Neural Networks 139 (2021) 326–334

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2021 Special Issue

End-to-end keyword search system based on attention mechanism and energy scorer for low resource languages

Zeyu Zhao, Wei-Qiang Zhang *

Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Available online 10 April 2021

Keywords:
Keyword search
End-to-end
Low resource language
Deep neural network

ABSTRACT

Keyword search (KWS) means searching for keywords given by the user from continuous speech. Conventional KWS systems are based on Automatic Speech Recognition (ASR), where the input speech has to be first processed by the ASR system before keyword searching. In the recent decade, as deep learning and deep neural networks (DNN) become increasingly popular, KWS systems can also be trained in an end-to-end (E2E) manner. The main advantage of E2E KWS is that there is no need for speech recognition, which makes the training and searching procedure much more straightforward than the traditional ones. This article proposes an E2E KWS model, which consists of four parts: speech encoder–decoder, query encoder–decoder, attention mechanism, and energy scorer. Firstly, the proposed model outperforms the baseline model. Secondly, we find that under various supervision, character or phoneme sequences, speech or query encoders can extract the corresponding information, resulting in different performances. Moreover, we introduce an attention mechanism and invent a novel energy scorer, where the former can help locate keywords. The latter can make final decisions by considering speech embeddings, query embeddings, and attention weights in parallel. We evaluate our model on low resource conditions with about 10-hour training data for four different languages. The experiment results prove that the proposed model can work well on low resource conditions.

1. Introduction

Keyword search (KWS) or keyword spotting is to detect and locate keywords entered by the user in continuous speech and output confidence scores to indicate the probabilities that the keywords occur in input speech. The entered keywords can be in text form or spoken form. In this article, we only discuss keywords in the text form. Any character sequence in the target language can be input as a query. Conventional ASR-based KWS systems (Trmal et al., 2017) usually decode speech into word lattices or other forms of decoding results by ASR systems. After that, a backend KWS system (Hartmann, Le, Messaoudi, Lamel, & Gauvain, 2014) is applied to find the keywords of interest. However, as DNNs become increasingly popular, KWS systems can also be developed in an end-to-end fashion (Audhkhasi, Rosenberg, Sethy, Ramabhadran, & Kingsbury, 2017a, 2017b; Kamper, Anastassiou, & Livescu, 2019; Kim, Lee, Lee, Kim, & Hwang, 2019; Lengerich & Hannun, 2016; Sacchi, Nanchen, Jaggi, & Cernak, 2019; Settle, Levin, Kamper, & Livescu, 2017; Sharma et al., 2020; Tanaka & Shinozaki, 2018, 2019). The advantage of the E2E KWS system is that we need neither the ASR system nor the corresponding decoding steps anymore (Audhkhasi et al., 2017a, 2017b). Besides, to process out-of-vocabulary (OOV) queries, traditional ASR-based KWS systems usually require large sub-word lattices (Mamou, Ramabhadran, & Siohan, 2007), confusion networks (Mangu, Kingsbury, Soltau, Kuo, & Picheny, 2014) or other technologies such as proxy words (Chen, Yilmaz, Trmal, Povey, & Khudanpur, 2013). However, the E2E system of a full neural network can avoid the above situation. Finally, it is sometimes difficult to train an ASR system when resources are insufficient. Still, intuitively, it is much easier to train E2E KWS systems with the same amount of data as KWS is a low-level task compared with ASR somehow.

A primary method of E2E ASR-free KWS is using encoders to extract embeddings or features from input speech and queries and perform KWS with them (Audhkhasi et al., 2017a, 2017b; Sacchi et al., 2019), which is also the baseline model in this article and will be briefly introduced in Section 2. In Audhkhasi et al. (2017a, 2017b), the authors used an auto-encoder to restore the original input speech features from the speech embeddings extracted by the encoder part. If the restoring effect is perfect, almost no information is lost during the input speech features' compression to the speech embeddings. However, we believe

* Corresponding author.
E-mail addresses: zzy17@mails.tsinghua.edu.cn (Z. Zhao), wqzhang@tsinghua.edu.cn (W.-Q. Zhang).

https://doi.org/10.1016/j.neunet.2021.04.002
0893-6080/

that the most crucial and essential information that speech embeddings should contain is the character or phoneme sequence corresponding to input speech, which is what subsequent KWS operations only need, but not the whole input speech features. To this end, we attempt to convert the speech embeddings into the corresponding character or phoneme sequence to lead the speech encoder to extract the necessary information into the speech embeddings. The authors of Audhkhasi et al. (2017a, 2017b) and Sacchi et al. (2019) converted the input speech into a fixed-length vector, where the time dimension was lost, but we believe the time domain is vital for the speech signal. From the experiment section of this article, we find that converting the input speech into a variable-length matrix is better, maintaining the time dimension. As for the processing of input queries, we find that the sequence-to-sequence (Seq2Seq) (Sutskever, Vinyals, & Le, 2014) model is better than the CNN-RNN character LM (language model) in Audhkhasi et al. (2017a, 2017b). Besides, by using different kinds of labels, we may obtain query embeddings containing various types of information, i.e., spelling or pronunciation. Fortunately, there has been some method readily available for us to apply to our practices. Connectionist temporal classification (CTC) (Graves, Fernández, Gomez, & Schmidhuber, 2006) makes it possible to train an ASR system in an E2E fashion without alignment (Graves, Mohamed, & Hinton, 2013). Thus, we apply CTC as one of the speech decoders to predict character or phoneme sequences. Seq2Seq (Sutskever et al., 2014) was initially proposed in the area of neural machine translation. According to our experiments, we find that it is much more efficient than CNN-RNN character LM in the baseline system (Audhkhasi et al., 2017a, 2017b). To deal with the difficulty when processing long sequences, some researchers proposed attention mechanisms (Chorowski, Bahdanau, Serdyuk, Cho, & Bengio, 2015; Luong, Pham, & Manning, 2015) as a supplement. In this article, we also use the attention-based Seq2Seq model as another kind of speech decoder. Besides, inspired by the attention mechanism (Chorowski et al., 2015; Luong et al., 2015) in sequence modeling, we introduce an attention mechanism for E2E KWS.

The final phase of Audhkhasi et al. (2017a, 2017b) is called the KWS system, a multilayer perceptron (MLP) that takes speech and query embeddings as input and outputs probabilities indicating how likely the keywords occur in the input speech. We find that a more sophisticated structure can replace it to obtain better performance. Specifically, we introduce an attention mechanism and a novel component called the *energy scorer* to accomplish this goal.

The main contributions of our work are that

1. We improve recovery accuracy by applying a more suitable network structure for query encoder–decoder;
2. By transforming speech and query embeddings into character or phoneme sequences, the encoders may extract spelling or pronunciation information from input speech and query respectively;
3. We apply two kinds of speech encode-decoder and compare the performances, and
4. We propose an attention mechanism and a novel energy scorer, which are much more sophisticated components than MLP.

In the following, in Section 2 we will first briefly introduce the baseline model and analyze its shortage, from which we propose our improved methods. After that, the proposed method will be described in depth in Section 3. Next, the experiment results and analysis will be given in Section 4. Finally, we will conclude this article in Section 5.

2. Baseline model

In this section, we will briefly introduce the baseline system (Audhkhasi et al., 2017a, 2017b). It consists of three components, including RNN acoustic auto-encoder, CNN-RNN character LM and KWS system.

To deal with input speech, they applied the encoder part of the RNN acoustic auto-encoder that takes speech features as input and compresses them into a fix-length vector, called speech embeddings. After that, the speech embeddings will be passed through the decoder part to obtain a prediction. They intended to make the prediction close to the original speech features as much as possible to maximize the information compressed in the speech embeddings. Like speech processing, the input query will also be compressed into a fix-length vector called query embeddings, extracted by the encoder part of the CNN-RNN character LM. They then attempted to recover the original input query from the query embeddings by the decoder part. If this works well, one can say that there is little information lost during compression. Finally, given speech and query embeddings, the KWS system, an MLP, takes the concatenation of them as input and gives out the final confidence score between zero and one, indicating the probability that the keyword occurs in the input speech.

We make some improvements in this article. First, we believe that the time domain is crucial for speech signal, so we try to keep it in the speech embeddings, resulting in speech embeddings of matrices. Besides, trying to recover the original input speech features may lead the encoder part to extract embeddings without much information loss. However, we argue that some information is redundant. What is only crucial is the character or phoneme sequences corresponding to the input speech, considering the downstream KWS. Thus, we try to transform the speech embeddings into the character or phoneme sequences by the speech decoder to lead the encoder to extract spelling or pronunciation information from input speech. As for the query encoder–decoder, we find that Seq2Seq model (Sutskever et al., 2014) is much more efficient CNN-RNN Character LM in the baseline system (Audhkhasi et al., 2017a, 2017b).

3. Method

In this section, we will first introduce the overall architecture of our model. We will then discuss each component, including speech encoder–decoder, query encoder–decoder, attention mechanism, and energy scorer, further in detail.

3.1. Overall structure

The overall structure of our proposed model is shown in Fig. 1. First, we calculate speech and query embeddings by speech and query encoders, respectively. Then, we pass them through the attention mechanism to obtain a set of attention weights, which will be used by the last component, the energy scorer. Finally, the energy scorer outputs a confidence scorer given speech embeddings, query embeddings, and attention weights.

3.2. Speech encoder–decoder

In the baseline system (Audhkhasi et al., 2017a, 2017b), the authors transformed input speech into a fixed-length vector, where the temporal dimension was eliminated. On the contrary, we argue that temporal information is essential for speech signals, which is inspired by the development of neural machine translation (NMT). Initially, a successful NMT model, Seq2Seq model (Neubig, 2017; Sutskever et al., 2014), was proposed,

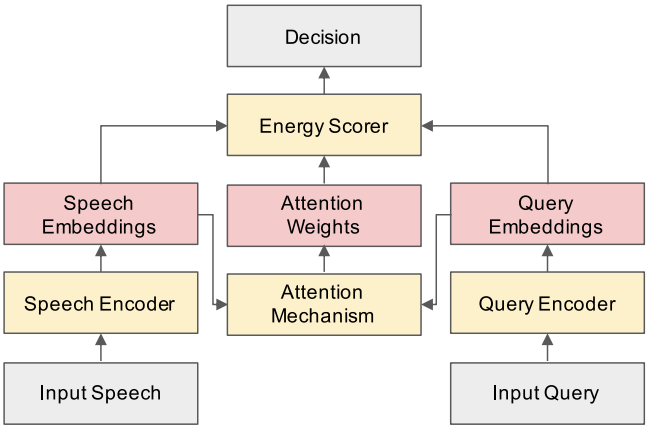

Fig. 1. The overall structure of our E2E KWS system.

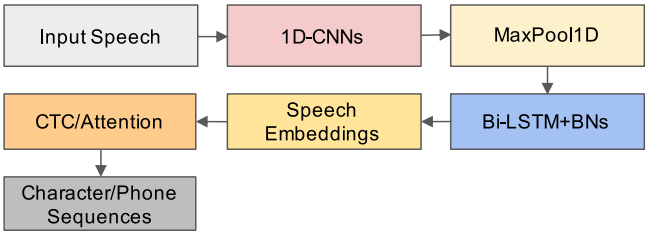

Fig. 2. The process of speech embeddings extraction and using CTC technique or attention-based decoder (CTC/Attention) to predict the character or phoneme sequences from the speech embeddings.

where the input sequence was firstly compressed into a fixed-length vector, from which the output sequence was generated. However, this method may not work well especially dealing with long input sequences (Neubig, 2017) and that is why some researchers proposed the attention-based Seq2Seq method (Bahar, Brix, & Ney, 2018; Libovický & Helcl, 2017; Neubig, 2017). In the above attention-based method, the input speech is usually transformed into high-level representations. We borrow that idea in our work and call that high-level representations speech embeddings instead.

The process of our speech encoder–decoder can be illustrated as Fig. 2. We first pass input speech features through 1-D CNNs followed by a 1-D max pooling with a stride of 2, resulting in half time-steps compared with the origin, because we want to make it quicker to train a recurrent neural network (RNN) later in the downstream. Then, we use gated recurrent units (GRU) (Chorowski et al., 2015) to extract the speech embeddings, which is a variable-length matrix but not a fixed-length vector. We have obtained speech embeddings, and they will be used in the following KWS process. However, to make these embeddings more explainable, we apply a CTC technique (Graves et al., 2006), or an Attention-based Seq2Seq (Luong et al., 2015) method to lead the speech encoder to extract the pertinent information from input speech. Note that when doing KWS, we only need to compute the speech embeddings given the input speech features but completely ignore CTC/Attention-GRU-Decoder in Fig. 2.

To describe the above process, denote input speech features as

$$X^* = \{x_1^*, x_2^*, \ldots, x_T^*\} \tag{1}$$

and after the process of 1-D CNNs and 1-D max pooling, we obtain $X = \{x_1, x_2, \ldots, x_{T/2}\}$. The speech embeddings $E_s = \{e_1, e_2, \ldots, e_{T/2}\}$ are computed by GRU-Encoder, which is a multilayer uni-directional GRU-RNN followed by a fully connected layer with ReLU activation function, as

$$E_s = \text{Encoder}(X). \tag{2}$$

Note that after ReLU activation function, $e_t > 0, 1 \le t \le T/2$.

After obtaining E_s, Attention-GRU-Decoder with Global Attention (Luong et al., 2015), at each decoding time step t, calculates the attention weight for each e_s in E_s, according to

$$\begin{aligned}\alpha_t(s) &= \text{align}(h_t, e_s) \\ &= \frac{\exp(\text{score}(h_t, e_s))}{\sum_{s'} \exp(\text{score}(h_t, e_{s'}))},\end{aligned} \tag{3}$$

where $\alpha_t(s)$ denotes the attention weight for the sth speech embedding at the tth decoding time step, h_t is the hidden state of the Attention-GRU-Decoder at time t and score(a, b) is the dot product of vector a and b.

Then, at each decoding time step t, a context vector c_t, which is the weighted sum of the speech embeddings, can be computed by

$$c_t = \sum_s \alpha_t(s) e_s. \tag{4}$$

Finally, the prediction for the tth time step o_t is obtained by concatenating the context vector c_t and the hidden state h_t and passing them through a fully connected layer (FCL) with softmax activation over all possible characters or phonemes in the target language. This can be expressed by

$$o_t = \text{FCL}([h_t^{\mathrm{T}}, c_t^{\mathrm{T}}]^{\mathrm{T}}). \tag{5}$$

Besides, given the speech embeddings, we may also apply the CTC technique to output the corresponding character or phoneme sequence. We only need to add an FCL with softmax activation and calculate the CTC loss (Graves et al., 2006) from it.

Note that the processing of extracting speech embeddings is constant, even though we use two different types of speech decoders, i.e., CTC or Attention-based Seq2Seq. Thus, we will compare and analyze them in the following experiment section.

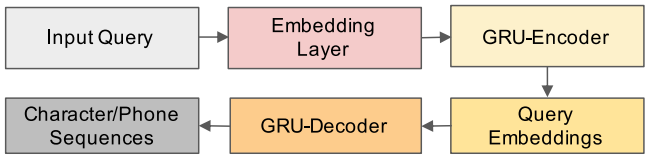

Fig. 3. The process of query embeddings extraction.

3.3. Query encoder–decoder

Like the speech encoder–decoder, to extract query embeddings from input query, a sequence of characters, and make the embeddings much more explainable, query encoder–decoder is applied in our system. The structure of query encoder–decoder is shown in Fig. 3, which is a Seq2Seq method (Sutskever et al., 2014).

We first transform the input query into query embeddings by query encoder and attempt to predict the original input character sequence or the corresponding phoneme sequence by query decoder. The query embeddings contain spelling or pronunciation information if the decoder can predict the character or phoneme sequence well from it. Specifically, the encoder and the decoder are both multilayer GRU-RNN with the same hyper-parameters to take the hidden state of the last time step in the encoder as query embeddings, which can be used as initial states by the decoder. Same with speech decoder, query decoder will not be taken into account when performing KWS.

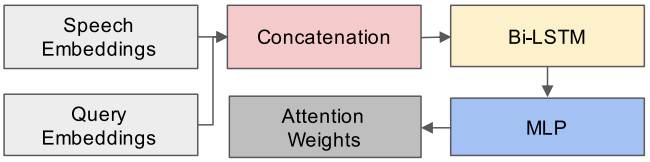

Fig. 4. The structure of the attention mechanism. The speech and query embeddings are first concatenated together (we should repeat the query embeddings first before concatenation, which is not shown in this figure for simplicity reason). Bi-LSTM and MLP denote bi-directional LSTM RNN and Multilayer Perceptron, respectively.

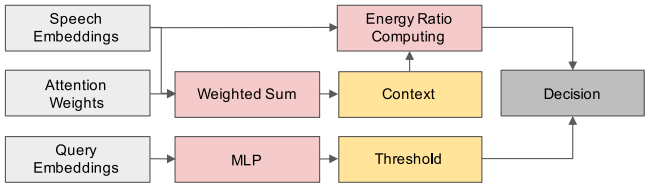

Fig. 5. The structure of the energy scorer. Firstly, we calculate the weighted sum of speech embeddings concerning the attention weights to obtain Context. Then we compute the energy ratio of the Context and the speech embeddings. Finally, we make the final decision by comparing the energy ratio and the threshold computed by passing the query embeddings through an MLP.

The overall process of the above can be denoted by

$$e_q = \text{Encoder}(\text{Embedding}(x_q)) \tag{6}$$

and

$$y_q = \text{Decoder}(e_q), \tag{7}$$

where e_q is the query embeddings (or query feature vector) extracted by query encoder, x_q is the input query (a sequence of characters) and y_q is the predicted sequence of characters. We can then minimize the cross-entropy loss between y_q and x_q to lead the encoder to extract robust embeddings from input queries.

3.4. Attention mechanism for KWS

The speech embeddings, which is a matrix, save the temporal dimension, and we want to know which part contains the keywords to search for them. That is why we introduce the attention mechanism for KWS. In recent works, the concept of attention mechanism is increasingly popular. For example, in Shan, Zhang, Wang, and Xie (2018), the authors proposed an attention mechanism for keyword searching, but that only takes as input the audio embeddings. In contrast, in our version, it takes as input audio and query embeddings simultaneously. Besides, in Zhang et al. (2019), an attention mechanism for speaker verification was proposed. However, that takes two sequences of audio embeddings as input, but in our method, the attention takes a series of audio embeddings and a query embedding vector as input.

The structure of the attention mechanism is shown in Fig. 4.

Firstly, to concatenate the speech and query embeddings, we should repeat the latter to make it the same length of time as the former, as shown in (8)

$$E_q^* = \{e_q, e_q, \ldots, e_q\}, \tag{8}$$

where e_q is the query embeddings (or query feature vector) extracted by query encoder, and there are $T/2$ e_q vectors in E_q^*.

After that, we concatenate E_s and E_q^* along the feature dimension, i.e., $E_{sq} = \{[e_1^{\mathrm{T}}, e_q^{\mathrm{T}}]^{\mathrm{T}}, [e_2^{\mathrm{T}}, e_q^{\mathrm{T}}]^{T}, \ldots, [e_{T/2}^{\mathrm{T}}, e_q^{\mathrm{T}}]^{\mathrm{T}}\}$, where E_{sq} is the result of the concatenation. Then, we pass E_{sq} through a bi-directional LSTM-RNN (Hochreiter & Schmidhuber, 1997) so that we can obtain $E_{sq}^* = \text{BLSTM}(E_{sq})$. Finally, the last layer is an FCL with a sigmoid activation and one single neural unit, which is shown as

$$\alpha = \text{FCL}(E_{sq}^*) = \{\text{FCL}(e_1^*), \text{FCL}(e_2^*), \ldots, \text{FCL}(e_{T/2}^*)\}, \tag{9}$$

where $\alpha = a_1, a_2, \ldots, a_{T/2}$ is the attention weights vector, and e_i^*, $1 \le i \le T/2$ are the vectors in E_{sq}^*.

In summary, given speech and repeated query embeddings, E_s and e_q, the overall function of the attention mechanism can be described as

$$\alpha = \text{Attend}(E_s, e_q), \tag{10}$$

where $\alpha = \{a_1, a_2, \ldots, a_{T/2}\}$ and $a_t \in (0, 1)$ denotes the attention weight for tth time step and E_s, e_q are speech and query embeddings respectively. The value of a_t indicates how likely the attention mechanism believes that a keyword occurs at time step t. As we will demonstrate in the experiment section, the attention mechanism can roughly locate the keywords' time region. However, after obtaining the attention weights, we cannot make a final decision, yet the time lengths of the keywords remain unknown to us. In other words, the question is how we can get the KWS results with the speech embeddings, the query embeddings, and the attention weights without knowing the length of the keywords. This will be tackled by the novel energy scorer, which can consider the above three items in parallel.

3.5. Energy scorer

Till now, we have got the speech embeddings E_s, the query embeddings E_q, and the attention weights α. The attention weights indicate how likely the keywords occur in each time step, but, as we said in the last section, we still cannot make the final decision because we have no prior knowledge about the duration of the keywords.

Here we introduce a novel component called *energy scorer* that combines the above three terms to output a confidence score ranging from zero to one, indicating the probability that the keywords occur in the input speech. The structure of it is shown in Fig. 5.

We first compute the context vector, which is the weighted sum of speech embeddings $E_s = \{e_1, e_2, \ldots, e_{T/2}\}$ according to attention weights $\alpha = \{a_1, a_2, \ldots, a_{T/2}\}$, by

$$c = \sum_t a_t e_t, \tag{11}$$

where $a_t \in (0, 1)$ and c denotes the context vector.

We define the energy of the context vector as

$$\begin{aligned} \text{Energy}_c &:= c^{\mathrm{T}} c \\ &= \left(\sum_t a_t e_t\right)^{\mathrm{T}} \left(\sum_t a_t e_t\right) \\ &= \sum_t a_t^2 e_t^{\mathrm{T}} e_t + \sum_{t_1 \neq t_2} a_{t_1} a_{t_2} e_{t_1}^{\mathrm{T}} e_{t_2} \end{aligned} \tag{12}$$

where $a_t \in (0, 1)$, $t, t_1, t_2 = 1, 2, \ldots, T/2$.

Obviously, Energy_c has an upper bound

$$\begin{aligned} \text{Energy}_c &\leqslant \sum_t e_t^{\mathrm{T}} e_t + \sum_{t_1 \neq t_2} e_{t_1}^{\mathrm{T}} e_{t_2} \\ &= \text{sum}(E_s^{\mathrm{T}} E_s) \\ &=: \text{Energy}_s \end{aligned} \tag{13}$$

where $\text{sum}(A)$ means the summation of elements in matrix A.

The energy ratio r between the context vector and the speech embeddings,

$$r = \text{Energy}_c / \text{Energy}_s. \tag{14}$$

represents the energy proportion of the context vector in speech embeddings.

The larger the energy ratio is, the more big weights from the attention mechanism and the more likely the keyword occurs in the input speech. We apply the concept of energy here to make it more robust against noise. As we know, the energy of noise is usually smaller than that of real speech. Thus, if the attention mechanism outputs some big weights for time steps that are full of noise, Energy_c may not be large enough, compared with Energy_s, to result in a false alarm decision.

Finally, the last thing is to pass the query embeddings through an MLP and get a threshold r_{th} ranging from zero to one so that we can compare it with the energy ratio r, which implies we generate a specific threshold for each query. The reason beneath this is that the query embeddings can be almost precisely transformed to the original input character sequences or corresponding phoneme sequences by the query encoder, meaning that nearly all the useful information is compressed into the query embeddings.

Therefore, we make the final decision by

$$r \underset{Y}{\overset{N}{\lessgtr}} r_{\text{th}}. \tag{15}$$

and the confidence score is $(r - r_{\text{th}} + 1)/2$ resulting in a value ranging from zero to one.

We calculate the binary cross-entropy loss between r and the ground truth target (0 or 1) to train our energy score. Simultaneously, we train the attention mechanism in the last section by binary cross-entropy according to the ground truth target. The final loss function is written as

$$L = \text{CrossEntropy}(r, y_t) + \beta \text{CrossEntropy}(\alpha, y_a), \tag{16}$$

where y_t and y_a are ground truth of the final decision and the attention boundary respectively, and β is a hyperparameter which can be adjusted manually. We set $\beta = 1$ to treat attention weights and final decisions equally in our following experiments.

4. Experiment and analysis

In this section, we will introduce the experiment and give the results mainly on the Assamese dataset. Finally, we will evaluate the proposed model on all the four low resource language datasets mentioned in this article, including Assamese, Bengali, Pashto, and Turkish. We will demonstrate the performance of the proposed model. For comparison, we also evaluate the baseline system (Audhkhasi et al., 2017a, 2017b) on the above language datasets. Specifically, the speech encoder–decoder's performance will be first given, followed by the query encoder–decoder part. Then, we will illustrate the attention mechanism's behavior by visualizing the attention weights for both positive and negative samples. Finally, we will compare the KWS performance of all models and analyze the results.

4.1. Data preparation

We use IARPA Babel-102, Babel-103, Babel-104, and Babel-105 datasets, which are Assamese, Bengali, Pashto, and Turkish, respectively, to train and evaluate the models. To demonstrate the performance on low resource condition, we only use the LimitedLP, which contains about 10 hour training data for each evaluation language. We cut the input speech into one-second pieces for simplicity and locating precision and label each piece of speech by corresponding character or phoneme sequences according to original annotation and lexicon from the datasets. We may treat this as a sliding window, and when evaluating, the test speech can be input to the model second-by-second with a proper search window stride so that the keywords may be detected and located as the speech continues. Given input speech pieces and the corresponding character or phoneme sequences, we can train our speech encoder–decoder.

As to query encoder–decoder, we take the sequences of characters as input using themselves or the corresponding phoneme sequences as labels for training.

As we mentioned above, each piece of speech pairs a character sequence and a phoneme sequence, so every word or phrase can be taken as a positive query for this piece of speech. To generate negative queries, we randomly choose words and phrases that do not occur in that piece of speech, where we should also consider the distribution of all words and phrases in the training corpus. In other words, the more frequently a word or phrase occurs as a positive query, the more regularly it does as a negative query. As for the training of attention mechanism, for each query in a piece of speech, we take the alignment from Kaldi (Povey, Ghoshal, Boulianne, et al., 2011) and generate label sequences (0 and 1 for negative and positive cases, respectively) for each time step of speech embeddings.

4.2. Experiment settings

We apply TensorFlow (Abadi, Agarwal, Barham, et al., 2015), and Keras (Chollet et al., 2015) deep learning framework to implement our proposed and the baseline model.

We frame the input speech with a frame length and shift of 0.025 and 0.010 s, respectively. Then we extract 40 Mel-filter-bank (fbank) features together with their first- and second-order differences to obtain 120-dim features for each frame of input speech finally. The speech encoder we use in this experiment consists of two 1D CNNs with 128 and 256 filters, one 1D max-pooling layer, and two uni-directional GRUs with 256 units. As for speech decoder, there are two types mentioned above, CTC (Graves et al., 2006) and Attention-based Seq2Seq (Luong et al., 2015). The former has a single FCL with softmax activation, whose output will be used to calculate CTC loss. The latter has a two-layer GRU with the same hyper-parameters with speech encoder so that we can take the last hidden states of encoder GRU as the initial states for decoder GRU. Also, the latter has an FCL with softmax activation on the top of GRUs, and we take cross-entropy as the loss function.

Both the proposed and the baseline model compress the input query into a fixed-length vector. We set the number 256 in our experiment. To be more specific, both the encoder and the decoder consist of two GRUs with 128 units, respectively. Besides, the dimension of the embedding layer in the query encoder is 128. On the top of decoder GRUs, there is an FCL with softmax activation, and we use cross-entropy as the loss function.

At last, for the attention mechanism, the number of units of bi-LSTM is 128 per direction. We also use an FCL with one neural unit and a sigmoid activation function to generate thresholds from query embeddings. Given the above settings, the sizes of models are about 12 MegaBytes and 13 MegaBytes for the proposed and baseline model, respectively.

Note that the hyper-parameters in the baseline model have approximately the same number of trainable parameters as the proposed model. We train the three parts of the models separately, i.e., speech encoder–decoder, query encoder–decoder, and attention mechanism together with energy scorer (KWS system for baseline model). We train our model by Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 on a Tesla P100 GRU with 8 GB memory. For each language, the evaluation set contains about 10 hour speech with annotation. The keywords can be divided into two classes, in-vocabulary (IV) and out-of-vocabulary (OOV). Thus, all KWS performance will be given in IV and OOV, respectively.

Table 1
The KWS performance of ACC and AUC with different speech decoders for Assamese IV and OOV.

		CTC	Attention Seq2Seq	Baseline
ACC	IV	0.7061	0.7343	0.6135
	OOV	0.7049	0.7072	0.6042
AUC	IV	0.7737	0.7787	0.6384
	OOV	0.7715	0.7577	0.6320

Table 2
The performance of ACC and AUC with pre-trained and un-pre-trained speech encoder–decoders for Assamese IV and OOV.

		Pre-trained	Un-pre-trained	Baseline
ACC	IV	0.7343	0.6380	0.6135
	OOV	0.7072	0.6318	0.6042
AUC	IV	0.7787	0.6945	0.6384
	OOV	0.7577	0.6930	0.6320

Three steps finish the overall training process. Firstly, we train the speech encoder–decoder. After that, the query encoder–decoder will be trained. Finally, we train the attention mechanism and the energy scorer together with the speech and query encoder fixed.

4.3. KWS performance

The metric that the baseline system used is accuracy (ACC), which is a function of threshold. To marginalize the threshold, we also use the Area Under Curve (AUC) (Bradley, 1997) as another metric to evaluate the models.

Note that we test in-vocabulary (IV) and OOV queries separately. For Assamese, Bengali, Pashto, and Turkish, the number of evaluation IV and OOV queries are (5285, 2088), (4957, 2368), (3228, 975), and (1937, 1234), respectively. Firstly, we analyze the influence of the choice of speech decoder. We compare the KWS performance with CTC speech decoder and Attention Seq2Seq speech decoder, shown in Table 1, where "CTC" and "Attention Seq2Seq" denote the model with CTC speech decoder and Attention-based Seq2Seq speech decoder, respectively.

From Table 1, we can see that the KWS performances with CTC or Attention Seq2Seq are approximately equal, and they both outperform the baseline system.

Then, we illustrate the KWS performance with or without the pre-trained Attention Seq2Seq speech encoder–decoder. Without training the speech encoder–decoder, we only initialize it randomly and fix them when training other parts. The final performances on the Assamese language dataset are shown in Table 2, where "Pre-trained" and "Un-pre-trained" represent the model with and without pre-trained speech encoder–decoder, respectively.

From Table 2, we find that the training of speech encoder–decoder is very important for the proposed model as it leads to extract the spelling information. However, after removing the pre-trained speech encoder–decoder, it still outperforms the baseline system; this is mainly because of the other components still serving well.

As mentioned above, there are two kinds of information, spelling and pronunciation, that we can extract from input speech and queries, according to which type of labels we use for training. Thus, there are four combinations, i.e., C–C, C–P, P–C, and P–P, where P–C denotes that we train speech and query encoder–decoder with phoneme and character sequences, respectively. The performance of these four systems and the baseline system is shown in Table 3.

According to Table 3, we find that "C-C" and "P-P" are slightly better than "C-P" and "P-C", which is because, in the former two models, the speech and query encoders extract the embeddings containing the same kind of information. However, it is not the case for the latter two models. Again, the performances of all four models are better than the baseline system.

Table 3
The performance of ACC and AUC with different combinations of speech and query encoder–decoders for Assamese IV and OOV.

		C-C	C-P	P-C	P-P	Baseline
ACC	IV	0.7343	0.7195	0.6995	0.7385	0.6135
	OOV	0.7072	0.6909	0.6876	0.7068	0.6042
AUC	IV	0.7787	0.7619	0.7651	0.7712	0.6384
	OOV	0.7577	0.7400	0.7467	0.7529	0.6320

Table 4
The KWS performance of ACC with CTC and Attention Seq2Seq speech encoder–decoders for other three language datasets.

		CTC	Attention Seq2Seq	Baseline	SGMM
Bengali	IV	0.7436	0.7450	0.5892	0.6334
	OOV	0.7217	0.7338	0.5787	0.5059
Pashto	IV	0.7615	0.7704	0.6112	0.6277
	OOV	0.7403	0.7377	0.6095	0.5034
Turkish	IV	0.7677	0.7422	0.6156	0.6429
	OOV	0.7153	0.7066	0.6025	0.5104

Table 5
The KWS performance of AUC with CTC and Attention Seq2Seq speech encoder–decoders for other three language datasets.

		CTC	Attention Seq2Seq	Baseline
Bengali	IV	0.7907	0.7873	0.6633
	OOV	0.7815	0.7789	0.6578
Pashto	IV	0.8230	0.8426	0.6554
	OOV	0.8025	0.8204	0.6527
Turkish	IV	0.8501	0.8648	0.6846
	OOV	0.8243	0.8340	0.6745

At last, we evaluate our models on three other language datasets, including Bengali, Pashto, and Turkish. The results are shown in Tables 4 and 5, where we follow the Kaldi recipe (Povey et al., 2011) to train SGMM model for comparison with convention ASR-based method. However, as for the ASR-based method, we only calculate the accuracy metrics (AUC) as how the authors did in the baseline model article (Audhkhasi et al., 2017a, 2017b).

We find that the performances are approximately the same as on the Assamese language dataset. Notably, we also notice that the ASR-based method does not work well in low resource conditions (less than 10 h for training). Besides, note that all the proposed models outperform the baseline model significantly. According to results on various language datasets, the proposed model can be applied for not only Assamese but also Bengali, Pashto, and Turkish as well.

4.4. Speech encoder–decoder

We have mentioned two types of speech encoder–decoders that share the same part of the encoder, and the only difference is the decoder part in . First, for the Attention-based Seq2Seq decoder, we apply accuracy as the metrics. The performance of it trained with character and phoneme labels respectively is shown in Table 6.

Besides, for the CTC-based decoder, the loss curves on training and evaluation set for all testing languages are shown in Fig. 6. Note that we only use character sequences as labels when training CTC-based speech encoder–decoder.

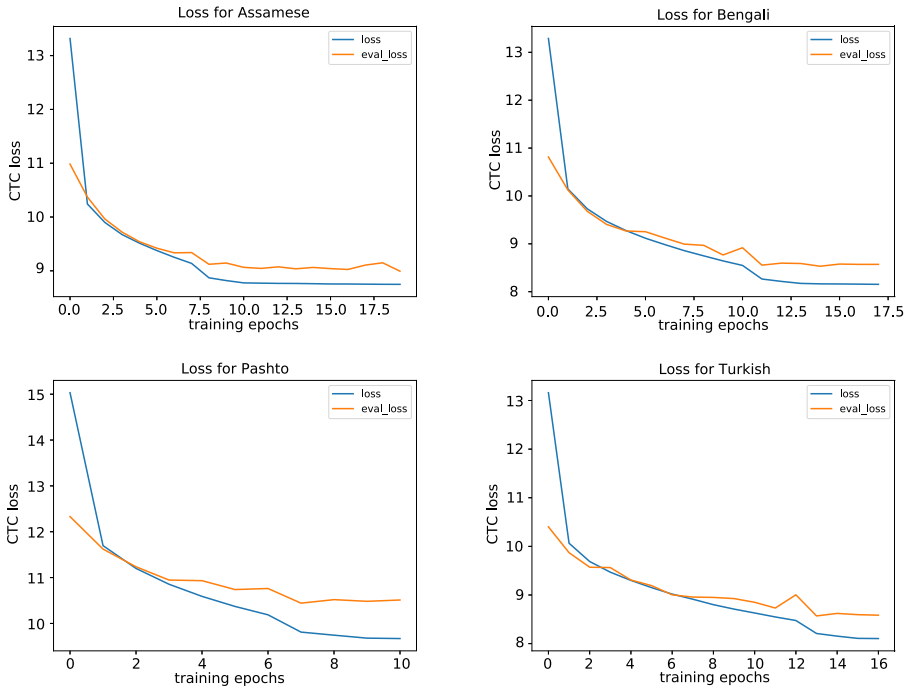

Fig. 6. CTC loss during the training process, where "loss" and "eval_loss" denote CTC loss on training and evaluation set respectively.

Table 6
The prediction accuracy of speech encoder and Attention-based Seq2Seq decoder with character or phoneme sequence labels.

	Character	Phoneme
Assamese	0.7458	0.7355
Bengali	0.7345	0.7598
Pashto	0.6757	0.7446
Turkish	0.7645	0.7881

Table 7
The prediction accuracy of characters and phonemes for the proposed and the baseline (character only) model on the validation set.

	Character	Phoneme	Character-Baseline
Assamese	0.9918	0.9241	0.8347
Bengali	0.9916	0.9631	0.8110
Pashto	0.9967	0.9616	0.8599
Turkish	0.9910	0.9115	0.7430

From Table 6 and Fig. 6, we can see that the speech embeddings, as we expect, contain spelling or pronunciation information to some degree because the decoder can predict the corresponding character or phoneme sequences well just from these speech embeddings.

4.5. Query encoder–decoder

The performances of query encoder–decoders for both the proposed and baseline model on the validation set are shown in Table 7. The validation set is randomly sampled from the whole dataset with a ratio of 10% and not used during the training process. Note that we use character or phoneme sequences as labels leading the encoder to extract spelling or pronunciation information, but only character sequences are used for the baseline model.

Comparing the "Character" and the "Character-Baseline" column in Table 7, we can see that the query encoder–decoder based on Seq2Seq is more efficient than CNN-RNN Character LM in the baseline model, where they both compress the input queries into 256-dim query embeddings. From Table 7, we also note that the speech embeddings contain spelling or pronunciation information according to which kind of labels we use.

Table 8
The KWS performance of ACC and AUC for Assamese IV and OOV.

		CTC	CTC-MLP	Baseline
ACC	IV	0.7061	0.6240	0.6135
	OOV	0.7049	0.6128	0.6042
AUC	IV	0.7737	0.7438	0.6384
	OOV	0.7715	0.7326	0.6320

4.6. Attention mechanism

As we mentioned in Section 3.4, the attention mechanism helps locate the keywords' time region from speech embeddings. The stereotype of attention weights for negative and positive samples are illustrated in Fig. 7. From Fig. 7, we see that the attention weights keep almost zero when processing negative samples while approximately depict the time region of positive queries. As a result, the attention mechanism makes it much easier for the energy scorer to make final decisions.

4.7. Ablation

In this section, we do some ablation experiments, mainly on the energy scorer part. First of all, we show the KWS results after removing the energy scorer and using an MLP instead. In this case, the MLP takes as input the context vector, which is calculated according to (11), which is a vector. This MLP contains only two layers, and the number of units for each layer is 256 and 1, respectively. The output of the MLP will be subtracted by a threshold computed from the query embeddings to get the final decision. If the subtraction result is positive, the system reckons that the keyword occurs in the input speech segment and vice versa.

We only conduct the experiments on the Assamese language dataset. The results are shown in Table 8, where we refer to the

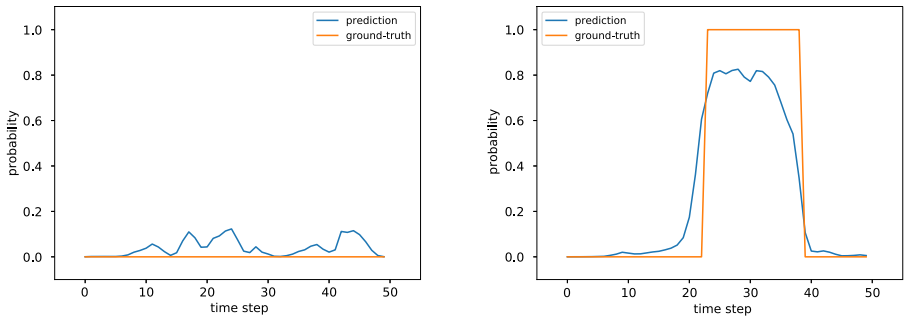

Fig. 7. The attention weights for negative (left) and positive (right) sample.

Table 9
The KWS performance of ACC and AUC for Assamese IV and OOV.

		CTC	CTC-noTH	Baseline
ACC	IV	0.7061	0.6028	0.6135
	OOV	0.7049	0.6011	0.6042
AUC	IV	0.7737	0.6374	0.6384
	OOV	0.7715	0.6225	0.6320

products from Table 1 as a comparison, CTC-MLP denotes the system after the removal of energy scorer. All the settings are kept the same with CTC in Table 1 except the change of energy scorer. We can see that the performance without the energy scorer degrades to some degree but is still slightly better than the baseline model in terms of both of ACC and AUC metrics. We think this is mainly because the energy ratio in (14) is essential for the KWS performance.

Besides, we argue that the threshold generated from the query embeddings is crucial for the KWS performance. We remove the threshold and compare the energy ratio in (14) with the hard threshold of 0.5 to make the final decision. The results are shown in Table 9 with the similar settings with Table 8, where the CTC-noTH denotes the same settings with CTC except the threshold generated from the query embeddings. From Table 9, we notice that without the threshold generated by the query embeddings, the performance of the proposed model can even be worse than the baseline model. Regarding the reason, we analyze it as follows. The lengths of the keywords may vary, and the corresponding speech may also in different lengths. The length of the corresponding speech obviously influences the energy ratio. Notably, in general, the longer the corresponding speech is, the higher the energy ratio is. Thus, it is essential to have a different energy ratio threshold for each and every keyword.

5. Conclusion

We propose an E2E KWS model that consists of four parts, speech encoder–decoder, query encoder–decoder, attention mechanism, and energy scorer. The former two components extract embeddings from the input speech and query by their encoder parts under their decoder parts' supervision. As a result, the speech and query embeddings may contain the spelling or pronunciation information depending on which kind of labels we used when training them. Then the attention mechanism locates the approximate time region of the keywords. Finally, the energy scorer makes final decisions by considering the above three parts' outputs in parallel.

We find that the Seq2Seq method is much more efficient to extract query embeddings than CNN-RNN Character LM in the baseline system. Besides, instead of compressing input speech into a fixed-length vector, keeping the time dimension is crucial for the speech signal. It will result in better performance when extracting the same kind of information from input speech and query. However, even if we use different kinds of embeddings, the KWS performance is still better than the baseline model because of the power of the attention mechanism and the energy scorer.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China, China under Grant No. U1836219, and in part by the National Key R&D Program of China, and the Institute for Guo Qiang of Tsinghua University, China under Grant No. 2019GQG0001, and the Cross-Media Intelligent Technology Project of Beijing National Research Center for Information Science and Technology (BNRist), China under Grant No. BNR2019TD01022.

References

Abadi, M., Agarwal, A., Barham, P., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. URL https://www.tensorflow.org/.

Audhkhasi, K., Rosenberg, A., Sethy, A., Ramabhadran, B., & Kingsbury, B. (2017a). End-to-end ASR-free keyword search from speech. *IEEE Journal of Selected Topics in Signal Processing, 11*(8), 1351–1359. http://dx.doi.org/10.1109/JSTSP.2017.2759726, arXiv:1701.04313.

Audhkhasi, K., Rosenberg, A., Sethy, A., Ramabhadran, B., & Kingsbury, B. (2017b). End-to-end ASR-free keyword search from speech. In *Proceedings of international conference on acoustics, speech and signal processing* (pp. 4840–4844). Institute of Electrical and Electronics Engineers Inc., http://dx.doi.org/10.1109/ICASSP.2017.7953076.

Bahar, P., Brix, C., & Ney, H. (2018). Towards two-dimensional sequence to sequence model in neural machine translation. arXiv preprint arXiv:1810.03975.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. http://dx.doi.org/10.1016/S0031-3203(96)00142-2.

Chen, G., Yilmaz, O., Trmal, J., Povey, D., & Khudanpur, S. (2013). Using proxies for OOV keywords in the keyword search task. In *Proceedings of workshop on automatic speech recognition and understanding* (pp. 416–421). http://dx.doi.org/10.1109/ASRU.2013.6707766.

Chollet, F., et al. (2015). Keras. URL https://keras.io.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. In *Advances in neural information processing systems*: *Vol. 2015-Janua*, (pp. 577–585). arXiv:1506.07503.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ACM international conference proceeding series*: *Vol. 148*, (pp. 369–376). http://dx.doi.org/10.1145/1143844.1143891.

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of international conference on acoustics, speech and signal processing* (pp. 6645–6649). http://dx.doi.org/10.1109/ICASSP.2013.6638947, arXiv:1303.5778.

Hartmann, W., Le, V. B., Messaoudi, A., Lamel, L., & Gauvain, J. L. (2014). Comparing decoding strategies for subword-based keyword spotting in low-resourced languages. In *Proceedings of the annual conference of the international speech communication association* (pp. 2764–2768). URL http://www.isca-speech.org/archive/interspeech_2014/i14_2764.html.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Kamper, H., Anastassiou, A., & Livescu, K. (2019). Semantic query-by-example speech search using visual grounding. In *ICASSP 2019 - 2019 IEEE international conference on acoustics, speech and signal processing* (pp. 7120–7124). http://dx.doi.org/10.1109/ICASSP.2019.8683275.

Kim, B., Lee, M., Lee, J., Kim, Y., & Hwang, K. (2019). Query-by-example on-device keyword spotting. In *2019 IEEE automatic speech recognition and understanding workshop* (pp. 532–538). http://dx.doi.org/10.1109/ASRU46091.2019.9004014.

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, ICLR 2015 - Conference track proceedings*. International Conference on Learning Representations, ICLR, arXiv:1412.6980.

Lengerich, C., & Hannun, A. (2016). An end-to-end architecture for keyword spotting and voice activity detection. arXiv preprint arXiv:1611.09405.

Libovický, J., & Helcl, J. (2017). Attention strategies for multi-source sequence-to-sequence learning. arXiv preprint arXiv:1704.06567.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Conference Proceedings - EMNLP 2015: Conference on empirical methods in natural language processing* (pp. 1412–1421). Association for Computational Linguistics (ACL), http://dx.doi.org/10.18653/v1/d15-1166, arXiv:1508.04025.

Mamou, J., Ramabhadran, B., & Siohan, O. (2007). Vocabulary independent spoken term detection. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 615–622). http://dx.doi.org/10.1145/1277741.1277847.

Mangu, L., Kingsbury, B., Soltau, H., Kuo, H. K., & Picheny, M. (2014). Efficient spoken term detection using confusion networks. In *Proceedings of international conference on acoustics, speech and signal processing* (pp. 7844–7848). Institute of Electrical and Electronics Engineers Inc., http://dx.doi.org/10.1109/ICASSP.2014.6855127.

Neubig, G. (2017). Neural machine translation and sequence-to-sequence models: A tutorial. arXiv preprint arXiv:1703.01619.

Povey, D., Ghoshal, A., Boulianne, G., et al. (2011). The kaldi speech recognition toolkit. In *Proceedings of IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.

Sacchi, N., Nanchen, A., Jaggi, M., & Cernak, M. (2019). Open-vocabulary keyword spotting with audio and text embeddings. *2019-Septe*, In *Proceedings of the annual conference of the international speech communication association* (pp. 3362–3366). http://dx.doi.org/10.21437/Interspeech.2019-1846.

Settle, S., Levin, K., Kamper, H., & Livescu, K. (2017). Query-by-example search with discriminative neural acoustic word embeddings. In *Proc. interspeech 2017* (pp. 2874–2878). http://dx.doi.org/10.21437/Interspeech.2017-1592.

Shan, C., Zhang, J., Wang, Y., & Xie, L. (2018). Attention-based end-to-end models for small-footprint keyword spotting. In *Proceedings of the annual conference of the international speech communication association*: *Vol. 2018-Septe*, (pp. 2037–2041). http://dx.doi.org/10.21437/Interspeech.2018-1777, arXiv:1803.10916.

Sharma, E., Ye, G., Wei, W., Zhao, R., Tian, Y., Wu, J., et al. (2020). Adaptation of RNN transducer with text-to-speech technology for keyword spotting. In *ICASSP 2020 - 2020 IEEE international conference on acoustics, speech and signal processing* (pp. 7484–7488). http://dx.doi.org/10.1109/ICASSP40776.2020.9053191.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*: *Vol. 4*, (pp. 3104–3112). Neural information processing systems foundation, arXiv:1409.3215.

Tanaka, T., & Shinozaki, T. (2018). F-measure based end-to-end optimization of neural network keyword detectors. In *2018 Asia-Pacific signal and information processing association annual summit and conference* (pp. 1456–1461). http://dx.doi.org/10.23919/APSIPA.2018.8659736.

Tanaka, T., & Shinozaki, T. (2019). Efficient free keyword detection based on CNN and end-to-end continuous dp-matching. In *2019 IEEE automatic speech recognition and understanding workshop* (pp. 637–644). http://dx.doi.org/10.1109/ASRU46091.2019.9004021.

Trmal, J., Wiesner, M., Peddinti, V., Zhang, X., Ghahremani, P., Wang, Y., et al. (2017). The Kaldi OpenKWS System: Improving low resource keyword search. In *Proceedings of the annual conference of the international speech communication association*: *Vol. 2017-Augus*, (pp. 3597–3601). http://dx.doi.org/10.21437/Interspeech.2017-601, URL http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0601.html.

Zhang, Y., Yu, M., Li, N., Yu, C., Cui, J., & Yu, D. (2019). Seq2seq attentional siamese neural networks for text-dependent speaker verification. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing* (pp. 6131–6135). IEEE.