Shi *et al. EURASIP Journal on Audio, Speech, and Music Processing* (2021) 2021:27
https://doi.org/10.1186/s13636-021-00212-9

EURASIP Journal on Audio, Speech, and Music Processing

RESEARCH Open Access

Timestamp-aligning and keyword-biasing end-to-end ASR front-end for a KWS system

Gui-Xin Shi, Wei-Qiang Zhang*, Guan-Bo Wang, Jing Zhao, Shu-Zhou Chai and Ze-Yu Zhao

Abstract

Many end-to-end approaches have been proposed to detect predefined keywords. For scenarios of multi-keywords, there are still two bottlenecks that need to be resolved: (1) the distribution of important data that contains keyword(s) is sparse, and (2) the timestamps of the detected keywords are inaccurate. In this paper, to alleviate the first issue and further improve the performance of the end-to-end ASR front-end, we propose the biased loss function for guiding the recognizer to pay more attention to the speech segments containing the predefined keywords. As for the second issue, we solve this problem by modifying the force alignment applied to the end-to-end ASR front-end. To get the frame-level alignment, we utilize a Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) based acoustic model (AM) for auxiliary. The proposed system is evaluated in the OpenSAT20 held by the National Institute of Standards and Technology (NIST). The performance of our end-to-end KWS system is comparable to the conventional hybrid KWS system, sometimes even slightly better. With fusion results of the end-to-end and conventional KWS systems, we won the first prize in the KWS track. On the dev dataset (a part of SAFE-T corpus), the system outperforms the baseline by a large margin, i.e., our system with GMM-HMM aligner has a lower segmentation-aware word error rates (relatively 7.9–19.2% decrease) and higher overall Actual term-weighted values (relatively 3.6–11.0% increase), which demonstrates the effectiveness of the proposed method. For more precise alignments, we can use DNN-based AM as alignmentor at the cost of more computation.

Keywords: OpenSAT20, End-to-end ASR, End-to-end KWS, Force alignment, Biased loss

1 Introduction

Accurate spoken term detection (STD), also named keyword searching (KWS), is a vital downstream application of automatic speech recognition (ASR). For information retrieval in massive data, people not only want to know the presence or absence but also the specific time region (we name it "timestamp") of the keyword in the utterance. The typical KWS pipeline is based on a lattice obtained from a Large Vocabulary Continuous Speech Recognition (LVCSR) system, wherein a neural acoustic model (AM) and a word-based language model (LM) are both applied to generate the word-level lattices by decoding [1]. Consequently, KWS can be performed mainly on lattices.

During the last decade, end-to-end ASR stimulated great interest and achieved many successes. For training, end-to-end ASR systems only require the segmented speech and the corresponding text. There are two popular approaches for end-to-end ASR systems, i.e., connectionist temporal classification (CTC) [2–4] and attention mechanism [5–9]. CTC introduces a blank symbol to match the output sequence length with the input sequence [10], and optimizes the sum over all possible output sequences instead of each individual output label. The attention-based encoder-decoder architecture such as the Listen, Attend, and Spell (LAS) [11] system includes an encoder which is analogous to a conventional acoustic

*Correspondence: wqzhang@tsinghua.edu.cn
Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China

model, an attender that acts as an alignment model, and a decoder that is analogous to a language model in the conventional system [9]. The joint CTC/attention ASR architecture combines these two approaches and has become a popular trend for end-to-end ASR [12–17].

Meanwhile, deep neural networks (DNNs) have become more and more popular for the KWS task. In [18], Chen et al. propose the famous DeepKWS framework, which achieves better performance than the conventional methods. Following DeepKWS, more advanced architectures have been successfully applied to the KWS task, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), deep residual networks (ResNets), and Graph Convolutional Networks (GCNs) [19–22]. However, the approaches mentioned above have two disadvantages: (1) they are ASR-free and designed for a small number of the keywords of interest, and (2) they neglect the timestamps of keywords. Some people work on ASR-free multi-keyword detection [23–25], but the timestamps of keywords are still neglected. Nonetheless, in some practical applications, the timestamps of a large amount of keywords are still required. In addition, the ASR-free KWS system is not suitable for scenarios with higher flexibility (keywords will change). When changing or adding keywords, ASR-based KWS only needs finetuning to meet the new requirements, but ASR-free KWS has to retrain a new model from scratch.

In this paper, we aim to develop an end-to-end ASR-based KWS system to overcome the two disadvantages. The main contributions of the proposed system are (1) we propose a biased training approach for an ASR front-end to improve the KWS by paying more attention to predefined keywords than the normal data, and (2) we use an auxiliary AM to force align the ASR recognition text with time, therefore the keyword region can be detected.

The rest of the paper is organized as follows. In Section 2, we introduce the OpenSAT20 and our Speech Activity Detection (SAD) frontend. Then we describe the proposed KWS system in Section 3. A series of experiments is shown in Section 4. Finally the conclusion is made in Section 5.

2 Background

2.1 OpenSAT20

The OpenSAT Series for speech analytic systems evaluations started last year (2019) and OpenSAT20 is the second. The OpenSAT20 Evaluation makes use of simulated public safety communications spoken in English and offers three evaluation tasks: SAD, KWS, and ASR. Participants may choose to participate in one, two, or all three of the tasks. In this paper, we focus on the ASR and KWS tasks, although our team completed all the tasks.

Summary statistics for the OpenSAT20 data used herein are given in Table 1. NIST did not release the transcripts of the evaluation dataset, so we evaluate our system based on the dev dataset. The development keyword lists consist of 559 single and multi-word query terms, as shown in Table 2. It should be noted that more than half of the segments do not contain keywords.

2.2 Speech activity detection frontend

In this work, our SAD module is based on a regression approach to speech enhancement based on DNN [26, 27]. Our SAD system is shown in Fig. 1. In order to get a better performance in low signal-noise ratio conditions, the raw input audios are enhanced by a pre-trained speech-enhancement module, resulting in enhanced audio signals. We extract FilterBank (FBank) features [28] of raw audios and enhanced audios before feeding them into subsystems. The subsystems contain the BUT phoneme recognizer (BUT) [29], Neural Network Classifiers and Subband OSF-Based VAD (SOV). BUT is a Hidden Markov Model/Artificial Neural Network (HMM/ANN) system, which implements the Temporal Pattern (TRAP) system [30] and utilizes a new Split Temporal Context (STC) system[29]. As for the Neural Network Classifiers, we utilize a convolutional recurrent neural network (CRNN) and a RNN [26, 27]. The SOV is based on the determination of the speech/nonspeech divergence by means of specialized order statistics filters (OSFs) working on the subband log-energies [31]. Results of different subsystems are fusioned by a model ensemble module, which is computed as follows:

$$s = \sum_{i=1}^{N} w_i \times s_i \tag{1}$$

where N is the number of sub-systems of SAD, w_i denotes the weight of each system, and s_i denotes the confidence score output by each subsystem. The weight of each system w_i is systematically calculated. We introduce BOSARIS toolkit [32] to the fusion module to

Table 1 OpenSAT20 data partition statistics

Corpus	#File	#Hour	Trans
SAFE-T train	194	80	Avaliable
SAFE-T dev	100	5	Avaliable
SAFE-T eval	173	8.7	Not avaliable

Table 2 Examples of OpenSAT20 keywords

PET HAS BEEN	KID'S	HOPEFULLY	GREEN
OOF MAN	FOCUS	ENTRANCE	WAIT
SHORT LITTLE	SAVED	REMOVING	SPREAD

train a model whose inputs are probabilities of speech existence in every frame output by subsystems and output is the optimized probability of speech existence in every frame. Actually the trained model is a vector with the weight of each subsystem and the corresponding offset.

In the presence of noise, recognition of the voice signal is inevitably weakened, often requiring speech enhancement. Therefore, as shown in Fig. 2, some "holes" that appear in speech segments in output need to be patched, and some boundaries need to be expanded. Each audio is divided into several segments marked speech or nonspeech in our system output, and this is just like a one-dimensional binary image. Therefore, our one-dimensional dilation-erosion algorithm is dilating or eroding speech segments of output. Here we give the concept of SAD parameter α, i.e. the length of eroding speech segments. Our one-dimensional dilation-erosion algorithm can be expressed as

$$\begin{aligned} t_{beg} &= \max[\,0, t_{beg} - \alpha] \\ t_{end} &= \min[\,T, t_{end} + \alpha] \end{aligned} \tag{2}$$

where α is a scalar in the SAD postprocess and varies from 0.1 to 0.5 in this work, t_{beg}, t_{end} denote the beginning and ending times of a detected speech segment, and T denotes the audio length. The mechanism of how α affects SAD output is also depicted in Fig. 2.

The metric to evaluate our SAD system is the detection cost function value ($DCF(\theta)$), which is also the official evaluation metric in OpenSAT19 and OpenSAT20 [33]. $DCF(\theta)$ is computed by the false positive rate (P_{FP}) and false negative rate (P_{FN}) as follows:

$$P_{\text{FP}} = \frac{\text{total FP time}}{\text{annotated total nonspeech time}} \tag{3}$$

$$P_{\text{FN}} = \frac{\text{total FN time}}{\text{annotated total speech time}} \tag{4}$$

$$DCF(\theta) = 0.75 \times P_{\text{FN}} + 0.25 \times P_{\text{FP}} \tag{5}$$

where θ denotes a given system decision-threshold setting [33]. A good SAD system should achieve a small value of $DCF(\theta)$ in (5). Besides, there is a collar with 0.5 s duration between every speech segment and nonspeech segment, and none of the collar segments are scored when calculating $DCF(\theta)$.

The output result of the SAD system will affect the end-to-end ASR performance, which will be specifically described and discussed in Section 4.

3 Proposed end-to-end ASR-based KWS system

Generally, an ASR model aims to recognize all the words in the speech while a KWS system focuses only on keywords. Furthermore, an end-to-end ASR model usually does not align the recognized text with timestamp information while the KWS model cares about whether and when the keywords appear [1, 34, 35]. In order to alleviate the above two problems and further improve the KWS performance, we proposed a biased-training ASR-based KWS system and utilize an auxiliary GMM-HMM

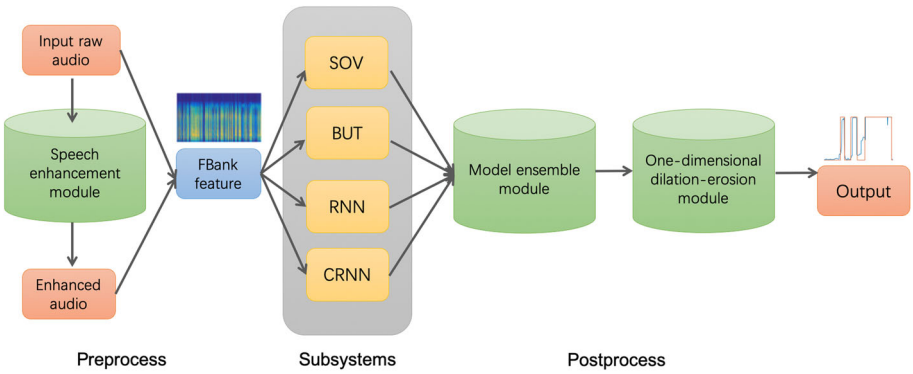


Fig. 1 Overview of our SAD architecture used in OPENSAT20

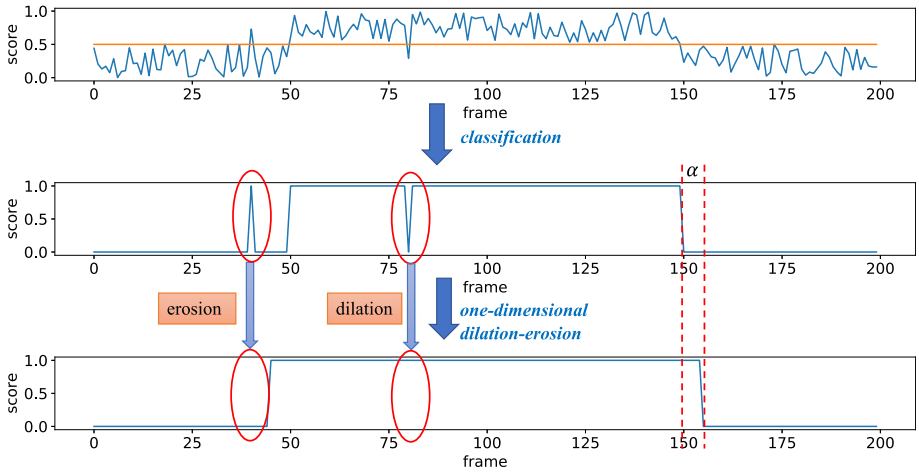


Fig. 2 One-dimensional dilation-erosion

model for alignment. Our proposed KWS architecture is depicted in Fig. 3.

3.1 Biased-loss ASR front-end for KWS

In this section, we will introduce our proposed model by first reviewing our baseline ASR model, and then detailing our improved end-to-end ASR.

We give an overview of our ASR baseline and summarize a few key components of the Transformer model. Transformer, a popular neural network architecture, has been more and more popular in ASR tasks. For full details, please refer to [11, 36–38].

Transformer was proposed by [11] as an encoder-decoder sequence transduction model, in which encoder has N_e repeated building blocks and decoder has N_d repeated building blocks, as shown in Fig. 4. In addition to the encoder-decoder structure, the conventional speech Transformer model [11] has a multi-head attention module, as explained in Equations (6)-(8). Given an input sequence of acoustic features $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ the encoder maps $\mathbf{X}$ to a sequence of hidden states $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$. Given $\mathbf{H}$, the decoder generates the outputs sequence $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ step by step, where n is the frame number of the input sequence, m is the length of the output sequence. Both encoder and decoder are stacked with attention and position-wise feed-forward networks and replace the commonly used recurrent layers with self-attention layers to learn the input representation by scaled dot-product attentions. The multi-head attention strategy is utilized by concatenating multiple self-attention outputs together to learn different subspaces concurrently:

$$\text{Attention}\,(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

$$\text{MultiHead}\,(Q,K,V) = \text{Concat}\,(\text{head}_1,\ldots,\text{head}_h)\,W^H \tag{7}$$

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), i = 1,\ldots,h \tag{8}$$

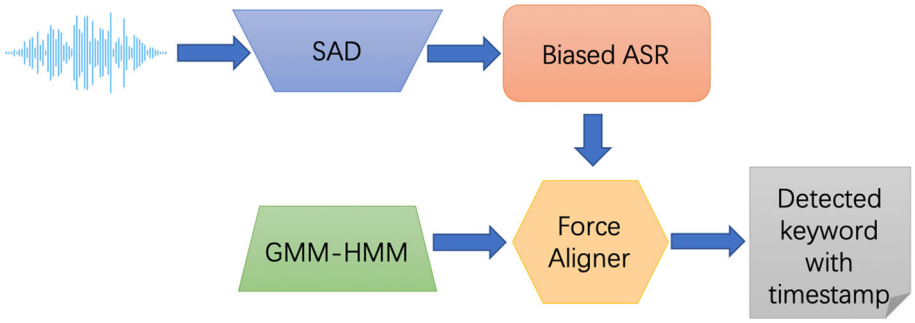


Fig. 3 Our proposed KWS architecture

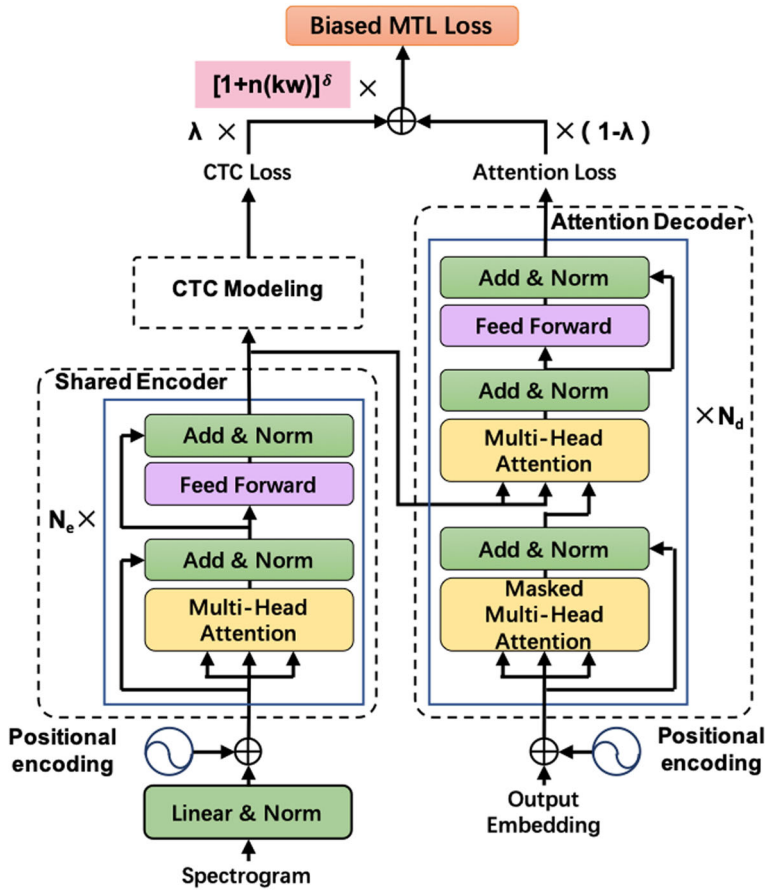


Fig. 4 An overview of our biased-loss ASR front-end based on Transformer [11]

where head$_i$ is the output of the i-th attention head, and h is the number of attention heads; $Q \in \mathbb{R}^{n_q \times d_q}$, $K \in \mathbb{R}^{n_k \times d_k}$, and $V \in \mathbb{R}^{n_v \times d_v}$ represent query, key and value matrices, respectively; n_* and d_* are sequence lengths and feature dimensions, respectively; d_{model} represents the number of input feature dimensions; $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $W^H \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are corresponding weight matrices, and $d_k = d_q = d_v = d_{\text{model}}/h$ [38].

After the multi-head attention network, there is a positionwise feedforward network with rectified linear unit (ReLU) activation:

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \tag{9}$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{\text{model}}}$ and the biases $b_1 \in \mathbb{R}^{d_{ff}}, b_2 \in \mathbb{R}^{d_{model}}$ [36]. In each block, the *FFN* have different learnable parameters.

The ASR architecture used in our work is based on the advanced joint CTC/attention models. According to [17], using CTC along with the attention decoder can improve robustness since the forward-backward algorithm of CTC can enforce monotonic alignment between speech and label sequences during training. That is, the forward-backward algorithm in CTC helps to speed up the process of estimating the desired alignment rather than solely depending on data-driven attention methods to estimate the desired alignments in long sequences. Moreover, CTC also assists the network in training speed. The idea of this architecture is to use CTC as an auxiliary objective function to train the attention-based seq2seq network. Specifically, we combine with the Transformer architecture [11] that uses self-attention.The objective to be maximized is a logarithmic linear combination of the CTC and attention objectives:

$$\mathcal{L}_{\text{MTL}} = \lambda \log p_{\text{ctc}}(C \mid X) + (1-\lambda) \log p_{\text{att}}(C \mid X) \tag{10}$$

where λ is the CTC module weight, C is the reference, and $\log p_{\text{ctc}}(C \mid X)$ and $\log p_{\text{att}}(C \mid X)$ are CTC and Transformer objectives, respectively.

As aforementioned in Section 2, the amount of data containing keywords is typically much less than the normal speech for ASR due to the cost of data acquisition, i.e.,

Table 3 Keyword occurrence statistics in training set

Keyword	KW-id	#Occurrence	Keyword	KW-id	#Occurrence
PET	KW-0240	6	ANY	KW-0030	4
HAS	KW-0031	12	POSSIBLE	KW-0025	19
BEEN	KW-0048	4	ANY POSSIBLE	KW-0358	0
PET HAS BEEN	KW-0537	0	ENTRANCE	KW-0199	10
OOF	KW-0239	5	GREEN	KW-0011	72
MAN	KW-0091	58	WAIT	KW-0036	48
OOF MAN	KW-0436	0	SPREAD	KW-0052	56
KID'S	KW-0089	11	SAVED	KW-0016	47
FOCUS	KW-0177	1	HOPEFULLY	KW-0088	1

the positive samples that containing at least one keyword will be much less than the negative samples. Here "sample" means handcrafted speech segment in training set. For some uncommon keywords, the number of positive samples is very small, even only one or two. Among 49,989 training samples, only 8.27% of training data (4048 samples) contain keyword(s). Table 3 offers some representative statistical information on positive samples of different keywords. Although some multi-word query terms, e.g., "PET HAS BEEN" and "ANY POSSIBLE", do not occur in the training set, the single words that consist of multi-word query terms are also in the keyword list and there are several corresponding positive samples. The distribution of keywords in the training dataset is sparse, how to use those precious examples efficiently is the key for the KWS task. When facing such a large class imbalance during training, a KWS system may lead to a degenerate model. Since the majority of samples do not contain keywords, the information learned from segments containing keywords is of vital importance.

To further improve the performance of the end-to-end ASR front-end, we propose the biased loss function to guide the recognizer to pay more attention to the segments containing the predefined keywords. Unlike a normal ASR system, we emphasize the recognition of predefined keywords. That is, we want to learn more carefully on positive samples since the aforementioned imbalance leads to inefficient training.

Here, we define a bias factor δ and the biased loss function as follows:

$$\mathcal{L}_{\mathrm{WL}} = [1 + n(kw)]^{\delta} \cdot \mathcal{L}_{\mathrm{MTL}} \tag{11}$$

where $n(kw)$ is the number of keywords in this utterance and $\mathcal{L}_{\mathrm{MTL}}$ is the original training loss. In particular, we

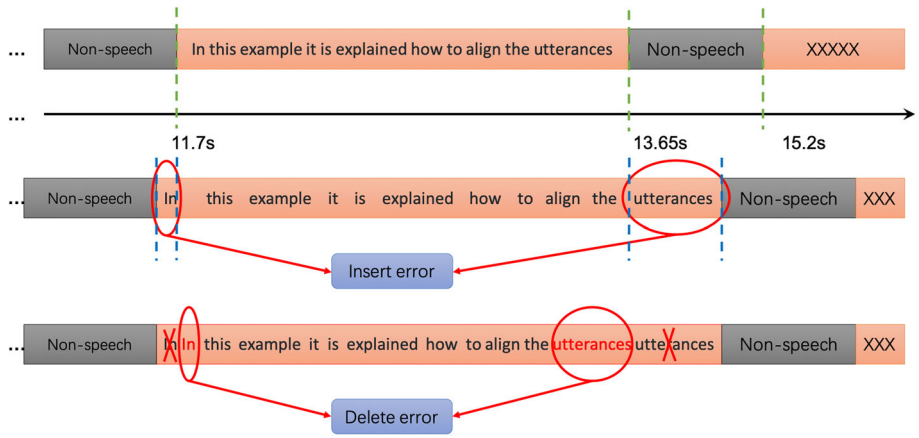
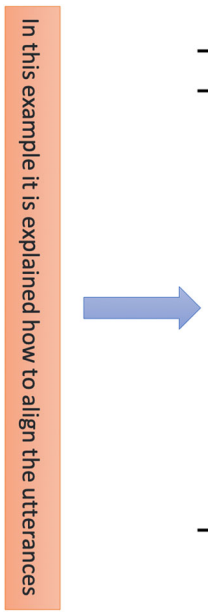


Fig. 5 An example how alignment affects segmentation-aware WER. Orange blocks represent handcrafted speech segments (above) or SAD-classfied speech segments (middle and bottom)



Word	Time beginning	Time duration
in	11.7	0.1
this	11.8	0.12
example	11.92	0.35
it	12.27	0.09
is	12.36	0.08
explained	12.44	0.44
how	12.88	0.13
to	13.01	0.20
align	13.21	0.10
the	13.31	0.09
utterances	13.40	0.25

Fig. 6 Example of system output CTM file. Timestamps consist of the starting time and KW duration

focus the model on the positive samples by up-weighting the contribution of positive samples duringz training.

3.2 Selection of the force alignment module for ASR and KWS

There are several tools to implement the force alignment. Montreal Force Alignment (MFA) and Gentle toolkits are built on top of Kaldi toolkit [39], while Munich Automatic Segmentation (MAUS) uses HTK [40]. These toolkits first transform the given transcript into a graph representing different sequences of phones by applying predefined rules. Afterwards, the actual alignment is estimated by finding the most probable path using a set of HMMs and pretrained acoustic models.

Usually, an end-to-end ASR outputs text without alignment information. Attention-based architectures, such as Transformers, encode the feature sequence into embedding, which is not explicitly associated with frames, so alignment information is not available. In addition, the CTC-based algorithm has an alignment function and can output alignment information. Recently, Ludwig Kürzinge et al. proposed to use a CTC-based network for the segmentation task and it outperforms the other existing segmentation tools [41]. It should be noted that here "segmentation task" in [41] is similar to the SAD task.

The authors proposed to use a CTC-based network to extract proper speech segments in the presence of additional unknown speech or non-speech segments at

Table 4 A comparison of using different bias factor. WERs and CERs are in percent and evaluated on the manually segmented dev dataset

bias factor δ	**Model 1: base + FT**	**Model 2: FT all epochs**
	WER / CER	**WER / CER**
0 (baseline)	9.2/10.5	9.2/10.5
1/4	9.0/10.2	9.2/10.4
1/3	9.0/10.2	9.2/10.5
1/2	9.0/10.2	9.2/ 10.5
1	9.1/10.4	9.2/10.4
2	9.0/10.3	9.1/10.3
4	9.1/10.3	10.1/11.4

Table 5 SAD performance on different α

	SAD α 0.1	SAD α 0.2	SAD α 0.3	SAD α 0.4	SAD α 0.5
P_{FN}	0.1884	0.1298	0.0940	0.0716	0.0570
P_{FP}	0.0181	0.0251	0.0317	0.0378	0.0440
$DCF(\theta)$	0.1458	0.1037	0.0784	0.0631	0.0537
# segments	3891	3262	2694	2191	1878

the beginning or end of the audio recording. It uses a CTC-based end-to-end network that is trained on already aligned data beforehand, e.g., as provided by a CTC/attention ASR system. For a given audio recording, the CTC network generates frame-based character posteriors. Therefore, it is possible to compute all possible maximum joint probabilities for aligning the text via dynamic programming. In [41], the CTC network is used for utterance-level segments and outperforms other existing segmentation tools.

However, because CTC is also a kind of sequence modeling, the CTC loss is actually the sum of the probabilities of multiple alignment paths. The objective of CTC loss is not for obtaining the alignment result and during decoding, the path with the highest probability usually is taken, so the real path cannot be completely determined, and the timestamp information obtained is not accurate enough. In addition, the existence of the blank symbol aggravates this situation.

Based on the previous analysis, we take the CTC-segmentation algorithm as our baseline of force alignment module to output the word-level segmentation. In other words, we use our SAD module in Section 2 and output segmented utterances for the ASR system, then the CTC-segmentation algorithm computes word-level text-time alignment.

In this paper, inspired by the idea of MFA as well as MAUS, we will train an auxiliary AM using training data to perform the alignment on the dev dataset, although we can utilize the CTC network in our ASR front-end to output alignment.

We train a triphone GMM-HMM system as our auxiliary AM to provide time information in order to perform KWS, which needs the accurate occurrence time, i.e., the timestamp, of keywords. We first prepare training data with annotated segments. Secondly, we train a conventional ASR-based on a triphone GMM-HMM AM module. The primary objective of the conventional ASR is to build a statistical model to infer the text sequences W from a sequence of feature vectors X, as explained in eq 13:

$$\mathrm{W}^* = \arg\max_{\mathrm{W}} P(\mathrm{X} \mid \mathrm{W}) \cdot P(\mathrm{W})/P(\mathrm{X}) \tag{12}$$

where $P(\mathrm{X} \mid \mathrm{W})$ represents a generative AM and $P(\mathrm{W})$ represents a generative Language Model (LM). We have Equation (13) after applying Bayes' Theorem:

$$\mathrm{W}^* = \arg\max_{\mathrm{W}} P(\mathrm{X} \mid \mathrm{W}) \cdot P(\mathrm{W}) \tag{13}$$

The function of the acoustic model is to model a sequence of feature vectors given a sequence of phones instead of words. But we will continue the use of the notation $P(\mathrm{X} \mid \mathrm{W})$ for the acoustic model. The distribution of features for phones can be modeled with a GMM, which can be learned from training data. The transition between phones and the corresponding observable is modeled with

Table 6 A comparison of using different bias factors and SAD dilation-erosion factors ASR Model 1. WERs are in percent and evaluated on the SAD segmented dev dataset. "G" and "C" represent GMM-HMM aligner and CTC aligner, respectively

bias factor	SAD α 0.1	SAD α 0.2	SAD α 0.3	SAD α 0.4	SAD α 0.5
δ	G	G	G / C	G / C	G / C
0 (baseline)	13.7	13.3	12.9/14.0	12.9/15.1	13.5/16.7
1/4	13.7	13.2	12.6/13.9	12.8/15.1	13.0/16.7
1/3	13.7	13.3	12.6/14.0	12.8/15.1	13.3/16.8
1/2	13.5	13.3	12.6/13.8	12.8/15.0	13.0/16.6
1	13.6	13.2	12.6/13.8	12.7/15.0	12.9/16.8
2	13.6	13.3	12.7/13.8	12.8/15.0	13.0/16.8
4	13.6	13.1	12.7/13.8	12.8/14.9	13.1/16.5

Table 7 A comparison of using different bias factors and SAD dilation-erosion factors on ASR Model 2. WERs are in percent and evaluated on the SAD segmented dev dataset. "G" and "C" represent GMM-HMM aligner and CTC aligner, respectively

bias factor	**SAD α 0.1**	**SAD α 0.2**	**SAD α 0.3**	**SAD α 0.4**	**SAD α 0.5**
δ	G	G	G / C	G / C	G / C
0 (baseline)	13.7	13.3	12.9/14.0	12.9/15.1	13.5/ 16.7
1/4	14.0	13.7	12.9/13.9	12.8/15.1	13.5/17.0
1/3	14.0	13.8	12.9/14.2	13.1/15.5	13.5/17.0
1/2	13.8	13.4	13.0/14.3	13.1/15.3	13.5/16.8
1	14.2	13.7	12.9/13.8	12.7/15.0	12.3/16.8
2	13.8	13.1	12.8/15.1	12.7/15.0	13.6/16.7
4	14.9	14.0	13.7/16.4	13.9/17.2	14.7/17.8

the HMM. The HMM model consists of hidden variables and observable variables, and the observable variables can be represented by features extracted from the corresponding audio frames. Lastly, given a text sequence, we align it with features by searching for the feature sequence with the maximum likelihood:

$$\mathrm{X}^* = \arg\max_{\mathrm{X}} P(\mathrm{X} \mid \mathrm{W}) \tag{14}$$

The GMM-HMM aligning is a special decoding operation with the text sequence provided from our end-to-end ASR front-end. The Weighted Finite-State Transducer (WFST)[42] is used to build the state map for decoding. Finally, we can obtain the decoding text results with annotated timestamps.

4 Experiment set up and results

In this section, we evaluate how well the proposed system recognizes and aligns word-wise text and audio.

4.1 Experiment dataset

To evaluate our network model, we participated in the OpenSAT20 Evaluation. The OpenSAT20 Evaluation comprised data from the simulated Public Safety Communications (PSC) data, which are simulated public safety communications spoken in English from the SAFE-T corpus that was collected by the Linguistic Data Consortium (LDC) and initially made available for the OpenSAT19 evaluation. The data have the following characteristics: (1) varying background noise types, (2) varying background noise levels, (3) speech in the presence of noise, and (4) speech under stress induced situations. In addition, we use the whole Librispeech corpus as the additional data to increase the system robustness.

Evaluation is performed on the dev dataset of the SAFE-T dataset, which consists of 100 180-s-length recordings from 25 unique speakers. Each recording consists of several speech and non-speech segments. This corpus contains labeled sentence-length utterances, each with the information of start and end of its segment in the audio recording. Note that there are no transcripts of the OpenSAT20 eval dataset so we evaluate the system performance on manually segmented as well as SAD segmented dev datasets.

4.2 Performance metric computation

In this paper, we use the OpenSAT20 metrics to measure our system's performance. ASR performance is measured by the segmentation-aware word error rate (WER). Here "segmentation-aware" means the text must be aligned in the segmentation block, otherwise result in insertion error. Segmentation-aware WER is a more strict metric than the normal WER. As depicted in Figs. 5 and 6, the word "in" occurs from 11.7 s to 11.8 s, if "in" is recognized rightly but aligned to a region before 11.7 s, insertion and deletion errors appear simultaneously.

KWS performance is measured by the Term-Weighted Value (TWV), a function of false-positive (false-alarm) and false-negative (missed detection) rates for the keyword and is determined by comparison to the reference. An Actual Term-Weighted Value (ATWV) is a measure of keyword detection performance at a given system's threshold setting (φ) and is calculated from the system decision threshold setting determined by the developer. The TWV is defined as

$$\mathrm{TWV}(\varphi) = 1 - [\,\mathrm{P}_{\mathrm{FN}}(\varphi) + \beta \cdot \mathrm{P}_{\mathrm{FA}}(\varphi)] \tag{15}$$

As for choosing φ, we choose a decision threshold for the "Actual Decisions" to optimize our term-weighted value, and all the "YES" system occurrences called the ATWV. Information for accessing and downloading the scoring software is available at the OpenSAT website. The scoring protocol is the "Keyword Occurrence Scoring" protocol that evaluates the system accuracy. For more detailed information, see the DRAFT KWS16 KEYWORD SEARCH EVALUATION PLAN, PLAN (KWS16-evalplan-V04), https://www.nist.gov/document-194.

Table 8 A comparison of using different bias factors and SAD dilation-erosion factors on ASR Model 1. ATWV values are evaluated on the SAD segmented dev dataset. "G" and "C" represent GMM-HMM aligner and CTC aligner, respectively

bias factor	SAD α 0.1	SAD α 0.2	SAD α 0.3	SAD α 0.4	SAD α 0.5
δ	G	G	G/C	G/C	G/C
0 (baseline)	0.5673	0.5783	0.5917/0.5593	0.5906/0.5542	0.5973 / 0.5656
1/4	0.5919	0.6077	0.6028/0.5795	0.6086/0.5741	0.5896/0.5488
1/3	0.5919	0.6050	0.6086/0.5872	0.6058/0.5730	0.5951/0.5591
1/2	0.5955	0.6031	0.6092/0.5858	0.6072/0.5766	0.5910/0.5550
1	0.5950	0.6007	0.6077/0.5844	0.6140/0.5796	0.5972/0.5602
2	0.5920	0.6107	0.6102/0.5809	0.6128/0.5822	0.5886/0.5521
4	0.5788	0.6024	0.6012/0.5749	0.5993/0.5688	0.5865/0.5455

4.3 Configuration of ASR model

We combine 80-dim FBank vectors with 3-dim pitch vectors as the input features for our E2E systems. 5002 word-piece units are used as output labels. In our system, we use the original lexicon provided in the Librispeech pack. It is worth noting that not all the words in the lexicon appear in the training transcripts, but we keep all the words in order to maximize our vocabulary size to reduce the OOV rate on the dev dataset.

As the alignments of the training dataset have been done manually, we use the well-segmented wave file for training. The ASR and LM training are implemented in the ESPnet toolkit [43]. The configuration of the pre-trained model (termed as "pytorch large Transformer with specaug (4 GPUs) + Transformer LM (4 GPUs)" in the ESPnet toolkit [43], model link: https://drive.google.com/open?id=17cOOSHHMKI 82e1MXj4r2ig8gpGCRmG2p) is briefly as follows: the Transformer-based model consists of an encoder made up of 12 attention blocks and a decoder made up of 6 attention blocks; within each attention block, the 8-head attention layer is 2048-dim; it has been trained on the whole 960-h Librispeech dataset; the amount of the parameters is about 230M. It should be noted that we just utilize the pre-trained ASR model. Our LM is trained from scratch. Our LM is trained with two parts: transcripts of the training data and the Librispeech dataset. We train a large LM consisting of four 2048-dimension RNN layers, which is pre-trained on Librispeech corpus data and then finetuned on the SAFE-T training corpus data. LM weight in decoding is 0.1.

In order to enhance the robustness of our ASR systems, we adopt speed perturbation as well as spec-augmentation [44] methods. For each audio file in the training dataset, we use both 1.1x and 0.9x speed perturbations. During training, Adam optimizer is used to train our model. We choose warmup = 250,000, dropout = 0.1, labelsmooth = 0.1 for all experiments. In order to do evaluation, we save the model parameters of 3 best epochs according to the validation sets and average them at the end of training. We use WER as the evaluation metric. The beam search is carried out with a beam size of 10 and length penalty of 0.6. LM is used during decoding with weight 0.1.

Table 9 A comparison of using different bias factors and SAD dilation-erosion factors on ASR Model 2. ATWV values are evaluated on the SAD segmented dev dataset. "G" and "C" represent GMM-HMM aligner and CTC aligner, respectively

bias factor	SAD α 0.1	SAD α 0.2	SAD α 0.3	SAD α 0.4	SAD α 0.5
δ	G	G	G / C	G / C	G / C
0 (baseline)	0.5673	0.5783	0.5917/0.5593	0.5906/0.5542	0.5973/0.5656
1/4	0.5808	0.6026	0.5935/0.5778	0.6089/0.5786	0.5923/0.5699
1/3	0.5891	0.5919	0.6009/0.5414	0.5889/0.5498	0.5975/0.5707
1/2	0.5744	0.5847	0.5830/0.5402	0.5789/0.5389	0.5901/0.5617
1	0.5753	0.5899	0.5910/0.5710	0.6077/0.5706	0.5997/0.5611
2	0.5805	0.5801	0.5956/0.5543	0.5749/0.5415	0.5657/0.5288
4	0.5445	0.5751	0.5837/0.5525	0.5699/0.5271	0.5467/0.4965

4.4 Configuration of auxiliary acoustic model

We apply Speaker Adapted Training (SAT) twice to provide the final GMM-HMM system with 4200 leaves and 40,000 Gaussians. Besides, we transform the features before SAT by splicing across 3 frames from both context sides, using the Linear Discriminant Analysis (LDA) for dimensionality reduction, and estimating over multiple iterations. This module is implemented in Kaldi toolkit [45]. The GMM-HMM aligner used in the OpenSAT20 Evaluation is trained by 4 iteration, named as "G". A GMM-HMM aligner trained by 3 iteration is named as "G3". The "T" aligner means an AM trained by the CNN-TDNNF network using the recipe of OpenSAT20 dataset, which is released in Kaldi toolkit recently (https://github.com/kaldi-asr/kaldi/tree/master/egs/opensat20/s5).

4.5 Results of the ASR task

4.5.1 Results on hand-labeled segments on the Dev set

As explained in section 2.2, one of the main motivations for force alignment is to match the text to audio in a robust manner. We first provide the benchmark results on hand-labeled segments on the dev dataset in Table 4. "Model 1: base + FT" represents only finetuning the last 3 epochs using biased loss fuction while "Model 2: FT all epoch" means all 28 epochs are finetuned by our new biased loss function.

It can be seen that our new training method can be beneficial to ASR and Model 1 outperforms Model 2 slightly. When bias factor δ is not too large, our improved ASR performs robustly.

4.5.2 Results on SAD segmented Dev set

Table 5 shows our SAD system performance on different factor α. We recognize these five sets segmented dev dataset and computer corresponding segmentation-aware WERs and ATWVs of Model 1 and Model 2, which are summarized in Tables 6, 7, 8, and 9.

We evaluate the impact on text aligned by CTC and GMM-HMM based aligners. In order to evaluate the impact of segmentation error on ASR, we first calculate segmentation-aware WER (the word should be recognized and aligned to right segment file) of different parameters and the results are shown in Tables 6 and 7. The CTC aligner fails when the SAD factor α is less than 0.2. Correspondingly, the results of ATWV are summarized in Tables 8 and 9.

According to Tables 4, 5, 6, and 7, Model 1 has better overall performance. This phenomenon shows that the loss function has considerable influence on training

Table 10 A comparison of using different bias factors and SAD dilation-erosion factors on ASR Model 1. ATWV values are evaluated on the SAD segmented dev dataset. "T", "G" and "G3" represent CNN-TDNNF aligner, GMM-HMM (4 iteration) aligner and GMM-HMM (3 iteration) aligner, respectively

bias factor	SAD α 0.1	SAD α 0.2	SAD α 0.3	SAD α 0.4	SAD α 0.5
δ	T / G / G3	T / G / G3	T / G / G3	T / G / G3	T / G / G3
	0.5998	0.6092	0.6056	0.6101	0.6004
1/4	0.5919	0.6077	0.6028	0.6086	0.5896
	0.5863	0.5967	0.5904	0.5915	0.5656
	0.6061	0.6043	0.6098	0.6069	0.6013
1/3	0.5919	0.6050	0.6086	0.6058	0.5951
	0.5688	0.5966	0.5984	0.5906	0.5713
	0.6056	0.6048	0.6124	0.6108	0.5999
1/2	0.5955	0.6031	0.6092	0.6072	0.5910
	0.5891	0.5895	0.5993	0.5919	0.5674
	0.6016	0.5998	0.6145	0.6204	0.5988
1	0.5950	0.6007	0.6077	0.6140	0.5972
	0.5897	0.5901	0.5989	0.6004	0.5686
	0.6022	0.6120	0.6206	0.6207	0.5889
2	0.5920	0.6107	0.6102	0.6128	0.5886
	0.5864	0.5794	0.6006	0.6023	0.5649
	0.5898	0.6087	0.6004	0.6019	0.5870
4	0.5788	0.6024	0.6012	0.5993	0.5865
	0.5773	0.5976	0.5955	0.5848	0.5667

results. If we bias the loss function from the first epoch, the accuracy of the ASR front-end does not necessarily increase.

We also notice that, when α varies from 0.1 to 0.5, $DCF(\theta)$ keeps dropping while WER and ATMV do not. As aforementioned in Section 2.2, smaller α means that we take shorter segments without dilation, therefore there are much more segments when $\alpha = 0.1$ compared to $\alpha = 0.5$. So large α brings lower P_{FN}. But insert and delete errors, i.e., shown in Fig. 5, happen more and result in higher WER. When α is too small, ASR cannot work well because segments are too short and some speech information is missing, leading to increasing segmentation-aware WER. In this case, CTC aligner can not work normally, which indicates that is not robust enough.

We notice that when $\delta = 1/4, 1/3, 1/2, 1$, and 2, our proposed Model 1 system outperforms baseline with $\delta = 0$ obviously. When $\delta = 1/4, 1/3, 1/2$, and 1, our proposed Model 2 system outperforms baseline with $\delta = 0$ overall. Compared to baseline with CTC aligner, our system performs better by a large margin. In addition, the results of the CTC aligner are less stable than those of the GMM-HMM aligner, because there are many error sources, including but not limited to ASR accuracy, alignment of CTC, and SAD precision. Generally speaking, the performance of ASR is positively correlated with the performance of KWS, which is also consistent with our common sense. Compared to CTC aligner, the GMM-HMM aligner is more beneficial to our KWS system. On the condition of $\delta = 0$, our system with GMM-HMM aligner has lower WERs (relatively 7.9–19.2% decrease) and higher overall ATWVs (relatively 3.6–11.0% increase), which demonstrates the effectiveness of the proposed method. With different values of δ and α, our system can achieve relative WER reduction of 7.9% to 25.9% and improves TWV significantly. In particular, take the $\delta = 0$ as example, our ATWV has relatively 5.6% to 6.6% increase; as for the case of $\delta = 1/3$, we get relatively 3.6% to 11.0% increase.

During the OpenSAT20 evaluation, we also have trained some conventinal hybrid ASR and KWS models, but that is not the point in this paper. Based on the experiments, we found that the performance of our end-to-end system (using GMM-HMM AM) is comparable to the conventional hybrid systems (using Kaldi recipes), sometimes even slightly better. The hybrid ASR system WER varies from 11.6 to 14% (dev dataset segmented by SAD), the KWS system ATWV varies from 0.68 to 0.58. The final submitted result of the OpenSAT20 was the fusion of multiple single system results. We won the first prize in both SAD and KWS tracks, and the second prize in the ASR track.

For further comparison, we use the CNN-TDNNF network for AM training, named as "T" aligner. We find the Kaldi toolkit released the OpenSAT20 recipe recently, and we use it to train the CNN-TDNNF-based "T" aligner. We also use a GMM-HMM aligner trained by 3 iteration, named as "G3" aligner. We recognize five sets segmented dev dataset and compute corresponding segmentation-aware ATWVs of Model 1, summarized in Table 10. The auxiliary AM trained using the CNN-TDNNF network is slightly more precise than the HMM-GMM model, while the performance gap between "G" and "G3" aligners are wider. A precise AM can indeed align the text and time better. Considering that the training cost of DNN-based networks is much higher than that of GMM-HMM models, it is acceptable to use GMM-HMM AM for alignment when computation resources is insufficient.

5 Conclusions

The end-to-end KWS system suffers from two problems: data imbalance and inaccurate keyword timestamps. In order to alleviate the first issue, we modify the training loss function to make the ASR front-end pay more attention to our predefined keywords, leading to better performance on the KWS task. In addition, we address the second problem by utilizing a GMM-HMM based AM as the force aligner for the frame-level alignment to get the accurate timestamps of keywords. To evaluate the performance of our system, we have participated in the OpenSAT20 Evaluation. The experiment results show that the improved approach on the ASR and KWS task can achieve satisfactory performance. Up-weighting the loss assigned to the samples containing keywords can benefit the ASR front-end slightly and improve the KWS system a lot. According to the ASR and KWS results of two kinds of training strategies (Model 1 and Model 2), fine-tuning the ASR front-end on the last several epochs is preferred. Thanks to the precise frame alignment of the GMM-HMM AM, we can detect the keyword occurrences with more accurate timestamps than the baseline. We further study the alignment performance of auxiliary AMs with different accuracies. The more accurate the auxiliary AM is, the better performance for alignment. With enough computing resources, DNN-based auxiliary AM is the best choice, else it is acceptable to use a GMM-HMM AM for alignment.

Abbreviations

ASR: Automatic speech recognition; WER: Word error rate; CER: Character error rate; STD: Spoken term detection; KWS: Keyword searching; GMM: Gaussian mixture model; HMM: Hidden markov model; AM: Acoustic model; LVCSR: Large vocabulary continuous speech recognition; LM: Language model; CTC: Connectionist temporal classification; DNN: Deep neural network; SAD: Speech activity detection; CNN: Convolutional neural networks; RNN: Recurrent neural networks; ResNet: Residual network; GCN: Graph convolutional network; MFA: Montreal force alignment; MAUS: Munich automatic segmentation; WFST: Weighted finite-state transducer; SAT: Speaker adapted training; ATWV: Actual term-weighted value

Acknowledgements
This research was conducted as part of the submitted system of the THUEE team in the OpenSAT20 Evaluation.

Authors' contributions
Gui-Xin Shi carried out these approaches, implemented the experiments, and finished this article. Wei-Qiang Zhang conceived of the study and participated in its design and helped to draft the manuscript. Guan-Bo Wang implemented the SAD front-end and helped to draft the manuscript. Jing Zhao and Shu-Zhou Chai participated in implementation of the experiments and helped to draft the manuscript. The authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of 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 under Grant No. 2019GQG0001, and the Cross-Media Intelligent Technology Project of Beijing National Research Center for Information Science and Technology (BNRist) under Grant No. BNR2019TD01022.

Availability of data and materials
The datasets used or analyzed during this paper are available from OpenSAT20 Evaluation.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 13 December 2020 Accepted: 31 May 2021
Published online: 08 July 2021

References

1. D. R. H. Miller, M. Kleber, C. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, H. Gish, in *Proc. of INTERSPEECH*. Rapid and Accurate Spoken Term Detection (International speech communication association (ISCA), Antwerp, 2007), pp. 314–317
2. A. Graves, S. Fernández, F. J. Gomez, J. Schmidhuber, in *Proc. of ICML*. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks (ACM, Pittsburgh, 2006), pp. 369–376
3. G. Zweig, C. Yu, J. Droppo, A. Stolcke, in *Proc. of ICASSP*. Advances in All-Neural Speech Recognition (IEEE, New Orleans, 2017), pp. 4805–4809
4. A. H. Liu, T. Tu, H. Lee, L. Lee, in *Proc. of ICASSP*. Towards Unsupervised Speech Recognition and Synthesis with Quantized Speech Representation Learning (IEEE, Barcelona, 2020), pp. 7259–7263
5. W. Chan, N. Jaitly, Q. Le, O. Vinyals, in *Proc. of ICASSP*. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition (IEEE, Shanghai, 2016), pp. 4960–4964
6. J. Chorowski, D. Bahdanau, K. Cho, Y. Bengio, *End-to-end Continuous Speech Recognition using Attention-Based Recurrent NN: First Results*. (CoRR, 2014). https://arxiv.org/abs/1412.1602. Accessed 4 Dec 2014
7. D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, in *Proc. of INTERSPEECH*. End-to-End Attention-Based Large Vocabulary Speech Recognition (International speech communication association (ISCA), Shanghai, 2016), pp. 4945–4949
8. J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, in *Proc. of NIPS*. Attention-Based Models for Speech Recognition (MIT Press, Montreal, 2015), pp. 577–585
9. C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitlyp, B. Li, J. Chorowski, M. Bacchianil, in *Proc. of ICASSP*. State-of-the-Art Speech Recognition with Sequence-to-Sequence Models (IEEE, Calgary, 2018), pp. 4774–4778
10. H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang, T. Sainath, Deep Learning for Audio Signal Processing. IEEE J. Sel. Top. Sig. Process. **13**(2), 206–219 (2019)
11. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, in *Proc. of NIPs*. Attention is All you Need (MIT Press, Long Beach, 2017), pp. 5998–6008
12. B. Liu, S. Nie, S. Liang, W. Liu, M. Yu, L. Chen, S. Peng, C. Li, in *Proc. of INTERSPEECH*. Jointly Adversarial Enhancement Training for Robust End-to-End Speech Recognition (International speech communication association (ISCA), Graz, 2019), pp. 491–495
13. A. H. Liu, H. Lee, L. Lee, in *Proc. of ICASSP*. Adversarial Training of End-to-end Speech Recognition Using a Criticizing Language Model (IEEE, Brighton, 2019), pp. 6176–6180
14. T. Hori, S. Watanabe, J. R. Hershey. Joint CTC/Attention Decoding for End-to-End Speech Recognition (IEEE, Vancouver, 2017), pp. 518–529
15. H. Miao, G. Cheng, P. Zhang, T. Li, Y. Yan, in *Proc. of INTERSPEECH*. Online Hybrid CTC/Attention Architecture for End-to-End Speech Recognition (International speech communication association (ISCA), Graz, Austria, 2019), pp. 2623–2627
16. T. Hori, S. Watanabe, Y. Zhang, W. Chan, in *Proc. of INTERSPEECH*. Advances in Joint CTC-Attention Based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM (International speech communication association (ISCA), Stockholm, 2017), pp. 949–953
17. S. Kim, T. Hori, S. Watanabe, in *Proc. of ICASSP*. Joint CTC-Attention Based End-to-End Speech Recognition using Multi-Task Learning (IEEE, New Orleans, 2017), pp. 4835–4839
18. G. Chen, C. Parada, G. Heigold, in *Proc. of ICASSP*. Small-Footprint Keyword Spotting using Deep Neural Networks (IEEE, Florence, 2014), pp. 4087–4091
19. S. Ö. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, A. Coates, in *Proc. of INTERSPEECH*. Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting (International speech communication association (ISCA), Stockholm, 2017), pp. 1606–1610
20. A. H. Michaely, X. Zhang, G. Simko, C. Parada, P. S. Aleksic, in *Proc. of ASRU*. Keyword Spotting for Google Assistant using Contextual Speech Recognition (IEEE, Okinawa, 2017), pp. 272–278
21. F. Ballati, F. Corno, L. D. Russis, in *Proc. of WoRIE 2018*. "Hey Siri, Do You Understand Me?": Virtual Assistants and Dysarthria (IOS Press, Rome, 2018), pp. 557–566
22. X. Chen, S. Yin, D. Song, P. Ouyang, L. Liu, S. Wei, in *Proc. of ASRU*. Small-Footprint Keyword Spotting with Graph Convolutional Network (IEEE, Sentosa, 2019), pp. 539–546
23. K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, B. Kingsbury, End-to-End ASR-Free Keyword Search From Speech. IEEE J. Sel. Top. Sig. Process. **11**(8), 1351–1359 (2017)
24. Z. Zhao, W. Zhang, in *Proc. of INTERSPEECH*. End-to-End Keyword Search Based on Attention and Energy Scorer for Low Resource Languages (ISCA, Shanghai, 2020), pp. 2587–2591
25. Z. Zhao, W. Zhang, End-to-End Keyword Search System Based on Attention Mechanism and Energy Scorer for Low Resource Languages. Neural Netw. **139**, 326–334 (2021)
26. G. Wang, W. Zhang, in *Proc. of ISSPIT*. A Fusion Model for Robust Voice Activity Detection (IEEE, Ajman, 2019), pp. 1–5
27. G. Wang, W. Zhang, in *Proc. of APSIPA ASC*. An RNN and CRNN Based Approach to Robust Voice Activity Detection (IEEE, Lanzhou, 2019), pp. 1347–1350
28. T. Yoshioka, A. Ragni, M. J. F. Gales, in *Proc. of ICASSP*. Investigation of Unsupervised Adaptation of DNN Acoustic Models with Filter Bank Input (IEEE, Florence, 2014), pp. 6344–6348
29. P. Schwarz, Phoneme recognition based on long temporal context. PhD. Thesis. Brno University of Technology, Faculty of Information Technology (2009)
30. H. Hermansky, S. Sharma, in *Proc. of ICASSP*. Temporal Patterns (TRAPs) in ASR of Noisy Speech (IEEE, Phoenix, 1999), pp. 289–292
31. J. Ramírez, J. C. Segura, M. C. Benítez, T.orre. de la Á, A. J. Rubio, An Effective Subband OSF-Based VAD with Noise Reduction for Robust Speech Recognition. IEEE Trans. Speech Audio Process. **13**(6), 1119–1129 (2005)
32. N. Brümmer, E. de Villiers, The BOSARIS Toolkit: Theory, Algorithms and Code for Surviving the New DCF. arXiv:1304.2865 (2013)
33. NIST, NIST Open Speech Analytic Technologies 2019 (OpenSAT19). [EB/OL] (2019). https://sat.nist.gov
34. W. Hartmann, D. G. Karakos, R. Hsiao, L. Zhang, T. Alumäe, S. Tsakalidis, R. M. Schwartz, in *Proc. of ICASSP*. Analysis of Keyword Spotting Performance Across IARPA Babel Languages (IEEE, New Orleans, 2017), pp. 5765–5769
35. J. Wintrode, J. Wilkes, in *Proc. of ICASSP*. Fast Lattice-Free Keyword Filtering for Accelerated Spoken Term Detection (IEEE, Barcelona, 2020), pp. 7469–7473

36. T. Moriya, et al., in *Proc. of INTERSPEECH*. Self-Distillation for Improving CTC Transformer-Based ASR Systems (International speech communication association (ISCA), Shanghai, 2020), pp. 546–550
37. T. Lohrenz, Z. Li, T. Fingscheidt, *Multi-Encoder Learning and Stream Fusion for Transformer-Based End-to-End Automatic Speech Recognition*. (CoRR, 2021). https://arxiv.org/abs/2104.00120. Accessed 31 Mar 2021
38. S. Karita, N. Soplin, S. Watanabe, M. Delcroix, T. Nakatani, in *Proc. of INTERSPEECH*. Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration (International speech communication association (ISCA), Graz, 2019), pp. 1408–1412
39. M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, M. Sonderegger, in *Proc. of INTERSPEECH*. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi (International speech communication association (ISCA), Stockholm, 2017), pp. 498–502
40. P. C. Woodland, C. J. Leggetter, J. Odell, V. Valtchev, S. J. Young, in *Proc. of ICASSP*. The 1994 HTK Large Vocabulary Speech Recognition System (IEEE, Detroit, 1995), pp. 73–76
41. L. Kürzinger, D. Winkelbauer, L. Li, T. Watzel, G. Rigoll, in *Proc. of SPECOM*. CTC-Segmentation of Large Corpora for German End-to-End Speech Recognition, vol. 1233 (Lecture Notes in Computer Science, St. Petersburg, 2020), pp. 267–278
42. X. L. Aubert, An Overview of Decoding Techniques for Large Vocabulary Continuous Speech Recognition. Comput. Speech Lang. **16**(1), 89–114 (2002)
43. S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, T. Ochiai, in *Proc. of INTERSPEECH*. ESPnet: End-to-End Speech Processing Toolkit (International speech communication association (ISCA), Hyderabad, 2018), pp. 2207–2211
44. D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, Q. V. Le, in *Proc. of INTERSPEECH*. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition (ISCA, Shanghai, 2020), pp. 2613–2617
45. D. Povey, et al., in *Proc. of ASRU*. The Kaldi Speech Recognition Toolkit (IEEE, Waikoloa, 2011), pp. 1–5

Publisher's Note