F-Measure Based End-to-End Optimization of Neural Network Keyword Detectors

Tomohiro Tanaka and Takahiro Shinozaki
Tokyo Institute of Technology, Tokyo, Japan
http://www.ts.ip.titech.ac.jp/

***Abstract*—F-measure is a widely used evaluation criterion for keyword detection where measures such as accuracy and cross-entropy are not appropriate since the numbers of positive and negative samples are largely unbalanced. In this work, we propose a soft decision version of the F-measure as an objective function to train end-to-end neural network based keyword detectors. It has advantages that the training objective is consistent with the final evaluation measure. We apply it to acoustic embedding based keyword detectors using long short term memory (LSTM). Evaluation experiments using the WSJ corpus show that the proposed F-measure based training improves keyword detection performance than cross-entropy training while eliminating a meta-parameter to balance the effect of positive and negative samples in training. Additionally, we propose end-to-end continuous dynamic programming (DP) matching using a two-dimensional recurrent neural network (2D-RNN) and train it with the proposed F-measure criterion. While it has unstable behavior in the training, higher detection performance than the embedding method has achieved.**

I. Introduction

Keyword detection is used for applications such as voice command recognition. A closely related task is spoken document retrieval where utterance database is searched for a spoken query. To detect a keyword or a query in the utterances, Dynamic programming (DP) matching [1] has long been used with acoustic features such as MFCC [2] and PLP [3], posterior features by a Gaussian mixture model (GMM) and by a deep neural network (DNN) [4], [5], and bottle-neck DNN features [6], etc.

Another approach is to use the acoustic embedding. In the approach, both query and target segment are encoded in a fixed dimensional vector, and the matching is evaluated by the similarity of the two vectors. It has an advantage that the computation is fast once the embeddings are obtained. Levin et al. have proposed several linear (e.g. principal component analysis) and non-linear (e.g. Laplacian eigenmaps) embedding techniques [7], [8]. Chen et al. have proposed to use LSTM to obtain acoustic embedding from its hidden activations as bottle-neck features, where the LSTM was trained using words as the output target [9]. It was reported that their system outperformed standard phoneme posterior based DP matching. In [10], Settle et al. have shown that a neural network based embedding outperform previous embedding approaches when labeled training data is available. For the training, they used Siamese recurrent neural networks with a contrastive triplet loss [11], [12].

Recently, several neural network based word embedding methods have emerged [13], [14], [15], [16], and several training approaches have been used. Among them, Audhkhasi et al. have proposed an end-to-end optimization approach [17] for a keyword search task, where the network makes a binary decision whether a keyword is included in a given utterance or not. In their experiment, training and test sets were made and used that consisted of the same number of positive and negative samples. Ao et al. introduced an attention mechanism [18] to better handle speech database in which the utterances were not pre-segmented at word boundaries [19]. The network was trained by a cross-entropy criterion for the binary decision. While the end-to-end approach has an advantage that the system is monolithically optimized, these works are limited in that unbalance of samples is not considered.

For the task where the number of positive (or negative) samples is much smaller than the number of negative (positive) samples, evaluation metrics such as accuracy and cross-entropy are not suited. For example, a strategy that always makes negative (positive) judgment gets very good score closed to perfect. In such a situation, F-measure is useful. For this reason, Xu et al. have proposed to use expected F-measure as an objective for RNN based shift-reduce parser training, where the expectation is obtained for an N-best list of the parsing results by computing a weighted average of F-measures of the hypotheses using their normalized probabilities [20].

In this paper, we propose soft F-measure as an objective function for training neural network based keyword detectors. The soft F-measure is first proposed as an evaluation measure for soft clustering [21], and our definition is an extension of it. Their difference is that while the original definition is based on the ratio of counts in the soft clustering result, our definition is based on binary prediction probability. In addition to the advantage that it makes the training and evaluation objectives consistent, it also has an advantage that it removes a weight meta-parameter in the training that is needed for cross-entropy criterion to compensate for unbalanced samples.

We apply the proposed soft F-measure to train an embedding based end-to-end keyword detector using LSTM. The soft F-measure can also be applied to the DP matching approach by implementing it as an end-to-end system using 2D-RNN [22]. The 2D-RNN based DP matching has originally been proposed by Che et al. to match patient records [23]. We apply it to keyword detection with an extension to continuous DP

matching [24] to allow free staring and ending time points and train it with the proposed criterion.

II. End-to-end training of keyword detectors

A. Weighted cross-entropy criteria

As a baseline training objective, we use cross-entropy as it is used in [19]. In the paper, the ratio of the number of positive and negative samples in the training is not described. However, we have found that the training sometimes does not work when the ratio is largely biased. When it is largely biased toward negative samples, the output becomes mostly 0. Therefore, we introduce a meta-parameter $w\,(>0)$ to weight the log likelihood of the positive samples as shown in Equation (1).

$$WCE = -\frac{1}{T}\sum_{t=1}^{T}\left(wl_t \log y_t + (1-l_t)\log\left(1-y_t\right)\right), \quad (1)$$

where $l_t \in \{0,1\}$ is a binary label and y_t is an output of the keyword detector at the t-th time frame.

B. Proposed Soft F-Measure

The (normal) F-measure is defined by Equations (2), (3), and (4), where $Precision$ is a ratio of correctly detected samples among all the detected samples, $Recall$ is a ratio of correctly detected samples among all the samples to be detected, TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The counts of TP, FP, and FN are obtained by comparing the hypothesis with the ground truth and they are all non-negative integers. The range of F-measure is $[0,1]$, and larger value means better.

$$F = 2\frac{Recall \cdot Precision}{Recall + Precision}, \quad (2)$$

$$Recall = \frac{TP}{TP+FN}, \quad (3)$$

$$Precision = \frac{TP}{TP+FP}. \quad (4)$$

The soft F-measure shares the mostly the same definitions as shown in Equations (5), (6), and (7). The difference is that the integer counts are replaced with soft counts based on the system output representing detection probability as defined in Equations (8), (9), and (10), where y_t is the system output for the t-th sample, and l_t is the corresponding ground truth.

$$F_{soft} = 2\frac{softRecall \cdot softPrecision}{softRecall + softPrecision}, \quad (5)$$

$$softRecall = \frac{softTP}{softTP+softFN}, \quad (6)$$

$$softPrecision = \frac{softTP}{softTP+softFP}. \quad (7)$$

$$softTP = \sum_t y_t l_t, \quad (8)$$

$$softFP = \sum_t y_t\left(1-l_t\right), \quad (9)$$

$$softFN = \sum_t \left(1-y_t\right) l_t. \quad (10)$$

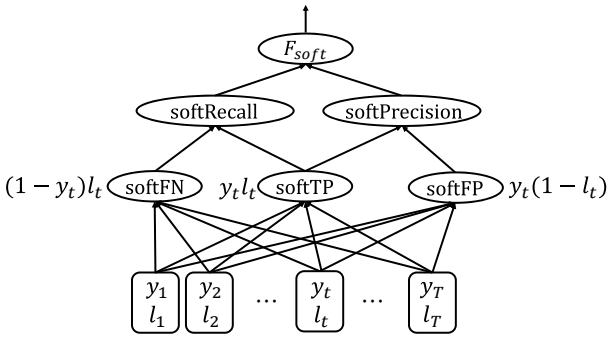

Fig. 1. Neural network implementation of the soft F-measure evaluation.

In this work, we use F-measures based on the acoustic feature frame unit. In the training, the soft F-measure is evaluated at each mini-batch that contains a set of speech segments. The gradients of the neural network parameters $\boldsymbol{\theta}$ is obtained by Equation (11) using the chain rule.

$$\frac{\partial F_{soft}}{\partial \boldsymbol{\theta}} = \sum_t \frac{\partial F_{soft}}{\partial y_t}\frac{\partial y_t}{\partial \boldsymbol{\theta}}, \quad (11)$$

where $\frac{\partial y_t}{\partial \boldsymbol{\theta}}$ is the partial derivative of the neural network output with respect to the network parameters, and $\frac{\partial F_{soft}}{\partial y_t}$ is the partial derivative of the soft F-measure with respect to the neural network output, which is given by Equation (12).

$$\frac{\partial F_{soft}}{\partial y_{t'}} = 2\frac{l_{t'}\sum_t y_t + l_{t'}\sum_t l_t - \sum_t y_t l_t}{\left(\sum_t y_t + \sum_t l_t\right)^2}. \quad (12)$$

As shown in Figure 1, the soft F-measure estimation from the reference labels and the neural network outputs can be implemented as an additional layer on top of the network. Therefore, the evaluation of the gradient (12) can simply be performed as a part of back-propagation for the extended network.

III. End-to-end continuous DP Matching

Continuous DP matching is a variant of DP matching that is applied when the starting and ending time frames of the keyword matching is unknown. Let $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_M\}$ be a frame sequence of a keyword and $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_T\}$ be a frame sequence of a speech segment subject to the search. Assuming $dist\left(\mathbf{a}, \mathbf{b}\right)$ is a distance between two vectors $\mathbf{a}$ and $\mathbf{b}$, a matching cost between the keyword and a subsegment of the speech segment whose length is n and ending time is t is defined by Equation 13 given an alignment, which is a sequence of pair of indexes of the keyword and the subsegment frames $\{(\alpha_1,\beta_1),(\alpha_2,\beta_2),\cdots,(\alpha_i,\beta_i),\cdots,(\alpha_L,\beta_L)\}$, where $\alpha_1 = 1$, $\beta_1 = t-n+1$, $\alpha_L = M$, and $\beta_L = t$, and L is the length of the alignment.

$$\sum_{i=1}^{L} dist\left(\mathbf{q}_{\alpha_i}, \mathbf{s}_{\beta_i}\right). \quad (13)$$

At each time frame t, continuous DP matching returns a minimum matching cost between the keyword and a subsegment of the target sequence ending at t with unknown starting

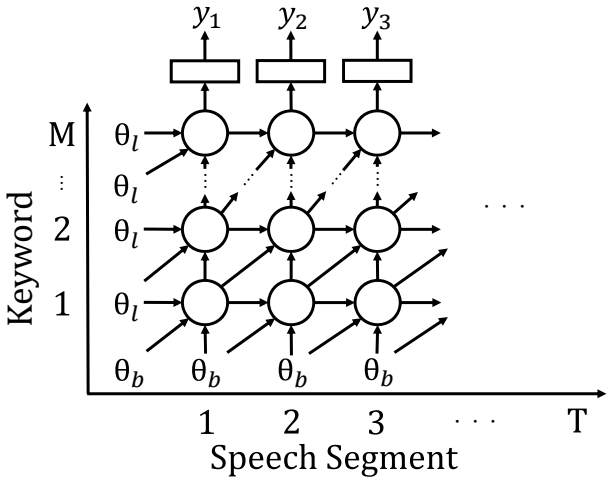

Fig. 2. Search process of continuous DP matching. The process can be interpreted as or simulated by 2D-RNN. The vertical axis corresponds to the frames of the keyword, and the horizontal axis is the time frames of a speech segment subject to the search.

frame based on an optimal alignment. By using dynamic programming [25], the minimum cost is efficiently obtained by searching all possible starting frames and alignments.

Figure 2 visualizes the search process by the dynamic programming. At each lattice node $\langle t, m\rangle$, $h_{t,m}$ in Equation (14) is computed. It receives accumulated messages from three directions and adds a distance of q_m and s_t to their minimum.

$$h_{t,m} = \min\left(h_{t-1,m}, h_{t-1,m-1}, h_{t,m-1}\right) + dist\left(q_m, s_t\right). \tag{14}$$

The order of the computation at each time frame t is from the bottom to the top, and it proceeds synchronizing to the time frames of the speech segment.

The boundary conditions $h_{0,m}$ and $h_{t,0}$ are given as $\forall_{m>0} h_{0,m} = \theta_l$ and $\forall_{t\geq 0} h_{t,0} = \theta_b$, where $\theta_l = \inf$ and $\theta_b = 0$. The condition $\theta_l = \inf$ guarantees that all the keyword frames are used for the matching, while the condition $\theta_b = 0$ allows free starting point in the speech segment. The minimum matching cost where the ending point is t is obtained from the top node at the t-th frame. By applying a threshold to the matching cost, a binary decision of detection is obtained.

The process of the continuous DP matching is simulated by a 2D-RNN by interpreting the nodes in Figure 2 as a neuron group. The parameters of the neuron group are shared by all the nodes in the lattice. As the 2D-RNN, we use 2D-GRU that has 2 units per a node. Figure 3 shows the unit, which is defined by Equations (15) to (20) where σ and ψ are sigmoid and tanh functions respectively. Functions f and g are defined by equations (21) and (22), where $\boldsymbol{w}_\gamma$, $\boldsymbol{b}_\gamma$, $\boldsymbol{W}_{\gamma'}$, and $\boldsymbol{b}'_{\gamma'}$ are their parameters. The distance $dist\left(q_m, s_t\right)$ is computed by a neural network shown in Figure 4. A sigmoid unit is used as the output layer at each time frame. Therefore, the output is normalized and represents a probability of detection while it is a distance and not normalized in the original DP matching. This makes the choice of the decision threshold easier.

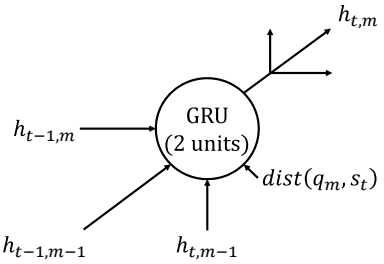

Fig. 3. A GRU node to implement a DP matching node.

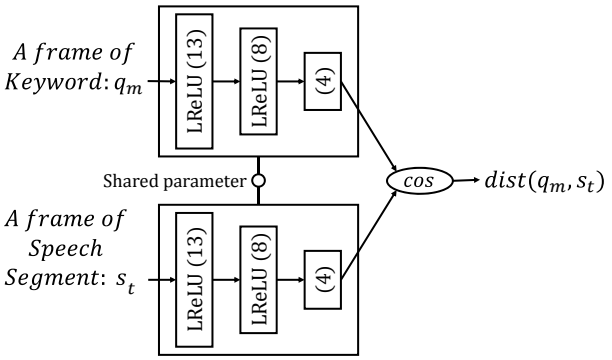

Fig. 4. A feed-forward network used to compute a distance of two input vectors as the front-end for the 2D-RNN based continuous DP matching. The numbers in the parentheses indicate the number of neuron units used in our experiments.

$$x_{t,m} = dist\left(q_m, s_t\right), \tag{15}$$
$$\boldsymbol{c}_{t,m} = \left[\boldsymbol{h}^T_{t-1,m}, \boldsymbol{h}^T_{t-1,m-1}, \boldsymbol{h}^T_{t,m-1}\right]^T, \tag{16}$$
$$\boldsymbol{r}_{t,m} = \sigma\left(f_r\left(x_{t,m}\right) + g_r\left(\boldsymbol{c}_{t,m}\right)\right), \tag{17}$$
$$\boldsymbol{z}_{t,m} = \sigma\left(f_z\left(x_{t,m}\right) + g_z\left(\boldsymbol{c}_{t,m}\right)\right), \tag{18}$$
$$\bar{\boldsymbol{h}}_{t,m} = \psi\left(f_v\left(x_{t,m}\right) + g_v\left(\boldsymbol{r}_{t,m} \odot \boldsymbol{c}_{t,m}\right)\right), \tag{19}$$
$$\boldsymbol{h}_{t,m} = g_u\left(\boldsymbol{z}_{t,m} \odot \bar{\boldsymbol{h}}_{t,m}\right) + g_o\left((\boldsymbol{1} - \boldsymbol{z}_{t,m}) \odot \boldsymbol{c}_{t,m}\right). \tag{20}$$

$$f_\gamma\left(x\right) = \boldsymbol{w}_\gamma x + \boldsymbol{b}_\gamma \qquad (\gamma = r, z, v), \tag{21}$$
$$g_{\gamma'}\left(\boldsymbol{c}\right) = \boldsymbol{W}_{\gamma'}\boldsymbol{c} + \boldsymbol{b}'_{\gamma'} \qquad (\gamma' = r, z, v, u, o). \tag{22}$$

IV. Experimental Setup

Experiments were performed using the Wall Street Journal corpus [26]. Kaldi toolkit [27] was used to extract 13-dimensional MFCC features. For the feature analysis, the window width was 25 ms and the shift was 10 ms. We pooled train_si286, dev93, eval92 and eval93 sets and then divided it into two exclusive sets A and B at the recording session level. The ratio of set A and set B was 5:1, and there was no speaker overlap between them. As the keywords, 540 words were selected from set A that appeared more than 70 and less than 400 times in it. Similarly, 200 words were selected from set B that appeared more than 20 and less than 50 times. The selection was random with a constraint that a word is

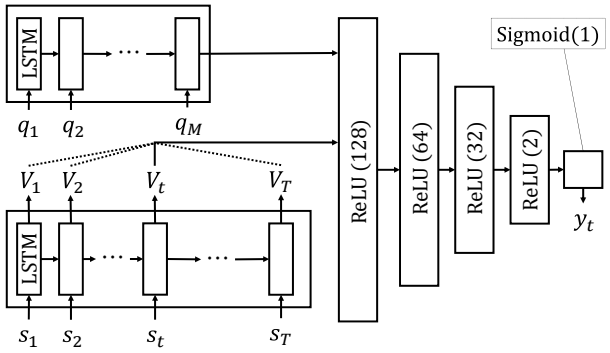

Fig. 5. Acoustic embedding based keyword detector using LSTM. An output is obtained at each time as a binary probability at each time frame indicating whether the frame corresponds to an endpoint of the keyword or not.

selected at most once through sets A and B. Therefore, there was no overlap in keywords between the two sets. To extract keyword templates, one utterance per a keyword was used. The utterance that provided the template was excluded from the segments used as the subject of the search of that keyword. The remaining utterances in sets A and B were concatenated for each original recording session, respectively, and then equally cut to make 5-second length segments for parallel processing. As a training-development set, 50 segments per a keyword from set A were used, where each segment contained one or more keywords. Since we had 540 keywords, 27000 segments were used in total. Among them, 24000 segments were used for the network training and 3000 segments were used as the development set. Similarly, 15 segments per a keyword from set B were used for the evaluation, which amounted to 3000 segments in total. To make a frame level binary reference label, forced alignment was performed using a GMM-HMM triphone model trained by the WSJ Kaldi recipe. The label takes 1 if the keyword being considered is completely included in the segment, and the frame is the end point of that keyword. To allow small errors in the timing for keyword detection, 9 preceding and 10 succeeding frames were also labeled as 1. Other frames were labeled as 0. The ratio of the labels 1 and 0 was around 1:23.6.

As a baseline, we trained an acoustic embedding based keyword detector using LSTM with the weighted cross-entropy criterion. The network structure is shown in Figure 5. A keyword is input to an LSTM consisting of two layers having 128 units and its embedded vector is obtained as a hidden activation of the final time frame. Frames of a speech segment are input to another LSTM that shares the same parameters with the keyword encoder. The LSTM outputs an embedded vector at each time frame. At each time frame, the embedded vectors of the keyword and the partial speech segment are concatenated to form a 256-dimensional vector and it is input to a feed-forward network. The final output is obtained as a probability of a binary variable indicating whether the time frame corresponds to an endpoint of the keyword or not. The backpropagation was performed using ADAM [28]. The number of training epochs was decided by monitoring the objective score on the development set. To evaluate the proposed soft F-measure based training, it was applied to the same network as the baseline. Additionally, it was also applied to train the continuous DP matching based keyword detector using 2D-RNN.

TABLE I
F-MEASURES ON THE DEVELOPMENT AND THE TEST SETS. THE CROSS-ENTROPY WEIGHT AND THE DECISION THRESHOLD ARE TUNED USING THE DEVELOPMENT SET.

Detector	Dev	Test
Continuous DP	19.3	19.7
LSTM (WCE)	52.6	46.7
LSTM (Soft F)	57.3	50.6
2D-RNN (Soft F)	60.8	62.1

V. RESULTS

Table I shows keyword detection performance on the development and the test set evaluated by F-measure. In the table, "LSTM" means the acoustic embedding based keyword detector using LSTM. "WCE" means baseline weighted cross-entropy based training, and "Soft F" means the proposed soft F-measure based training. "2D-RNN (Soft F)" means 2D-RNN based continuous DP matching using the soft F-measure training criterion. As we have observed instability in the training of the 2D-RNN, we repeated the training 6 times and selected the best model based on the development set. Results by conventional continuous DP matching is also shown as "Continuous DP" for comparison purpose. For these results, the cross-entropy weight and the decision threshold were optimized using the development set. As can be seen, the LSTM based detectors give much better results than the conventional continuous DP matching. The soft F-measure training gave 3.9% absolute improvement from the weighted cross-entropy training for the LSTM based detectors. While the 2D-RNN needs computational cost linear to the length of the keyword and it has instability in the training, it gave the highest performance. Another observation is that differences in the performance between the development and the test sets are relatively large for the embedding methods. This is probably because the learned embedding is vulnerable to the difference of the speakers. On the other hand, the differences are small when the conventional and 2D-RNN based continuous DP matchings are used.

Figure 6 analyzes the relationship between the cross-entropy weight during the training of the LSTM based detector and the keyword detection performance on the test set, where the detection performance was evaluated by F-measure. The decision thresholds were optimized using the development set for each weight condition. For the weighted cross-entropy based training, the best result of 46.7% was obtained when the weight was 1.0. Soft F-measure results are also shown in the figure for the comparison purpose. The proposed soft F-measure based training gave higher performance both when a fixed threshold 0.5 was used (48.8%) and when the threshold

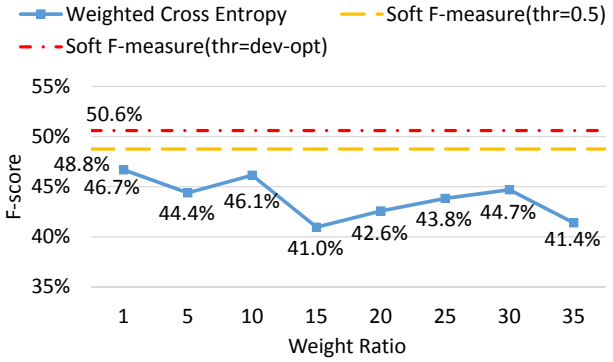

Fig. 6. A relationship between the weight for the cross-entropy training and the keyword detection performance on the test set. The decision thresholds were tuned using the development set for each weight. The results of soft F-measure training are also shown for comparison purpose where a predefined threshold of 0.5 and an optimized threshold based on the development set are used.

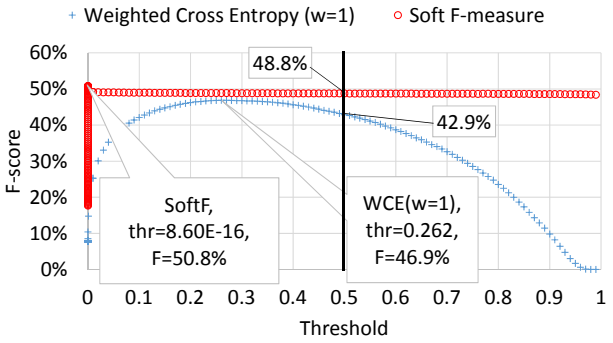

Fig. 7. A relationship between the decision threshold and the keyword detection performance on the test set. The highest F-measure values and the ones when the threshold is 0.5 are annotated.

was optimized using the development set (50.6%), while removing the training weight meta-parameter.

Figure 7 plots the relationship between the decision threshold and the keyword detection performance. It is seen that while the performance of the weighted cross-entropy trained LSTM keyword detector makes a bell-like curve, the soft F-measure trained LSTM detector makes a mostly flat curve. The highest F-measure values were mostly the same as those ones obtained by using thresholds decided by the development set. To further analyze, a histogram of the output detection probability of the test set is shown in Figure 8. Note that the vertical axis is in log scale. It is found that the soft F-measure trained detector tends to give an output score that is close to either 0.0 or 1.0.

VI. Discussion

While the 2D-RNN gave the highest F-measure on the test set, we have observed significant instability in its training. Figure 9 shows an example of the changes in the soft F-measure objective function in the training where the training failed. The instability was probably because of an interaction of the soft F-measure evaluation and the DP matching lattice structure that

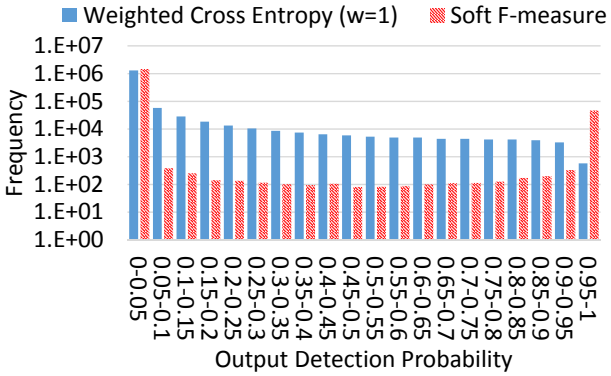

Fig. 8. Histogram of the output detection probability by the keyword detectors.

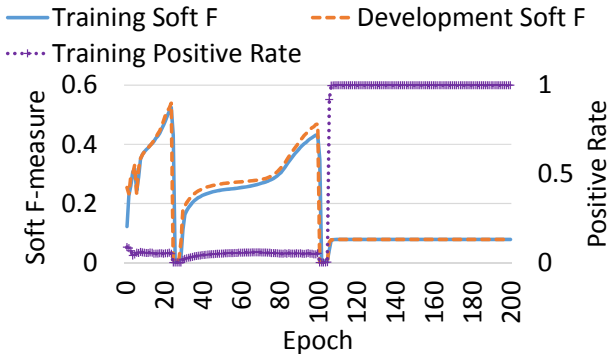

Fig. 9. Example of failed 2D-RNN training. The graph plots the number of training epochs vs. the soft F-measure score of the 2D-RNN (left axis) and rate of positive decisions of the detector for the training set samples (right axis). The decision threshold was set to 0.5.

makes very deep neural network layers. Although we could have chosen the best model in the evaluation in this case since the correlation between the development set score and the test set performance was good, there may be a room for further improvement if we could address this instability.

Two things that we can read from the figure are that 1) there are some timings that the soft F-measure objective score suddenly and drastically falls, and 2) there is a case where the dropped score does not recover. We have found that the large drop of the score is due to sudden large change of the model parameters. We conjecture that the persistent low score is partly due to the property of the gradient of the soft F-measure. As shown in Equation (23), there is an upper bound that the gradient of the soft F-measure can take.

$$\begin{aligned}\left|\frac{\partial F_{soft}}{\partial y_{t'}}\right| &= 2\left|\frac{l_{t'}}{\sum_t y_t + \sum_t l_t} - \frac{\sum_t y_t l_t}{\left(\sum_t y_t + \sum_t l_t\right)^2}\right| \\ &\leq 2\left(\left|\frac{\sum_t y_t + \sum_t l_t}{\sum_t y_t + \sum_t l_t}\right| + \left|\frac{\left(\sum_t y_t + \sum_t l_t\right)^2}{\left(\sum_t y_t + \sum_t l_t\right)^2}\right|\right) \\ &= 4. \end{aligned} \quad (23)$$

Since the gradient of sigmoid approaches 0 when the output approaches 1.0 or 0.0, their product becomes close to zero as

shown in Equation (24).

$$\begin{aligned} \left|\frac{\partial F_{soft}}{\partial x_{t'}}\right| &= \left|\frac{\partial F_{soft}}{\partial y_{t'}}\right| \left|y_{i'}\left(1-y_{i'}\right)\right| \\ &\leq 4\exp\left(\left|x_{t'}\right|\right), \end{aligned} \tag{24}$$

where $y_{t'} = sigmoid\left(x_{t'}\right)$. It is confirmed this is actually happening in the training as we can see in the figure that there is a strong correlation between the ratio of positive decisions made by the detector and the low soft F-measure score.

VII. Conclusion

We have proposed soft F-measure as an objective function to train end-to-end neural network keyword detectors. We have applied it to acoustic embedding based keyword detector using LSTM and showed that it improves the keyword detection performance compared to the existing cross-entropy based training. We have also proposed 2D-RNN based end-to-end continuous DP matching and trained it with the proposed soft F-measure criterion. It has been shown that the 2D-RNN keyword detector gives better performance than the LSTM detector. However, we also have observed significant instability with its training. Future work includes addressing the instability problem of the 2D-RNN based keyword detector, and utilizing word based soft F-measure instead of the frame based soft F-measure to make the training and the evaluation criteria more consistent to further improve the performance.

VIII. Acknowledgements

Part of this work was supported by JSPS KAKENHI Grant Number 17K20001. Part of this work was also supported by Microsoft Research Core 12 Program.

References

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

[2] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transaction on Acoustic Speech and Singal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[3] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am*, vol. 87, no. 4, pp. 1738–1752, 1990.

[4] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams," in *Proc. ASRU*, 2009, pp. 398–403.

[5] T. Shinozaki and S. Watanabe, "Structure discovery of deep neural network based on evolutionary algorithms," in *Proc. ICASSP*, 2015, pp. 4979–4983.

[6] Y. Bao, H. Jiang, L. Dai, and C. Liu, "Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition," in *Proc. ICASSP*, 2013, pp. 6980–6984.

[7] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proc. ASRU*, 2013, pp. 410–415.

[8] K. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *Proc. ICASSP*, 2015, pp. 5828–5832.

[9] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *Proc. ICASSP*, 2015, pp. 5236–5240.

[10] 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.

[11] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. CVPR*, 2005, vol. 1, pp. 539–546.

[12] Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *TACL*, vol. 2, pp. 207–218, 2014.

[13] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *Proc. ICASSP*, 2016, pp. 4950–4954.

[14] Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-yi Lee, and Lin-Shan Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," in *Proc. Interspeech*, 2016, pp. 765–769.

[15] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," in *Proc. SLT*, 2016, pp. 503–510.

[16] Wanjia He, Weiran Wang, and Karen Livescu, "Multi-view recurrent neural acoustic word embeddings," in *Proc. ICLR*, 2017.

[17] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, Dec 2017.

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[19] Chia-Wei Ao and Hung-yi Lee, "Query-by-example spoken term detection using attention-based multi-hop networks," in *Proc. ICASSP*, 2018, pp. 6264–6268.

[20] Wenduan Xu, Michael Auli, and Stephen Clark, "Expected f-measure training for shift-reduce parsing with recurrent neural networks," in *HLT-NAACL*, 2016, pp. 210–220.

[21] Tim Gollub, Matthias Busse, Benno Stein, and Matthias Hagen, "Key-queries for clustering and labeling," in *Information Retrieval Technology*, Cham, 2016, pp. 42–55, Springer International Publishing.

[22] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber, "Multi-dimensional recurrent neural networks," in *Artificial Neural Networks – ICANN 2007*, Berlin, Heidelberg, 2007, pp. 549–558, Springer Berlin Heidelberg.

[23] Chao Che, Cao Xiao, Jian Liang, Bo Jin, Jiayu Zho, and Fei Wang, "An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 198–206.

[24] Ryuichi Oka, "Spotting method for classification of real world data.," *Comput. J.*, vol. 41, no. 8, pp. 559–565, 1998.

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

[26] Douglas B. Paul and Janet M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the Workshop on Speech and Natural Language*, Stroudsburg, PA, USA, 1992, HLT '91, pp. 357–362, Association for Computational Linguistics.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovskı, G. Stemmer, and K. Veselı, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[28] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.