a, x_p, x_n) = \max\{0, -\theta + d_+\} + \max\{0, \theta - d_-\} \quad (5)$$

where θ denotes a fixed threshold to distinguish the same hypothesized detections (x_a, x_p) and the different hypothesized detections (x_a, x_n).

By joint-training these three specifically-designed loss functions, the total objective function L can be calculated as follows:

$$L(x_a, x_p, x_n) = \alpha TL + \beta RTL + \gamma HL \quad (6)$$

where $\alpha/\beta/\gamma$ are coefficients of the triplet loss TL in Equation 2, the reversed triplet loss RTL in Equation 4, and the hinge loss HL in Equation 5 respectively. With the BLSTM network trained using the total objective function L, AWE-based keyword verification can be applied to all hypothesized detections.

2.3. Calibration scores during keyword verification

A template matching method based on AWEs is used to verify the hypothesized detections from Deep KWS. Fig. 3 shows the diagram of AWE-based template matching method. The method consists of three modules, including generating AWEs using BLSTM network, scoring and decision. First of all, for the k_{th} hypothesized detection x_k, the AWEs $f(x_k)$ are generated via the trained deep BLSTM network. Then n keyword examples of the same label with the hypothesized detection x_k are selected as the corresponding keyword templates $[y_{k1}, ..., y_{ki}, ..., y_{kn}]$. The keyword templates are converted into AWEs $[f(y_{k1}), ..., f(y_{ki}), ..., f(y_{kn})]$ via the trained deep BLSTM network.

Next, a scoring process is applied to provide final scores between the hypothesized detection and the keyword templates over their corresponding AWEs. More specifically, a similarity score S_{ki} is used to measure the cosine similarity between the AWEs of hypothesized detection $f(x_k)$ and the

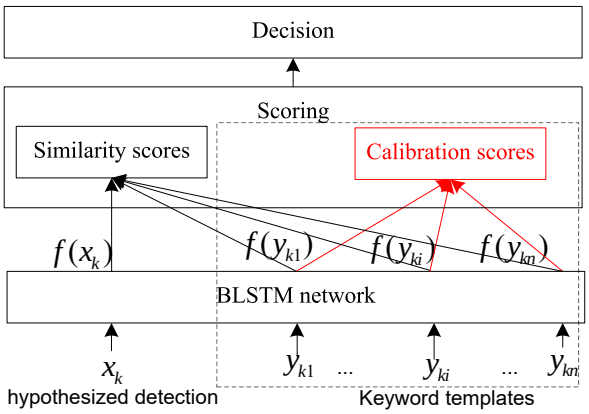

Fig. 3. The diagram of AWE-based template matching for keyword verification, where the calculation of calibration scores are highlighted in red.

AWEs of i_{th} keyword template $f(y_{ki})$, which is calculated by:

$$S_{ki}(f(x_k), f(y_{ki})) = \cos(f(x_k), f(y_{ki})), i = 1, 2, ..., n. \tag{7}$$

All these similarity scores can be merged into one final score for decision. Note that for the hypothesized detections of different keywords, the corresponding templates are different. Hence there is considerable difference in the resulting overall scores.

To mitigate such difference, calibration scores are added to the similarity scores for final decision. The use of calibration scores compensates for the fact that some words are more variable in pronunciation. Given a hypothesized detection x_k, the calibration score CS_k can be obtained by averaging the cosine distance between all the keyword templates $[f(y_{k1}), ..., f(y_{ki}), ..., f(y_{kj}), ..., f(y_{kn})]$. It is calculated as follows:

$$CS_k = \frac{2}{n(n-1)} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \cos(f(y_{ki}), f(y_{kj})) \tag{8}$$

Note that in order to accelerate the verification process, the corresponding AWEs of the templates and the calibration score CS_k (the black dashed box in Fig. 3) can be calculated in advance. With the calibration score CS_k, the final scores are calculted by:

$$FC_{ki} = S_{ki} - CS_k - bias, i = 1, 2, ..., n. \tag{9}$$

where $bias$ is a constant bias for all hypothesized detections.

Finally, a decision process is used to provide a final decision. For the AWEs of hypothesized detection $f(x_k)$ and the AWEs of i_{th} keyword template $f(y_{ki})$, the decision D_{ki} is calculated by:

$$D_{ki} = \begin{cases} 1 & FC_{ki} <= 0, i = 1, 2, ..., n. \\ 0 & FC_{ki} > 0, i = 1, 2, ..., n. \end{cases} \tag{10}$$

Then, all the decisions are merged into one final decision by:

$$D(x_k) = \begin{cases} 1 & \frac{1}{n} \sum_{i=1}^{n} D_{ki} >= 0.5 \\ 0 & \frac{1}{n} \sum_{i=1}^{n} D_{ki} < 0.5 \end{cases} \tag{11}$$

where $D(x_k) = 1$ represents the hypothesized detection x_k is correct.

3. EXPERIMENTS

3.1. Experimental setup

To evaluate the effectiveness of our proposed AWE-based keyword verification approach, experiments are conducted on the KWS data collected from the Tencent AI Open Platform [1]. The collected data is in a live broadcast scenario with heavy speaker accent and various types of background music and noise. A set of 42 Chinese words is used as keywords. Each keyword is represented by a single target label in the first stage Deep KWS system. The speech detections are about 15.4k/6.8k, 2.1k/0.9k, 7.4k/3.4k positive/negative keyword examples as the training, development and test set, respectively. They are obtained from the Deep KWS system in the first stage. Each keyword examples has the durations from 0.2 to 1.5 seconds. The training set can make up 42.0k triplets for neural network training. Each triplet is represented by 43-dimensional Mel-frequency cepstral coefficients (MFCCs) features to learn AWEs. During the keyword verification stage, 100 keyword templates are used ($n = 100$ in Equation 7) to verify each speech detection in the test set.

The Deep KWS system [12] is reimplemented as the baseline, in which the network predicts the posteriors of keyword units and a filler. The "filler" represents the speech that does not contain any keywords. Specifically, the Deep KWS system is trained in a live broadcast Chinese speech corpus that consists of 3790 hours of manually transcribed utterances. The DNN model consists of 5 hidden layers and 512 hidden nodes per layer with rectified linear unit (ReLU) activation function. An input window with 10 left frames and 5 right frames is used. All the speech frames are represented by 40-dimensional Fbank features. The length of sliding window w_{max} is set to 100 in the Deep KWS system.

In AWE-based keyword verification, AWEs are learned via a deep BLSTM network that consists of 2 BLSTM layers with 256 hidden units per direction. The BLSTM network runs independently on the hypothesized detections of deep KWS system. AWEs are obtained by concatenating the last states of the BLSTM output. Our AWE-based keyword verification has roughly the same network parameters and computation costs as the first stage Deep KWS. The margin

[1] https://ai.qq.com/

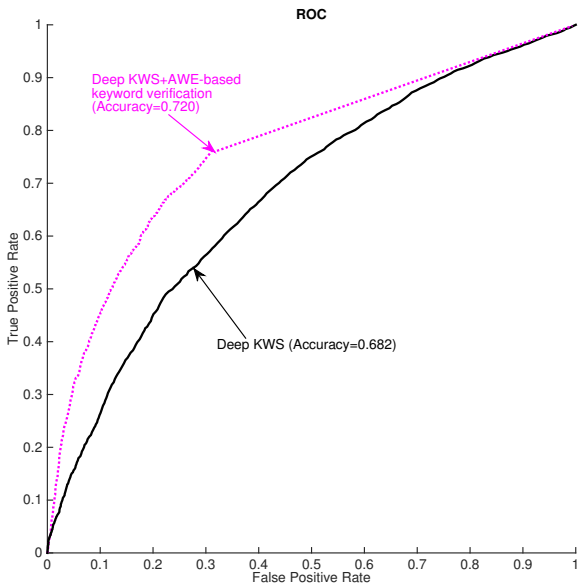

Fig. 4. ROC comparison of two systems: Deep KWS and Deep KWS+AWE-based keyword verification.

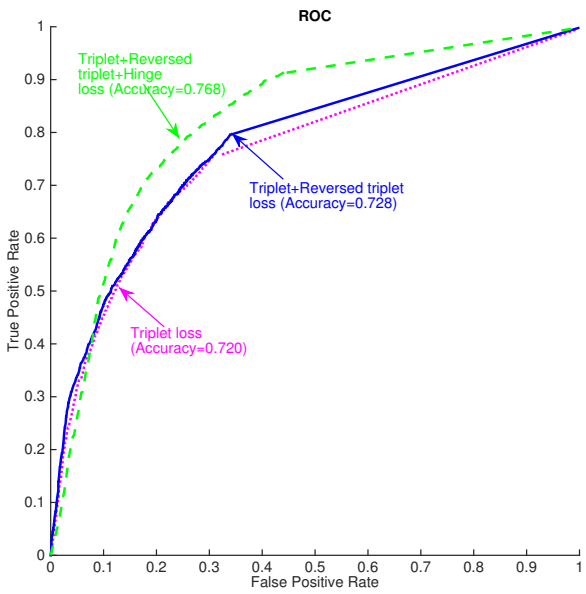

Fig. 5. ROC comparison of AWE-based keyword verification using different composition of loss functions.

constraint δ_1 is set to 0.15 in triplet loss, and another margin constraint δ_2 is set to 0.6 in reversed triplet loss. The fixed threshold θ is set to 0.3. In total objective function, $\alpha/\beta/\gamma$ are set to 0.3/0.3/0.4 in Equation 6 respectively. The weights of deep BLSTM network are initialized from -0.05 to 0.05. An Adam optimizer [25] is used for updating the weights with the mini-batch size of 100 and the learning rate of 0.0001. During the training of AWE-based keyword verification, the network and the best constant bias $bias$ are fine-tuned on the development set every five epochs. The model of AWE-based keyword verification is implemented using the Tensorflow toolkit [26].

The performance of AWE-based keyword verification is evaluated by three different metrics: 1) true positive rate (TPR), which is the rate of true predictions in all positive examples; 2) false positive rate (FPR), which is the rate of false predictions in all negative examples; 3) Accuracy, which is the rate of correct predictions in all examples at the best fixed threshold. Lower FPR, higher TPR and accuracy represent better performance.

3.2. Performance of AWE-based keyword verification

Experiments are carried out to compare the performance of two KWS system. The one is the Deep KWS baseline. The other is our proposed two-stage cascaded KWS that uses a second-stage AWE-based keyword verification on Deep KWS detection from the first stage. The AWE-based keyword verification model is only trained with the triplet loss. The comparison results are plotted by receiver operating curves (ROC). As shown in Fig. 4, we can find that adding an AWE-based keyword verification achieves consistently better performance on the keyword detection stage with 5.6% relative improvement in accuracy. It can effectively improve TPR and reduce FPR. This result indicates that our proposed two-stage cascaded KWS is effective. The AWE-based keyword verification approach has an obvious effect on improving hypothesized detections.

3.3. Effect of using extra hinge loss

To demonstrate the effectiveness of extra hinge loss, a series of comparison experiments are conducted in learning AWE-based keyword verification with different composition of loss functions, including TL, $RTL + TL$ and $HL + RTL + TL$, where HL, RTL, TL represent triplet loss, reversed triplet loss and hinge loss, respectively. As shown in Fig. 5, we find that the composition of adding hinge loss (the green curve in Fig. 5) obviously performs best among these three compositions in the AWE-based keyword verification. More specifically, with the hinge loss, the performance of AWE-based keyword verification achieves 5.5% relative improvement in accuracy. This result shows the superiority of using the hinge loss in AWE-based keyword verification.

In addition, the performance of adding reversed triplet loss RTL to the triplet loss TL achieves a small relative improvement of 1.1% in accuracy. We also find that only using the hinge loss cannot learn effective AWEs. Here the triplet losses and the hinge loss are complementary to each other in AWE-based keyword verification. The hinge loss aims to penalize misclassified hypothesized detections, thus

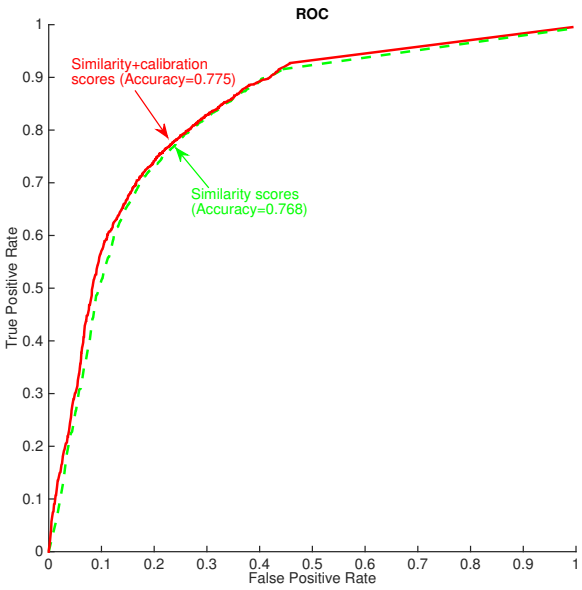

Fig. 6. ROC comparison of two different scoring methods during keyword verification: only using similarity scores and adding calibration scores on similarity scores.

enhancing the robustness of AWE-based keyword verification.

3.4. Effect of extra calibration scores

To investigate the effect of the extra calibration scores during keyword verification, a comparison experiment is carried out on two different scoring methods, including only using similarity scores and adding calibration scores on similarity scores. Notice that both scoring methods use the same trained deep BLSTM network to generate AWEs. Their only difference is that whether using calibration scores in scoring. As shown in Fig. 6, we can find that adding calibration scores achieves a small performance gain in terms of both TPR and FPR. The relative improvement in accuracy is 0.9%. This result demonstrates that there exists unavoidable bias of using different keyword templates, and the calibration scores can reduce a little bit of bias between different templates for different hypothesized detections.

3.5. Accuracy in different confidence scores

To further analyze the improvement, the accuracy is tested on both Deep KWS and our best two-stage cascaded KWS under different confidence scores. The two-stage cascaded KWS refers to the second-stage AWE-based keyword verification on Deep KWS detection from the first stage. All these confidence scores belong to the corresponding hypothesized detections produced by the first-stage Deep KWS. Before evaluation, we discard hypothesized detections with the confidence scores less than 70, and thus all the remaining

Table 1. Accuracy comparison of two systems in different confidence scores.

Confidence scores by Deep KWS	Percentage of hypothesized detections	Accuracy	
		Deep KWS	Two stages cascaded KWS
(70,80]	48.3%	0.562	**0.785**
(80,90]	41.1%	0.772	**0.775**
(90,100]	10.6%	**0.878**	0.730
Total	100%	0.682	**0.775**

hypothesized detections have the confidence scores from 70 to 100. As listed in Table 1, the confidence score is relatively low for a majority of Deep KWS detections in the live broadcast scenario. When the confidence score is less than 80, the AWE-based keyword verification achieves significant improvement on hypothesized detections. This result suggests that the hypothesized detections have a large confusion in relatively low confidence score conditions, and the proposed keyword verification is effective to distinguish these hypothesized detections. When the confidence score is in $(80, 90]$, the accuracy of AWE-based keyword verification is a little better than that of deep KWS. When the confidence scores continue to increase ($>$90), the accuracy of deep KWS is better than that of AWE-based keyword verification. As a whole, the Deep KWS has a good performance in high confidence scores, while our proposed AWE-based keyword verification has obvious superiority in relatively low confidence scores. Therefore, our proposed AWE-based keyword verification is effective to complement with the Deep KWS system. We can appropriately decrease the confidence score to improve recall from Deep KWS, and then use the verification model to improve accuracy for the hypothesized detections.

4. CONCLUSION

We have proposed a template matching method based on AWEs as the second stage to verify Deep KWS detection in a live broadcast scenario. The AWE-based keyword verification achieves a significant accuracy improvement on the hypothesized detections for the first Deep KWS stage, especially in relatively low confidence score conditions. With an extra hinge loss besides two triplet losses, we can obtain more effective AWEs that bring further accuracy improvement. During the keyword verification stage, we also demonstrate the effectiveness of using calibration scores.

5. REFERENCES

[1] David RH Miller, Michael Kleber, Chia-Lin Kao, Owen Kimball, Thomas Colthurst, Stephen A Lowe, Richard M Schwartz, and Herbert Gish, "Rapid and accurate spoken term detection," in *Proc. INTERSPEECH*, 2007, pp. 314–317.

[2] Jonathan Mamou, Bhuvana Ramabhadran, and Olivier Siohan, "Vocabulary independent spoken term detection," in *Proc. SIGIR*, 2007, pp. 615–622.

[3] Siddika Parlak and Murat Saraclar, "Spoken term detection for Turkish broadcast news," in *Proc. ICASSP*, 2008, pp. 5244–5247.

[4] Guoguo Chen, Oguz Yilmaz, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proc. ASRU*, 2013, pp. 416–421.

[5] Stavros Tsakalidis, Roger Hsiao, Damianos Karakos, Tim Ng, Shivesh Ranjan, Guruprasad Saikumar, Le Zhang, Long Nguyen, Richard Schwartz, and John Makhoul, "The 2013 BBN Vietnamese telephone speech keyword spotting system," in *Proc. ICASSP*, 2014, pp. 7829–7833.

[6] Nancy F Chen, Sunil Sivadas, Boon Pang Lim, Hoang Gia Ngo, Haihua Xu, Bin Ma, Haizhou Li, et al., "Strategies for Vietnamese keyword search," in *Proc. ICASSP*, 2014, pp. 4121–4125.

[7] Van Tung Pham, Haihua Xu, Xiong Xiao, Nancy F Chen, Eng Siong Chng, and Haizhou Li, "Keyword search using query expansion for graph-based rescoring of hypothesized detections," in *Proc. ICASSP*, 2016, pp. 6035–6039.

[8] Richard C Rose and Douglas B Paul, "A hidden Markov model based keyword recognition system," in *IProc. ICASSP*, 1990, pp. 129–132.

[9] Jay G Wilpon, Lawrence R Rabiner, C-H Lee, and ER Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, 1990.

[10] JG Wilpon, LG Miller, and P Modi, "Improvements and applications for keyword recognition using hidden Markov modeling techniques," in *Proc. ICASSP*, 1991, pp. 309–312.

[11] M. Weintraub, "Keyword-spotting using SRI's DECIPHER large-vocabulary speech-recognition system," in *Proc. ICASSP*, 1993, pp. 463–466.

[12] Guoguo Chen, Carolina Parada, and Georg Heigold, "Small-footprint keyword spotting using deep neural networks.," in *Proc. ICASSP*, 2014, pp. 4087–4091.

[13] Preetum Nakkiran, Raziel Alvarez, Rohit Prabhavalkar, and Carolina Parada, "Compressing deep neural networks using a rank-constrained topology," in *Proc. INTERSPEECH*, 2015, pp. 1473–1477.

[14] Tara Sainath and Carolina Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2015, pp. 1478–1482.

[15] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Gengshen Fu, and Shiv Vitaladevuni, "Model compression applied to small-footprint keyword spotting.," in *Proc. INTERSPEECH*, 2016, pp. 1878–1882.

[16] Raphael Tang and Jimmy Lin, "Deep residual learning for small-footprint keyword spotting," in *Proc. ICASSP*, 2018, pp. 5484–5488.

[17] Ming Sun, Varun Nagaraja, Björn Hoffmeister, and Shiv Vitaladevuni, "Model shrinking for embedded keyword spotting," in *Proc. ICMLA*, 2015, pp. 369–374.

[18] Minhua Wu, Sankaran Panchapagesan, Ming Sun, Jiacheng Gu, Ryan Thomas, Shiv Naga Prasad Vitaladevuni, Bjorn Hoffmeister, and Arindam Mandal, "Monophone-based background modeling for two-stage on-device wake word detection," in *Proc. ICASSP*, 2018, pp. 5494–5498.

[19] Shane Settle, Keith Levin, Herman Kamper, and Karen Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," in *Proc. INTERSPEECH*, 2017, pp. 2874–2878.

[20] Yougen Yuan, Cheung-Chi Leung, Lei Xie, Hongjie Chen, Bin Ma, and Haizhou Li, "Learning acoustic word embeddings with temporal context for query-by-example speech search," in *Proc. INTERSPEECH*, 2018, pp. 97–101.

[21] Yougen Yuan, Cheung-Chi Leung, Lei Xie, Hongjie Chen, Bin Ma, and Haizhou Li, "Query-by-example speech search using recurrent neural acoustic word embeddings with temporal context," *IEEE Access*, vol. 7, no. 1, pp. 67656–67665, 2019.

[22] Tzeviya Fuchs and Joseph Keshet, "Robust spoken term detection automatically adjusted for a given threshold," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1–8, 2017.

[23] Keith Levin, Katharine Henry, Aren Jansen, and Karen Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. ASRU*, 2013, pp. 410–415.

[24] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015, pp. 815–823.

[25] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2014.

[26] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.