STREAMING SMALL-FOOTPRINT KEYWORD SPOTTING USING SEQUENCE-TO-SEQUENCE MODELS

Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, Ian McGraw

Google Inc.,
Mountain View, CA, U.S.A.
{yanzhanghe,prabhavalkar,kanishkarao,mweili,bakhtin,imcgraw}@google.com

ABSTRACT

We develop streaming keyword spotting systems using a recurrent neural network transducer (RNN-T) model: an all-neural, end-to-end trained, sequence-to-sequence model which jointly learns acoustic and language model components. Our models are trained to predict either phonemes or graphemes as subword units, thus allowing us to detect arbitrary keyword phrases, without any out-of-vocabulary words. In order to adapt the models to the requirements of keyword spotting, we propose a novel technique which biases the RNN-T system towards a specific keyword of interest.

Our systems are compared against a strong sequence-trained, connectionist temporal classification (CTC) based "keyword-filler" baseline, which is augmented with a separate phoneme language model. Overall, our RNN-T system with the proposed biasing technique significantly improves performance over the baseline system.

Index Terms— Keyword spotting, sequence-to-sequence models, recurrent neural network transducer, attention, embedded speech recognition.

1. INTRODUCTION

Keyword spotting (KWS), sometimes also referred to as spoken term detection, is the task of detecting specific words, or multi-word phrases in speech utterances. Many previous works consider the problem of developing "offline" (i.e., non-streaming) KWS technologies. In this setting, the dominant paradigm consists of recognizing the entire speech corpus using a large vocabulary continuous speech recognizer (LVCSR) to build word or sub-word lattices, which can then be indexed to perform efficient search, e.g., [1, 2, 3].

In contrast to the methods described above, there is growing interest in building "online" (i.e., streaming) KWS systems which can be deployed on mobile devices which are significantly limited in terms of memory and computational capabilities. In such applications, *when deployed for inference*, the KWS system must continuously process incoming audio, and only trigger when a specific keyword is uttered. In order to simplify the problem further, most previous works assume that the model will only be required to detect a small number of possible keywords, thus allowing the development of keyword-specific models. Many previous works propose to train neural networks to identify word targets in individual keywords: for example, using feed-forward deep neural networks [4, 5, 6], convolutional networks [7] or recurrent neural networks [8, 9, 10]. Such systems assume the availability of a large number of examples of the keywords of interest in order to train models robustly. Prominent examples of such technologies include speech-enabled assistants such as "Okay/Hey Google" on Google Home [11], "Alexa" on the Amazon Echo, and "Hey Siri" on Apple devices. There has also been some prior work which has explored building low-footprint KWS systems which can detect arbitrary keywords in the incoming speech: for example, using structured support vector machines [12, 13], and techniques based on matching incoming audio to example templates of the keyword (Query-by-Example) [14, 15].

Recently, end-to-end trained, sequence-to-sequence models have become popular for speech recognition. Examples of such models include the recurrent neural network transducer (RNN-T) [16, 17], the recurrent neural aligner [18], connectionist temporal classification (CTC) [19] with grapheme [20, 21], syllable [22] or word targets [23], and attention-based models [24, 25, 26]. Such models combine the acoustic, and language model components of a traditional speech recognition system into a single, jointly trained model. In recent work, we have shown that RNN-T and attention-based models, trained on $\sim$12,500 hours of transcribed speech data to directly predict grapheme sequences without a separate language model, perform competitively on dictation test sets when compared against a state-of-the-art, discriminatively sequence-trained, context-dependent phone-based recognizer, augmented with a large language model [27]. We have also shown, that sequence-to-sequence models trained to predict phoneme-based targets, can be effective when used in a second pass rescoring framework [28].

There has been some recent work which has explored sequence-to-sequence models in the context of KWS. Zhuang et al. [29] use a long short-term memory (LSTM) [30] network with CTC to train a KWS system that generates phoneme lattices for efficient search. Rosenberg et al. [31] apply attention-based models to compute n-best lists of recognition results which are then indexed for efficient search; performance, however, was found to be worse than a traditional lattice-based KWS approach. Audhkhasi et al. [32] train an end-to-end system to predict whether a given keyword (represented as a grapheme string) is present in the speech utterance without explicitly decoding utterances into output phoneme or word strings.

In the present work, we explore the use of sequence-to-sequence models, specifically, RNN-T, to build a streaming KWS system which can be used to detect arbitrary keywords. Unlike a number of previous works which have only examined sequence-to-sequence models in the context of graphemes, we train RNN-T systems to predict graphemes as well as phonemes as sub-word units. Additionally, we propose a novel technique to *bias the search towards a specific keyword of interest* using an attention mechanism (described in more detail in Section 2.3). We find that RNN-T system trained to predict phonemes, when augmented with an additional "end-of-word" symbol (see Section 3.2) strongly outperforms a strong keyword-filler baseline derived from a sequence-trained CTC-based recognizer [33]. Overall, our best performing system achieves a false reject (FR) rate of 8.9% at 0.05 false alarms (FA) per hour,

To appear in *Proceedings of IEEE ASRU 2017*.

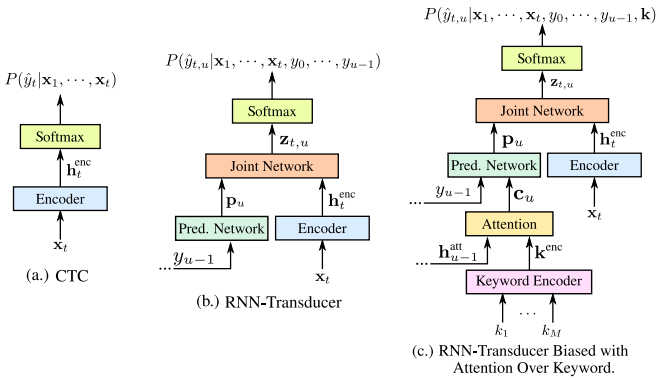


Fig. 1: A schematic representation of the models used in this work.

compared to the baseline which achieves 14.5% at the same FA threshold, which corresponds to a 39% reduction in the FR rate.

The organization of the rest of the paper is as follows. In Section 2 we describe various modeling strategies used in this paper. Section 3 describes our baseline approaches for keyword spotting. We present our experimental setup in Section 4, and discuss our results in Section 5, before concluding in Section 6.

2. MODELING STRATEGIES

In subsequent sections, we denote a sequence of parameterized acoustic features as, $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_T]$, where, $\mathbf{x}_t \in \mathbb{R}^d$; T denotes the number of acoustic frames in the utterance. We denote the corresponding sequence of output targets (e.g., graphemes or phonemes) corresponding to the utterance as $\mathbf{y} = [y_1, \cdots, y_L]$, where, $y_i \in \mathcal{Y}$. In the context of ASR, the input label sequence is typically much longer than the target label sequence, i.e., $T > L$.

2.1. Connectionist Temporal Classification

CTC [19] is a technique for modeling a conditional probability distribution over sequence data, $P(\mathbf{y}|\mathbf{x})$, when frame-level alignments of the target label sequence are unknown. CTC augments the set of output targets with an additional symbol, referred to as the *blank* symbol, denoted as $\langle b \rangle$. We denote by $\hat{\mathbf{y}} = [\hat{y}_1, \cdots, \hat{y}_T] \in \mathcal{B}(\mathbf{x}, \mathbf{y})$, the set of all label sequences of length $|\mathbf{x}| = T$, such that $\hat{\mathbf{y}}_t \in \{\mathcal{Y} \cup \langle b \rangle\}$, for $1 \leq t \leq T$, which are equivalent to $\mathbf{y}$ after first removing consecutive identical symbols, and then removing any blank symbols: e.g., $xx\langle b\rangle\langle b\rangle y\langle b\rangle \rightarrow xy$.

CTC models the output probability of the target sequence, $\mathbf{y}$, conditioned on the input, $\mathbf{x}$, by marginalizing over all possible frame-level alignments, where each output label is assumed to be independent of the other labels, conditioned on $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \mathcal{B}(\mathbf{x},\mathbf{y})} P(\hat{\mathbf{y}}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \mathcal{B}(\mathbf{x},\mathbf{y})} \prod_{t=1}^{T} P(\hat{y}_t|\mathbf{x}_1, \cdots, \mathbf{x}_t) \tag{1}$$

The conditional probability, $P(\hat{y}_t|\mathbf{x}_1, \cdots, \mathbf{x}_t)$, can be computed using a recurrent neural network (which we refer to as the *encoder* network), as illustrated in Figure 1(a.). As shown in the figure, the encoder maps each input frame, $\mathbf{x}_t$, into a higher-level representation, $\mathbf{h}_t^{\text{enc}}$, followed by a softmax layer which converts $\mathbf{h}_t^{\text{enc}}$ into a probability distrubution $P(\hat{y}_t|\mathbf{x}_1, \cdots, \mathbf{x}_t)$ over the output labels in $\{\mathcal{Y} \cup \langle b \rangle\}$. The model can be trained using stochastic gradient descent to optimize likelihood over the training set, given paired input and target sequences $(\mathbf{x}, \mathbf{y})$. The gradients required for this process can be computed using the forward-backward algorithm [19].

2.2. RNN Transducer

Although CTC has been used successfully in many previous works in the context of ASR (e.g., [34, 35, 23]), it makes a strong conditional independence assumption since it assumes that outputs at each step are independent of the history of previous predictions. The RNN-T model improves the CTC approach by augmenting it with an additional *prediction network* [16, 17], which is explicitly conditioned on the history of previous outputs, as illustrated in Figure 1(b.). The RNN-T model may be viewed as a type of sequence-to-sequence model architecture [24, 25], where the encoder (referred to as a *transcription network* in [16]) corresponds to the RNN acoustic model in a traditional recognizer, and the *prediction network* (together with the *joint network*) corresponds to the *decoder*. The decoder network may be viewed as an RNN language model which attempts to predict the current label given the history of labels. We note that unlike most attention-based models that have been explored in the past (e.g., [24, 25]), output targets can be extracted from the RNN-T in a streaming fashion, since the model does not have to examine the entire encoded utterance in order to compute an output target label.

The prediction network is provided with the previous non-blank input label, $y_u \in \mathcal{Y}$, as input, and produces a single output vector, denoted as $\mathbf{p}_u$. The prediction network is fed a special symbol at the start of decoding, $y_0 = \langle \text{sos} \rangle$, which denotes the start of the sentence.

The joint network consists of a set of feed-forward layers which compute logits $\mathbf{z}_{t,u}$ for every input frame t and label u, using additional parameters A, B, b, D, d, as follows:

$$\mathbf{h}_{t,u}^{\text{joint}} = \tanh(A\mathbf{h}_t^{\text{enc}} + B\mathbf{p}_u + b) \tag{2}$$

$$\mathbf{z}_{t,u} = D\mathbf{h}_{t,u}^{\text{joint}} + d \tag{3}$$

These logits are passed to a final softmax layer which computes probabilities over targets in $\{\mathcal{Y} \cup \langle b \rangle\}$.[1]

The model can be trained to optimize likelihood over the training set, by marginalizing over all possible alignments (i.e., $\mathcal{B}(\mathbf{x}, \mathbf{y})$) similar to CTC, using stochastic gradient descent where the required gradients are computed using the dynamic programming algorithm described in [16, 17].

2.3. Biasing the RNN-Transducer with the keyword of interest using the attention mechanism

Previous works that have examined the use of sequence-to-sequence models for KWS (e.g., [31]) have typically only done so indirectly; the models is trained for ASR, and used to generate n-best lists which can be indexed for efficient search. A notable exception, is work by Audhkhasi et al. [32] where the model is trained directly for the KWS task which is similar to the query-by-example approach that has been investigated previously [14].

With the goal of improving KWS performance, we extend the RNN-T system described in Section 2.2 with an attention-based keyword biasing mechanism in the prediction network to make the model aware of the keyword of interest during the search process. This model can be thought of as a variant of the RNN-T model augmented with attention, proposed in our previous work [27], wherein we replace the prediction network with an attention-based decoder that computes attention over the targets in the keyword phrase. The

[1] These equations correspond to Eq. 15–18 in [17].

intuition is that during inference, when the suffix of the current predicted label sequence is close to the prefix of the keyword, the attention vector is activated in the corresponding position within the keyword. This, in turn, generates a context vector to bias the network prediction towards the remaining part of the keyword. Critically, since the keyword phrase only consists of a small number of targets, the use of attention over the keyword does not introduce any latency or significant computational overhead during inference. This model is depicted in Figure 1(c.).

Specifically, at each step, the prediction network recieves, in addition to the previous non-blank label y_{u-1}, a *context vector*, $\mathbf{c}_u$ which is computed using dot-product attention [24] over the keyword targets (phoneme targets, in our experiments). We denote the sequence of phoneme targets in the keyword phrase to be detected, as $\mathbf{k} = [k_1, \cdots, k_M, k_{M+1}]$, where M is the number of targets in the keyword phrase, and k_{M+1} is a special target that corresponds to "not applicable", denoted $\langle$n/a$\rangle$.[2] The keyword encoder takes as input the phoneme sequence, and outputs a matrix $\mathbf{k}^{\text{enc}} = [k_1^{\text{enc}}, \cdots, k_M^{\text{enc}}, k_{M+1}^{\text{enc}}]$, where k_i^{enc} is a one-hot embedding vector of k_i, and k_{M+1}^{enc} is a zero vector. If we denote the state of the prediction network after predicting $u-1$ labels as $\mathbf{h}_{u-1}^{\text{att}}$, the context vector, $\mathbf{c}_u$ is computed as follows:

$$\beta_{j,u} = \left\langle \phi(k_j^{\text{enc}}), \psi(\mathbf{h}_{u-1}^{\text{att}}) \right\rangle \qquad \text{for each } 1 \leq j \leq M+1 \quad (4)$$

$$\alpha_{j,u} = \frac{e^{\beta_{j,u}}}{\sum_{j'=1}^{M+1} e^{\beta_{j',u}}} \quad (5)$$

$$\mathbf{c}_u = \sum_{j=1}^{M+1} \alpha_{j,u} k_j^{\text{enc}} \quad (6)$$

where, $\phi(\cdot)$ and $\psi(\cdot)$ represent linear embeddings, and $\langle \cdot, \cdot \rangle$ represents the dot product between two vectors. Thus, the prediction network produces an output $\mathbf{p}_u$ conditioned on both the previously predicted labels, as well as the keyword of interest.

Unlike the RNN-T model, which can be trained given pairs of input and output sequences $(\mathbf{x}, \mathbf{y})$, in order to train the RNN-T model with keyword biasing, we need to also associate a keyword phrase, $\mathbf{k}$, with the training instance. We create examples where the keyword, $\mathbf{k}$, is present in $\mathbf{x}$, as well as examples where the keyword is absent in $\mathbf{x}$ as follows: with probability p^{kw} we uniformly sample one of the words in $\mathbf{x}$ as the keyword, $\mathbf{k}$, and with probability $1 - p^{\text{kw}}$ we uniformly sample a word which is not in $\mathbf{x}$ as the keyword, $\mathbf{k}$. If we select one of the words in $\mathbf{x}$ as the target, we modify the target labels $\mathbf{y}$ by inserting a special symbol $\langle$eokw$\rangle$ after the occurence of the keyword. For example, when training with phoneme targets, for the utterance `the cat sat`, (which corresponds to the phoneme sequence[3] [D V $\langle$eow$\rangle$ k { t $\langle$eow$\rangle$ s { t $\langle$eow$\rangle$]), if we sampled $\mathbf{k}$ =`cat` as the keyword, then we would modify the target labels as, $\mathbf{y}$ = [D V $\langle$eow$\rangle$ k { t $\langle$eow$\rangle$ $\langle$eokw$\rangle$ s { t $\langle$eow$\rangle$]. Note that the $\langle$eow$\rangle$ token marks the end of each word token (see Section 3.2). The intuition behind adding the $\langle$eokw$\rangle$ at the end of the keyword phrase in the transcript, is that it might serve as a marker that the model should attend to the targets in the keyword phrase. As a final note, the training and inference algorithms for this model are similar to the standard RNN-T model.

[2]We also experimented with excluding this symbol, and only using the targets in the keyword, and found that the overall performance was similar to a model with this target. In this work, we only present results with the $\langle$n/a$\rangle$ keyword target.

[3]We use X-SAMPA to denote phonemes throughout the paper.

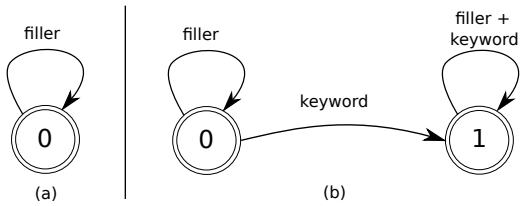


Fig. 2: Two decoder graphs representing the building blocks of our baseline CTC-based keyword spotters.

3. BASELINE SYSTEMS

We present two baseline approaches for the task of streaming KWS. First, we adapt an embedded LVCSR system, designed for efficient real-time recognition on a wide variety of smartphones, developed in our previous work [33]. Second, we explore "keyword-filler" models [36] using the acoustic model component of the LVCSR system. These approaches are described in the following sections.

3.1. LVCSR with CTC

Our first approach directly uses an embedded LVCSR system developed in our previous work [33] to recognize input utterances; this is followed by a simple confidence estimation scheme in order to detect a particular keyword of interest. In particular, we recognize the input utterance, $\mathbf{x}$, and create an n-best list of hypotheses, denoted as $\mathcal{W}$. Note that the output vocabulary of the system is limited to 64K words, which results in a significant number of out-of-vocabulary words during the search process. In previous works, e.g., [37], the KWS confidence metric is defined as a likelihood ratio of the keyword model to a background model. Similar to the approaches, we define a simple confidence metric based on the n-best list, as follows. Given an utterance $\mathbf{x}$, we identify the highest probability hypothesis in $\mathcal{W}$ containing $\mathbf{k}$: $P(\mathbf{w}^+|\mathbf{x})$, and the highest probability hypothesis in $\mathcal{W}$ which does not contain $\mathbf{k}$: $(P(\mathbf{w}^-|\mathbf{x}))$, setting these to 0 if no such hypothesis exists in the n-best list. We can then compute a confidence metric $C(\mathbf{x}) \in [0, 1]$ as:

$$C(\mathbf{x}) = \frac{P(\mathbf{w}^+|\mathbf{x})}{P(\mathbf{w}^+|\mathbf{x}) + P(\mathbf{w}^-|\mathbf{x})} \quad (7)$$

Thus, in the case where all n-best entries contain the keyword, the confidence score is set to one; when none of the entries contain the keyword, the score is set to zero. This same confidence metric is used for all systems, including the RNN-T systems presented in this paper.

3.2. Keyword-Filler Models with CTC

An alternative approach to KWS is through the use of "keyword-filler" models [36], which corresponds to constucting a decoder graph with two basic paths: the first is a path through the keyword(s), and the second is a path through a filler (background) that models all non-keyword speech. We use this approach to create our next set of keyword spotters.

Instead of defining a single decoder graph with keyword and filler paths, we find it advantageous to use two decoders on separate graphs as depicted in Figure 2. This effectively corresponds to using two beams during decoding: one for the filler model (Figure 2 (a)), and one for the keyword paths, (Figure 2 (b)). The scores of the most likely paths from each these graph can be used to estimate $P(\mathbf{w}^-|\mathbf{x})$ and $P(\mathbf{w}^+|\mathbf{x})$, respectively, which can be used to generate a confidence score using Equation 7.

The simplest example of a filler model is a phone loop. However, we remove all paths from the filler model which contain the keyword's phones, so that any path containing the keyword must pass through the keyword model.

In previous work it has been shown that constraining filler models yields accuracy improvements [38, 39, 40]. We therefore explore two variants along these lines. In the first, we replace the simple phone loops with unweighted word loops (using the 64k word vocabulary from [33]), thus adding in word-level constraints. In the second, we apply an n-gram phone LM, trained on automatically generated phonetic transcriptions of the same utterances that are used to train the word-level LM in [33]; the number of parameters in the phone LM is trained to match the number of parameters of the word LM in [33]. In this case, we compose the LM with both the filler and keyword graphs.

In preliminary experiments, we found that a source of false-positives during KWS with phoneme based models was when a part of word's phonetic transcription matched that of the keyword. For example, the keyword `Erica (E r\ @ k @)` is incorrectly detected in utterances containing the word, `America (@ m E r\ @ k @)`; `Marilyn (m E r\ @ l @ n)` is incorrectly detected in utterances containing the word, `Maryland (m E r\ @ l @ n d)`. We therefore expanded the phoneme LM by inserting a special symbol ⟨eow⟩ at the end of each word's pronunciation when creating training data, e.g., `the cat sat` → `D V` ⟨eow⟩ `k { t` ⟨eow⟩ `s { t` ⟨eow⟩. The ⟨eow⟩ token is the analog of the space symbol which delimits words in their graphemic representation; from the long context along with the ⟨eow⟩ symbol, the phone LM is expected to implicitly model word-level dependencies and learn the correct segmentation of a phone sequence into words. During search, we only consider keywords in between two end-of-word markers, or between a start-of-sentence marker and an end-of-word marker, in the hypotheses. For instance, `Erica` would not be false triggered in the phrase: `In America (I n` ⟨eow⟩ `@ m E r @ k @` ⟨eow⟩), but will correctly trigger when the utterance contains `Call Erica (k O l` ⟨eow⟩ `E r\ @ k @` ⟨eow⟩).

The idea of using an end-of-word symbol has also been explored in [29], however the authors added it to the transcript for training the CTC acoustic model instead. We believe it would be more explicit and effective to use the symbol for LM training, in which the label dependencies are modeled directly, whereas in CTC the output targets are conditionally independent to each other. As is shown in the results below, we also use the end-of-word symbol for training RNN-T models and find it useful, where AM and LM are jointly trained.

4. EXPERIMENTAL DETAILS

4.1. Data and Evaluation Metric

Our models are trained on a set of ∼22M hand-transcribed anonymized utterances extracted from Google voice-search traffic, which corresponds to ∼18,000 hours of training data. In order to improve system robustness to noise and reverberation, multi-condition training (MTR) data are generated: training utterances are artificially distorted using a room simulator, by adding in noise samples extracted from YouTube videos and environmental recordings of daily events. To further improve robustness to variation in signal loudness, we perform multi-loudness training by scaling the loudness of each training utterance to a randomly selected level.

We construct separate development and test sets to measure KWS performance. As keyword phrases we consider personal names which contain three or more syllables (e.g., `Olivia` or `Erica`). The development set consists of 328 keywords, each of which is contained in ∼75 positive utterances, collected from multiple speakers, of the form "keyword, query", (e.g., `Olivia, how tall is the Eiffel tower?`). A set of ∼37K negative utterances (∼50 hours in total) are shared across keywords, which are collected as queries without a keyword, to form the full development set. Each keyword is evaluated separately on a set consisting of its own positive utterances and the shared negative utterances. A test set is created similarly, with 228 keywords each contained in ∼500 positive utterances, and a set of ∼20k negative utterances (∼60 hours in total) shared across keywords, which consist of hand-transcribed anonymized utterances extracted from Google traffic from the domains of open-ended dictation and voice-search queries.

We evaluate performance in terms of the receiver operating characteristic (ROC) curve [41], which is constructed by sweeping a threshold over all possible confidence values and plotting false reject (FR) rates against false alarm (FA) rates. Our goal is to achieve low FR rates while maintaining extremely low FA rates (e.g. no more than 0.1 false alarms per hour of audio).

Following [42], we employ a score normalization approach to map system confidence score at the utterance level for a keyword to the probability of false alarm (pFA) for that keyword, which allows us to use a single consistent score for all keywords and set the decision threshold reliably. A confidence-score-to-pFA mapping is estimated from the development set, and applied to both the development and the test sets. All ROC curve results in this work are plotted after the score normalization.

4.2. Model Details

The input acoustic signal is represented with 80-dimensional log-mel filterbank energies, computed with a 25ms window, and a 10ms frame-shift. Following previous work [34], we stack three consecutive frames and present only every third stacked frame as input to the encoder. The same acoustic frontend is used for all experiments described in this work.

The CTC acoustic model (AM) consists of 5 layers of 500 LSTM cells, that predict context-independent phonemes as output targets. The system is heavily compressed, both by quantization [43], and by the application of low-rank projection layers with 200 units between consecutive LSTM layers [44]. The AM consists of 4.6 million parameters in total. The model is first trained to optimize the CTC objective function [19] until convergence. Once CTC-training is complete, the model is discriminatively sequence-trained to optimize expected word errors by minimizing word-level, edit-based, minimum Bayes risk (EMBR) proposed recently by Shannon [45].

The encoder networks used in all RNN transducer models are identical in size and configuration to the encoder used in the CTC model (without the softmax output layer). During the training of an RNN transducer, the weights from the encoders are initialized from a pre-trained CTC model, since this was found to significantly speed up convergence, following which the weights are trained jointly with the rest of the network. For the RNN-T model that is trained to directly output grapheme targets, the CTC model used for initialization is also trained to predict graphemes. The grapheme inventory includes the 26 lower-case letters (`a-z`), the numerals (0-9), a label representing 'space' (⟨space⟩), and punctuation symbols (e.g., the apostrophe symbol (`'`), hyphen (`-`), etc.).

The prediction network used in the RNN transducer models,

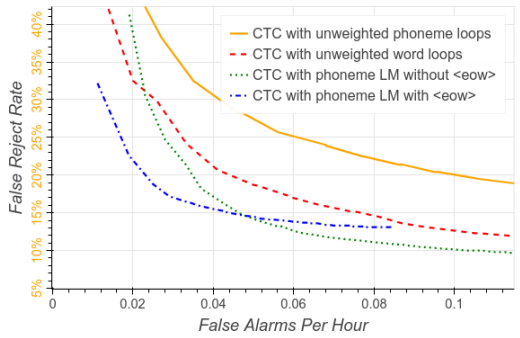


Fig. 3: Comparison among multiple CTC baseline systems on the test set.

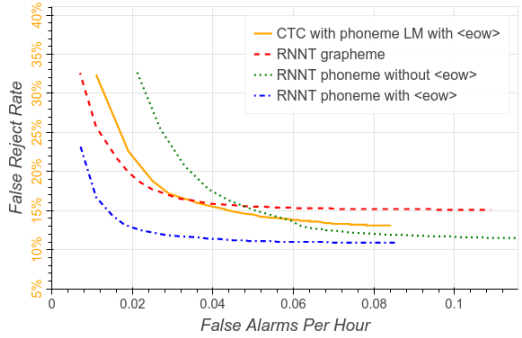


Fig. 4: A comparison of the various RNN-T systems against the best performance CTC baseline on the test set.

both with and without attention, consists of a single layer of 500 LSTM cells with coupled input and forget gate (CIFG) [46], and the joint network consists of a single feed-forward layer of 500 units with a $\tanh$ activation function, as described in Section 2.2. The decoder network (including prediction network and the joint network) has 1.5 million parameters in total.

The RNN transducer models are decoded using a beam-search algorithm [16], where at most 50 highest scoring candidates are retained at every step during decoding. In general, the output posterior distribution of sequence-to-sequence models like RNN-T is peaky (i.e., low entropy); such over-confidence is typically suboptimal for keyword spotting, since diversity in hypotheses is critical to reduce the number of false rejects. We find that smoothing the output posteriors with a temperature τ, i.e. mapping each posterior to its τ-th root and renormalizing them, can help improve KWS performance significantly. The optimal temperature value is determined by tuning on the development set; we set $\tau = 2.0$ for all RNN-T models without attention, and $\tau = 2.2$ for the ones with attention. However smoothing the output posteriors of the CTC acoustic model does not help, possibly because it does not combine well with the LM.

The language model (LM) used in the keyword-filler model with CTC is trained to predict phoneme targets on the same $\sim$22M utterances used for training RNN-T models. The LM is pruned to $\sim$1.5-million 6-grams using entropy pruning, similar to the number of parameters in the decoder of our RNN-T models. We choose $n = 6$ which is optimized from the development set.

The LM for our embedded LVCSR system is a standard word-level 5-gram, which is trained on a larger corpus with $\sim$100M automatically-transcribed anonymized utterances extracted from Google voice-search traffic. This LM is also pruned to $\sim$1.5-million n-grams using entropy pruning. The vocabulary is limited to 64K words, allowing us to shrink the data structures used to maintain the LM [33]. Note that the fixed vocabulary results in out-of-vocabulary keywords on the development and test sets. Utterances are decoded with a heavily pruned version of the LM in the first-pass, while rescoring with the full LM on-the-fly, thus allowing us to reduce the size of the decoder graph used in the first-pass.

All models are trained using asynchronous stochastic gradient descent [47], and are implemented in TensorFlow [48].

5. RESULTS

5.1. Baselines

The performance of our CTC-trained "keyword-filler" baseline models is shown in Figure 3. As can be seen, we find that a phoneme language model is important for a keyword-filler system even with a strong CTC model for extremely low FA rates ($\leqslant$ 0.05 FAs per hour). The effect of the language model can be seen by comparing different levels of constraints added in the keyword-filler graphs.

CTC with unweighted phoneme loops allows for arbitrary phoneme paths in the graph to be treated as *equally likely*, thus entirely relying on the CTC model to recognize the keyword from the background, which performs the worst. Adding word constraints in the graph, albeit without weights, helps to improve performance since it eliminates many confusable paths that correspond to invalid words. Note that in this case, we can add the keyword phrase into the vocabulary for the search since the word loop filler models are unweighted.

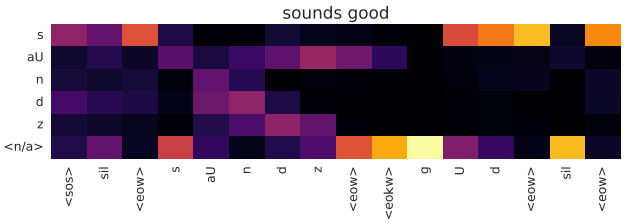
The addition of a phoneme language model without the $\langle$eow$\rangle$ token helps to recognize phoneme sequences in context, but does not account for word constraints. As described in Section 3.2, this model has an increased number of false triggers (e.g., the keyword `Erica` is detected incorrectly in utterances containing the word `America`). The addition of $\langle$eow$\rangle$ to the phoneme language models, however, significantly improves performance over the other baseline systems.

For reference, a KWS system constructed from the embedded LVCSR system, as described in Section 3.1, achieves an FR rate of 29.8% at 0.05 FAs per hour on the test set for only in-vocabulary keywords (196 out of total 228 keywords), while the best CTC system above achieves 13.4% on the same set.

5.2. RNN-T Models with Graphemes and Phonemes Targets

Compared to CTC with a phoneme n-gram LM, an RNN-T model with phoneme targets jointly trains an acoustic model component and a language model component in a single all-neural system. As can be seen from Figure 4, an RNN-T phoneme model (with $\langle$eow$\rangle$) outperforms the best CTC baseline. If the $\langle$eow$\rangle$ token is not used, however, the RNN-T phoneme system has significantly higher false alarms as explained in Section 3.2.

The RNN-T system trained to predict grapheme targets performs worse than the one trained with phoneme targets. We conduct an analysis to determine the cause of this performance degradation and found that it is partly due to variant orthographic representations of some of the keyword phrases: e.g., the keyword `kathryn` is encountered very rarely in the training data, and as a result the RNN-T model typically recognizes these examples as `catherine`, which is more common in the training data. We therefore considered a variant system where we replace each keyword with the most frequent orthographic representation (as determined by its unigram probability) during the search. This technique significantly improves false



(a) Attention matrix of a positive utterance for the keyword "sounds", with the transcript "sounds good".

(b) Attention matrix of a negative utterance for the keyword "afternoon", with the transcript "you're welcome you know".

Fig. 5: Attention matrices for two representative utterances computed by the RNN-T phoneme system with keyword biasing. The Y-axis corresponds to targets $k_1, \cdots, k_{M+1}$ in the keyword $\mathbf{k}$. The X-axis corresponds to the expected sequence of phoeneme targets given the utterance transcript. The entry at row j and column u corresponds to $\alpha_{j,u}$ in Equation 5, with values in each column summing up to 1. Brighter colors correspond to values closer to 1, while darker colors correspond to values closer to 0.

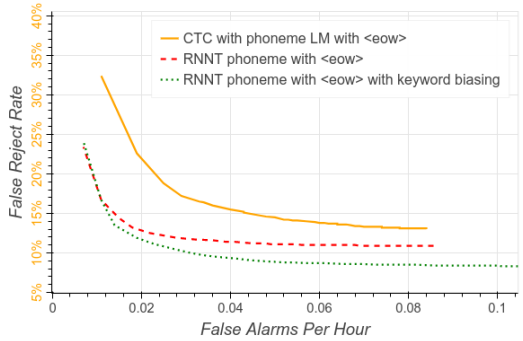


Fig. 6: A comparison of the RNN-T phoneme model with keyword biasing against the best CTC baseline and the RNN-T phoneme system without biasing on the test set. All systems use the ⟨eow⟩ token.

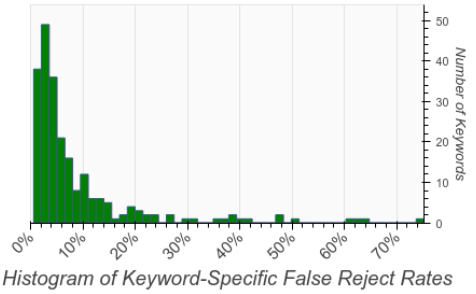


Fig. 7: Histogram of keyword-specific false reject rates for the RNN-T phoneme system with keyword biasing at 0.05 FAs per hour, plotted for the keywords on the test set.

reject rates for the RNN-T grapheme system from 15.5% to 14.0% at 0.05 FAs per hour; however this system was still worse than the RNN-T phoneme model, which achieves an FR rate of 11.1% at 0.05 FAs per hour.

5.3. RNN-T with Keyword Biasing

We train an RNN-T phoneme system with ⟨eow⟩ and ⟨eokw⟩ labels, by setting $p^{\text{KW}} = 0.5$, determined by tuning on the development set. As is shown in Figure 6, adding attention-based keyword biasing to an RNN-T phoneme system improves the overall performance significantly. The final results are reported on the test set, where CTC, RNN-T phoneme and RNN-T phoneme with biasing achieve 14.5%, 11.1% and 8.9% false reject rates respectively at 0.05 FAs per hour.

We also plot a histogram of the FR rates across keywords at a threshold corresponding to 0.05 FAs per hour for the RNN-T phoneme system with keyword biasing in Figure 7. As can be seen in the figure, most of the keywords have low FR rates in the 0–15% range, with only a few outliers.

Finally, in Figure 5 we plot representative examples of the attention weights $\alpha_{j,u}$ computed by the attention model during inference on a positive (Figure 5 (a)) and a negative (Figure 5 (b)) utterance extracted from the training data. These plots were generated by feeding as input the expected target label sequence (i.e., the labels are not determined by a beam-search decoding).

As can be seen in the figure, when decoding the positive utterance, the attention weights are concentrated on the first target. When the model begins to predict the phonemes corresponding to the keyword (`sounds (s aU n d z)`), the attention weights are focussed on consecutive keyword targets, as revealed by the prominent diagonal pattern (although admittedly, the model also appears to attend to other keyword targets during this process). We also note the prominent attention weight assigned to the ⟨n/a⟩ label after the keyword has been detected.

In the case of the negative utterances, however, the attention does not evolve diagonally across the labels, but is instead spread across the second keyword target (i.e., the initial part of the hotword), and the ⟨n/a⟩ label.

6. CONCLUSIONS

In this work, we developed streaming keyword spotting systems using a recurrent neural network transducer, a sequence-to-sequence model that jointly trains acoustic and language model components. We proposed a novel techinque which biases the RNN-T system towards a specific keyword of interest based on an attention mechanism over the keyword. In experimental evaluations, we find that our RNN-T system trained with phoneme targets performs significantly better on keyword spotting than a strong CTC-based keyword-filler baseline which is augmented with a phoneme n-gram LM. We also find that the proposed biasing techique provides further gains over the vanilla RNN-T model.

7. REFERENCES

[1] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. of Special Interest Group on Information Retrieval (SIGIR)*, 2007, pp. 51–57.

[2] D. R. H. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Proc. of Interspeech*, 2007.

[3] D. Vergyri, I. Shafran, A. Stolcke, R. R. Gadde, M. Akbacak, B. Roark, and W. Wang, "The SRI/OGI 2006 spoken term detection system," in *Proc. of Interspeech*, 2007.

[4] G. Chen, C. Parada, and G. Heigold, "Small footprint keyword spotting using deep neural networks," in *Proc. of ICASSP*, 2014.

[5] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. N. Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks," in *Proc. of ICASSP*, 2015.

[6] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting," in *Proc. of Interspeech*, 2016.

[7] T. N. Sainath and C. Parada, "Convolutional neural networks for small footprint keyword spotting," in *Proc. of Interspeech*, 2015.

[8] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. of IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 474–480.

[9] S. Ö. Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *ArXiv e-prints*, March 2017.

[10] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proc. of International Conference on Artificial Neural Networks (ICANN)*, 2007, pp. 220–229.

[11] B. Li, T. N. Sainath, J. Caroselli, A. Narayanan, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K. Sim, R. J. Weiss, K. W. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon, "Acoustic modeling for google home," in *Proc. of Interspeech*, 2017.

[12] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.

[13] R. Prabhavalkar, K. Livescu, E. Fosler-Lussier, and J. Keshet, "Discriminative articulatory models for spoken term detection in low-resource conversational settings," in *Proc. of ICASSP*, 2013, pp. 8287–8291.

[14] T. J. Hazen, W. Shen, and C. M. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *Proc. of ASRU*, 2009, pp. 421–426.

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

[16] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. of ICASSP*, 2012.

[17] A. Graves, A.-R. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. of International Conference on Machine Learning: Representation Learning Workshop*, 2013.

[18] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence-to-sequence mapping," in *Proc. of Interspeech*, 2017.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labeling unsegmented sequence data with recurrent neural networks," in *Proc. of the International Conference on Machine Learning (ICML)*, 2006.

[20] K. Hwang, M. Lee, and W. Sung, "Online keyword spotting with a character-level recurrent neural network," *ArXiv e-prints*, December 2015.

[21] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," *ArXiv e-prints*, November 2016.

[22] Y. Bai, J. Yi, H. Ni, Z. Wen, B. Liu, Y. Li, and J. Tao, "End-to-end keywords spotting based on connectionist temporal classification for mandarin," in *Proc. of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2016, pp. 1–5.

[23] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition," *ArXiv e-prints*, October 2016.

[24] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. of ICASSP*, 2016, pp. 4960–4964.

[25] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. of ICASSP*, 2016, pp. 4945–4949.

[26] L. Lu, X. Zhang, and S. Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *Proc. of ICASSP*, 2016.

[27] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Proc. of Interspeech*, 2017.

[28] R. Prabhavalkar, T. N. Sainath, B. Li, K. Rao, and N. Jaitly, "An analysis of "attention" in sequence-to-sequence models," in *Proc. of Interspeech*, 2017.

[29] Yimeng Zhuang, Xuankai Chang, Yanmin Qian, and Kai Yu, "Unrestricted vocabulary keyword spotting using lstm-ctc," in *Proc. of Interspeech*, 2016, pp. 938–942.

[30] S. Hochreiter and J. Schmidhuber, "long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[31] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *Proc. of ICASSP*, 2017, pp. 5280–5284.

[32] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," in *Proc. of ICASSP*, 2017, pp. 4840–4844.

[33] I. McGraw, R. Prabhavalkar, R. Alvarez, M. Gonzalez Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized speech recognition on mobile devices," in *Proc. of ICASSP*, 2016.

[34] H. Sak, A. W. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. of Interspeech*, 2015.

[35] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *Proc. of ASRU*, 2015, pp. 167–174.

[36] I. Szöke, P. Schwarz, P. Matějka, L. Burget, M. Karafiát, and J. Černocký, *Phoneme Based Acoustics Keyword Spotting in Informal Continuous Speech*, pp. 302–309, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[37] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *Proc. of ICASSP*, 1990, pp. 129–132.

[38] G. Wang and K. C. Sim, "Context dependent acoustic keyword spotting using deep neural network," in *Proc. of APSIPA*, 2013, pp. 1–10.

[39] I.-F. Chen, C. Ni, B. P. Lim, N. F. Chen, and C.-H. Lee, "A keyword-aware grammar framework for lvcsr-based spoken keyword search," in *Proc. of ICASSP*, 2015, pp. 5196–5200.

[40] Mitchel Weintraub, "Keyword-spotting using sri's decipher large-vocabulary speech-recognition system," in *Proc. of ICASSP*, 1993, pp. 463–466.

[41] C. Cortes and M. Mohri, "Confidence intervals for the area under the roc curve," in *Proc. of NIPS*, 2004.

[42] B. Zhang, R. Schwartz, S. Tsakalidis, L. Nguyen, and S. Matsoukas, "White listing and score normalization for keyword spotting of noisy speech," in *Proc. of Interspeech*, 2012.

[43] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," in *Proc. of Interspeech*, 2016.

[44] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *Proc. of ICASSP*, 2016, pp. 5970–5974.

[45] M. Shannon, "Optimizing expected word error rate via sampling for speech recognition," in *Proc. of Interspeech*, 2017.

[46] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, 2016.

[47] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large Scale Distributed Deep Networks," in *Proc. of NIPS*, 2012, pp. 1223–1231.

[48] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M.Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.