VOICE TRIGGER DETECTION FROM LVCSR HYPOTHESIS LATTICES USING BIDIRECTIONAL LATTICE RECURRENT NEURAL NETWORKS

Woojay Jeon, Leo Liu, and Henry Mason

Apple
One Apple Park Way, Cupertino, California
{woojay,lliu9,hmason}@apple.com

ABSTRACT

We propose a method to reduce false voice triggers of a speech-enabled personal assistant by post-processing the hypothesis lattice of a server-side large-vocabulary continuous speech recognizer (LVCSR) via a neural network. We first discuss how an estimate of the posterior probability of the trigger phrase can be obtained from the hypothesis lattice using known techniques to perform detection, then investigate a statistical model that processes the lattice in a more explicitly data-driven, discriminative manner. We propose using a Bidirectional Lattice Recurrent Neural Network (LatticeRNN) for the task, and show that it can significantly improve detection accuracy over using the 1-best result or the posterior.

Index Terms— voice trigger, detection, lattice, RNN

1. INTRODUCTION

Speech-enabled personal assistants are often conveniently activated by use of a trigger phrase. In the case of the Apple personal assistant Siri, English users can say "Hey Siri" to activate the assistant and make a request in a single step, e.g. "Hey Siri, how is the weather today in Cupertino?"

Typically, an on-device detector [1] decides whether the trigger phrase was spoken, and if so allows the audio (including the trigger phrase) to flow to a server-based large vocabulary continuous speech recognizer (LVCSR). Because the on-device detector is resource-constrained, its accuracy is limited and leads to occasional "false triggers" where the user did not speak the trigger phrase but the device wakes up anyway and gives an unexpected response.

To reduce false alarms, one could conceive of a secondary trigger phrase detector running on the server, utilizing a much larger statistical model than the one on the device to more accurately analyze the audio and override the device's trigger decision when it sees that no trigger phrase is present. This method would optimally improve the accuracy because we are using a dedicated acoustic model that is specifically trained for the detection task. However, since this must be done for every utterance, a more resource-efficient approach is to use the output of the server-side LVCSR, which is run anyway for every utterance, to perform the secondary detection.

The obvious method is to check whether the top recognition result produced by the LVCSR begins with the trigger phrase or not. However, the LVCSR is often biased toward recognizing the trigger phrase at the beginning of the audio, and therefore "hallucinates" the phrase in the 1-best result even though it does not exist.

LVCSR output has been leveraged to keyword spotting (or keyword search, which is closely related) in many past studies. An early method used the sum of the likelihoods of hypotheses containing the keyword in an n-best list [2]. However, an n-best list is a lossy representation of the word hypothesis lattice, which is a richer representation of the output of an ASR [3]. Hence, subsequent works have acted directly on the hypothesis lattice by computing word posteriors [4] or normalized confidence scores [5] or contextual features via neural networks [6].

We will begin by examining how a secondary voice trigger detector can be built using known processing techniques on the LVCSR's hypothesis lattice to compute a posterior probability for the trigger phrase [4]. This method, however, is strictly limited by the reliability of the acoustic and language models of the LVCSR, which may not be accurate enough for falsely-triggered audio that often contains diverse and unpredictable sounds that may or may not be speech. To actively overcome some of the LVCSR's errors, we consider the use of a statistical model that can interpret the hypothesis lattice in a discriminative, data-driven manner. We propose the use of a bidirectional version of "LatticeRNN" [7] for this purpose, and show that a significant gain in accuracy can be obtained compared to using the simple posterior probability.

2. VOICE TRIGGER DETECTION BASED ON LATTICE POSTERIORS

Consider the probability that a speech utterance with acoustic features X starts with a designated trigger phrase. The trigger phrase is a fixed sequence of n words, $V = [v_1, v_2, \cdots, v_n]$. The probability of the first n words of the utterance,

$w_1, \cdots, w_n$ being the trigger phrase is

$$P\left(w_1 = v_1, w_2 = v_2, \cdots, w_n = v_n \mid X\right) \quad (1)$$

If we can compute the above probability, we could simply apply a threshold to obtain a trigger phrase detector.

The probability in Eq. (1) can be written as

$$\sum_{r_1, r_2 \cdots} P(w_1 = v_1, , \cdots, w_n = v_n, \\ w_{n+1} = r_1, w_{n+2} = r_2, \cdots \mid X) \quad (2)$$

where $r_1, r_2, \cdots$ are the words following the trigger phrase.

With some abuse of notation, we write this as

$$\sum_{R} P\left(V, R \mid X\right) = \frac{\sum_{R} p\left(V, R, X\right)}{\sum_{W} p\left(W, X\right)} \quad (3)$$

where V, R, and W represent the trigger phrase terms, the words following the trigger phrase, and all the words in the utterance, respectively.

If we assumed that the hypothesis lattice provided by an LVCSR spans the entire space of all possible word sequences for R and W (which is obviously a significant approximation, since the lattice would be heavily pruned and show only a small set of possible hypotheses), Eq. (3) can be solved in a straightforward way using the well-known lattice forward-backward algorithm [4, 8].

Stated more formally for our specific case, we have

$$p\left(V, R, X\right) = \sum_{\mathbf{q}} p\left(V, R, \mathbf{q}, X\right) = \sum_{\mathbf{q} \in Q_{VR}} p\left(X, \mathbf{q}\right) \quad (4)$$

where $\mathbf{q}$ is a path through the hypothesis lattice, and Q_{VR} is the set of all paths that contain the word sequence $V + R$.

Let $\mathbf{q}_V$ be the front part of $\mathbf{q}$ that contains the trigger phrase V and $\mathbf{q}_R$ be the remainder of $\mathbf{q}$ that contains the rest of the words R. Also let X_V and X_R denote the speech frames consumed by $\mathbf{q}_V$ and $\mathbf{q}_R$, respectively. We now have

$$p\left(V, R, X\right) = \sum_{\mathbf{q}_V \in Q_V} \sum_{\mathbf{q}_R \in Q_R(\mathbf{q}_V)} p\left(X_V, X_R, \mathbf{q}_V, \mathbf{q}_R\right) \quad (5)$$

where Q_V is the set of all initial partial paths that contain V, and $Q_R(\mathbf{q}_V)$ is the set of all residual paths that follow $\mathbf{q}_V$.

This becomes

$$p\left(V, R, X\right) = \sum_{\mathbf{q}_V \in Q_V} p\left(X_V, \mathbf{q}_V\right) \beta\left(X_R, \mathbf{q}_v\right) \quad (6)$$

where

$$\beta\left(X_R, \mathbf{q}_v\right) = \sum_{\mathbf{q}_R \in Q_R(\mathbf{q}_V)} p\left(X_R, \mathbf{q}_R\right) \quad (7)$$

The joint distribution $p(X_V, \mathbf{q}_V)$ in Eq. (6) is simply obtained by multiplying the joint probabilities along the path

$\mathbf{q}_V$. If $\mathbf{q}_V$ consists of n arcs $\lambda_1, \cdots, \lambda_n$, each i'th arc consuming the acoustic features X_i and storing an acoustic model score $p\left(X_i \mid \lambda_i\right)$ and a contextual transition score (which includes the language model score and pronunciation score) $P\left(\lambda_i \mid \lambda_1 \cdots \lambda_{i-1}\right)$, we have

$$p\left(X_V, \mathbf{q}_V\right) = \prod_{i=1}^{n} p\left(X_i \mid \lambda_i\right) P\left(\lambda_i \mid \lambda_1 \cdots \lambda_{i-1}\right) \quad (8)$$

$\beta(X_R, \mathbf{q}_V)$ in Eq. (6) is the "backward" score of the node at the end of path $\mathbf{q}_V$ that we obtain by the lattice forward-backward algorithm.

3. BIDIRECTIONAL LATTICE-RNN FOR VOICE TRIGGER DETECTION

In the previous section, we discussed how to compute a voice trigger posterior probability from the hypothesis lattice to perform voice trigger detection. A fundamental limitation to such an approach is that it is directly exposed to errors in the LVCSR's acoustic and language model scores. If the LVCSR is overly biased toward giving high scores to the voice trigger phrase, the posterior in Eq. (3) will be consistently high and the detection accuracy will suffer. The only tunable parameter in the system is the detection threshold applied to the posterior, and one parameter (applied across all utterances) is not sufficient for overcoming modeling errors in the LVCSR.

This motivates us to build a more general statistical model with many more parameters that can use training examples to learn how to process the hypothesis lattice in a data-driven manner. In effect, the model learns the "mistakes" in the LVCSR's scores and actively tries to correct them. In our proposed method, we employ Lattice Recurrent Neural Networks [7] that can read entire hypothesis lattices without requiring us to heuristically convert them into lossy forms such as n-best lists or word confusion networks.

3.1. The Bidirectional LatticeRNN

"LatticeRNN" [7] was originally introduced for the task of classifying user intents in natural language processing. For a topologically-sorted hypothesis lattice, the feature vector of each arc is input to the neural network along with the state vector of the arc's source node, and the output of the neural network becomes the arc's state. For a given arc e that has input feature vector $\mathbf{x}(e)$ whose source node $p(e)$ possesses state vector $\mathbf{h}_f\left(p\left(e\right)\right)$, the neural network characterized by input transformation U_f, state transformation V_f, bias $\mathbf{b}_f$, and activation function $g(\cdot)$ outputs the arc's state $\mathbf{h}_f(e)$:

$$\mathbf{h}_f\left(e\right) = g\left\{U_f^T \mathbf{x}\left(e\right) + V_f^T \mathbf{h}_f\left(p\left(e\right)\right) + \mathbf{b}_f\right\} \quad (9)$$

For a given node s, the state vector $\mathbf{h}_f\left(s\right)$ is obtained via a pooling function applied to the states of all incoming arcs:

$$\mathbf{h}_f\left(s\right) = f_{pool}\left(\{\mathbf{h}_f\left(e\right) : n\left(e\right) = s\}\right) \quad (10)$$

Since information propagates only in a forward direction in this neural network, the state of an arc is determined only by the arcs that precede it, and is unaffected by any arc that succeeds it. If we imagined the state of the arcs for the words "hey" and "Siri" as a measure of how relevant they are for the detection task (analogous to their posterior probabilities), it would be desirable for the states to also depend on the succeeding words. Words asking about the weather, for instance, should make the trigger word arcs more relevant than a random string of words resembling a TV commercial. Based on this intuition, we also perform the same sort of propagation in the reverse direction, in a similar manner as is done for conventional RNNs [9].

In parallel to the forward propagation in Eq. (9), we have another neural network parameterized by U_b, V_b, and $\mathbf{b}_b$, which takes arc e's feature vector $\mathbf{x}(e)$ and its destination node's *backward* state vector $\mathbf{h}_b(n(e))$ and outputs the arc's backward state vector $\mathbf{h}_b(e)$:

$$\mathbf{h}_b(e) = g\left\{U_b^T\mathbf{x}(e) + V_b^T\mathbf{h}_b(n(e)) + \mathbf{b}_b\right\} \qquad (11)$$

The backward state vector for a node s is obtained by applying the pooling function to the outgoing arcs:

$$\mathbf{h}_b(s) = f_{pool}(\{\mathbf{h}_b(e) : p(e) = s\}) \qquad (12)$$

The forward state vector $\mathbf{h}_f(s_{\text{term}})$ of the lattice's terminal node and the backward state vector $\mathbf{h}_b(s_{\text{init}})$ of the lattice's initial node are concatenated to form a single state vector $\mathbf{h}_{\text{lat}}$ that represents the entire lattice.

$$\mathbf{h}_{\text{lat}} = \left[\mathbf{h}_f^T(s_{\text{term}}) \quad \mathbf{h}_b^T(s_{\text{init}})\right]^T \qquad (13)$$

An additional feed forward network is then applied to $\mathbf{h}_{\text{lat}}$ to output a single scalar value that represents how likely the input lattice starts with the voice trigger phrase.

4. EXPERIMENT

Data Type	VT	No VT	Total
Training	12,271	6,731	19,002
Development	3,347	1,836	5,183
Evaluation	6,693	3,672	10,365

Table 1. Counts of utterances with (VT) and without (no-VT) the voice trigger phrase collected for this experiment. Note that "no-VT" utterances are relatively rare in production because most of them are immediately discarded by the device and it is only those that are falsely accepted by the device-side detector that can become part of our data. Hence, the "no-VT" utterances must be collected over a much longer period of time than "VT" utterances.

A labeled set of utterances was used for the experiment, where some began with the "Hey Siri" trigger phrase and the

Method	P_M (%)	P_{FA} (%)	EER (%)
Baseline	0.84	78.92	-
Posterior	0.84	79.08	35.68
Lattice RNN	0.84	22.55	5.51
Bidir Lat RNN	0.84	17.05	4.59

Table 2. Results on development data, showing the Probability of Miss (P_M), Probability of False Alarm (P_{FA}), and Equal Error Rate (EER). The "Baseline" method is to check the 1-best result of the LVCSR, and is not tunable. The "Posterior" method is to threshold the trigger phrase posterior probability in Section 2. "Lattice RNN" and "Bidir Lat RNN" are the neural network models in Section 3. For each of the bottom three methods, P_M and P_{FA} is from the operating point where P_M is closest to the baseline P_M.

Method	P_M (%)	P_{FA} (%)
Baseline	0.84	78.46
Posterior	0.84	78.87
Lattice RNN	1.20	20.81
Bidir Lat RNN	1.15	17.57

Table 3. Results on evaluation data. For the bottom three methods, detection thresholds corresponding to the operating points in Table 2 (with fixed P_M on the development data) were used to obtain the values in this table.

rest did not. Table 1 shows the number of positive and negative examples used for training, development, and evaluation.

The "baseline" method is to simply look at the top recognition result from the LVCSR and check whether it begins with "Hey Siri" or not. As shown in Tables 2 and 3, the Probability of Miss (failing to recognize the trigger phrase when it is present) is usually less than 1%, but the Probability of False Alarm ("hallucinating" the trigger phrase when it is not present) is around 79% on this data set. Note that an actual Siri user would experience far less false alarms because most utterances without the trigger phrase get immediately discarded by the device-side detector, and it is only those occasional few that slip past the detector that become part of our negative data.

The "posterior" method is that described in Section 2, where the voice trigger posterior probability is directly computed from the hypothesis lattice. As is evident in the ROC curve in Figure 1, above a certain threshold (where there is a clear sharp angle in the curve) the voice trigger posterior tends to be evenly distributed between true triggers and false triggers and is hence a poor discriminant. Below the turning point, however, most of the inputs are false triggers, so the detector performs much better when the false alarm probability is around 55% or higher. For the development data, the P_{FA} is 79.08% when P_M is the same as the baseline (0.84%). The threshold obtained from this operating point was applied to the evaluation data to obtain the values in Table 3.

For the lattice RNNs, the arc feature vector $\mathbf{x}(e)$ consists of 19 features: the log acoustic score, the log language model score, the number of speech frames consumed by the arc, a binary feature indicating whether the word is "hey", a binary feature indicating whether the word is "Siri", and 14 features representing the phone sequence of the arc's word. The phone sequence, which is variable length, is converted to a 51-dimensional binary bag-of-phones vector and reduced to 14 dimensions via an autoencoder. The autoencoder is trained using a lexicon of pronunciations for 700K words. The unidirectional lattice RNN has 24 dimensions for the state vector, which is fed to a feedforward network with 20 hidden nodes, resulting in a total 1,577 parameters. The bidirectional lattice RNN has 15 dimensions in each state vector, and is used with a feedforward network with 15 hidden nodes, resulting in a total 1,531 parameters. All inputs are mean- and variance-normalized, with the scale and bias computed from the training data. The pooling function in Eq. (10) and (12) is the arithmetic mean.

A huge accuracy gain is observed when using the lattice neural network compared to the baseline or posterior-based method, and more when using the bidirectional instead of unidirectional lattice RNN.

In terms of runtime computational complexity, the proposed method adds minimal latency to the existing LVCSR because 1) the lattice is usually compact; over the training data, the average number of arcs per lattice is 42.7 whereas the average number of acoustic feature frames is 406, and 2) the lattice RNN is small, with only around 1,500 parameters.

To retain the order of the phone sequence in each arc, we also tried replacing the bag-of-phones features with the encoding from a sequence-to-sequence autoencoder, but did not observe accuracy improvement with the given data.

5. CONCLUSION AND FUTURE WORK

We have proposed a novel method of voice trigger phrase detection based on the hypothesis lattice of an LVCSR using a Bidirectional Lattice Recurrent Neural Network, and showed that it can significantly reduce the occurrence of false triggers in a digital personal assistant.

Given that the LVCSR is being used for both trigger detection and speech recognition in this case, future work will investigate an objective function that jointly maximizes both recognition accuracy and detection accuracy, which would better fit the true goal of the system.

6. ACKNOWLEDGEMENTS
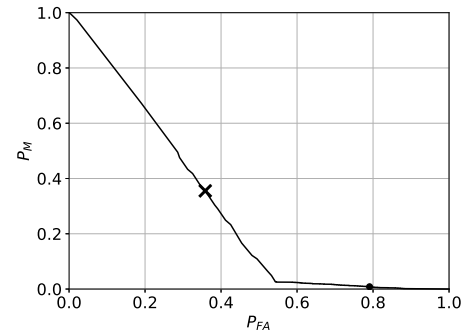
Fig. 1. ROC curve for posterior-based voice trigger detection on development data showing Probability of Miss (P_M) vs. Probability of False Alarm (P_{FA}). "×" marks the equal error rate operating point and "•" marks the point where P_M is closest to the baseline's.
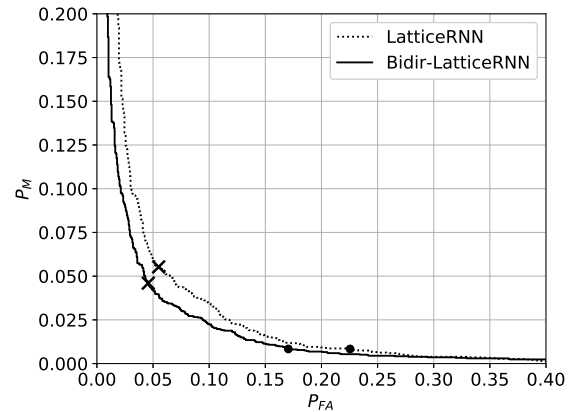
Fig. 2. ROC curve on development data using Lattice RNN and Bidirectional Lattice RNN. "×" indicates the equal error rate operating points and "•" indicates the operating point where P_M is closest to the baseline's.
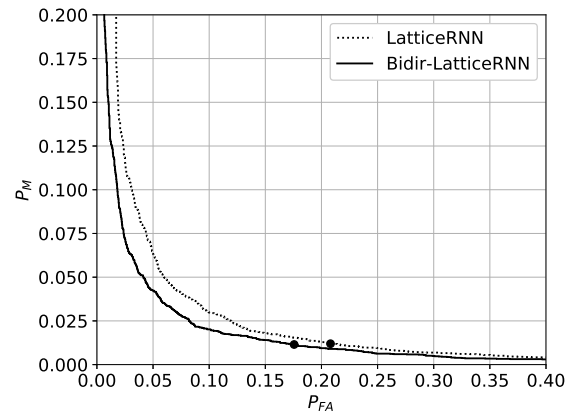
Fig. 3. ROC curve on evaluation data for voice trigger detection using Lattice RNN and Bidirectional Lattice RNN. The "•" for each curve indicates the operating point where the threshold from the "•" in Figure 2 is applied to the evaluation data.

7. REFERENCES

[1] S. Sigtia, R. Haynes, H. Richards, E. Marchi, and J. Bridle, "Efficient voice trigger detection for low resource hardware," in *Proc. Interspeech 2018*, 2018, pp. 2092–2096.

[2] M. Weintraub, "LVCSR log-likelihood ratio scoring for keyword spotting," in *ICASSP*, 1995.

[3] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *HLT-NAACL*, 2004.

[4] I. Szöke, P. Schwarz, P. Matějka, and M. Karafiát, "Comparison of keyword spotting approaches for informal continuous speech," in *Eurospeech*, 2005.

[5] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel, and V. Le, "Score normalization and system combination for improved keyword spotting," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 210–215.

[6] Z. Chen and J. Wu, "A rescoring approach for keyword search using lattice context information," in *Interspeech 2017*, pp. 3592–3596.

[7] F. Ladhak, A. Gandhe, M. Dreyer, L. Mathias, A. Rastrow, and B. Hoffmeister, "LatticeRNN: Recurrent neural networks over lattices," in *Interspeech 2016*, pp. 695–699.

[8] F. Wessel, R. Schluter, K. Macherey, and H. Ney, "Confidence measures for large vocabulary continuous speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 288–298, March 2001.

[9] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997.