DYNAMIC SPEECH ENDPOINT DETECTION WITH REGRESSION TARGETS

Dawei Liang**, Hang Su*†*, Tarun Singh*†*, Jay Mahadeokar*†*, Shanil Puri*†*, Jiedan Zhu*†*,*
Edison Thomaz**, Mike Seltzer*†

* University of Texas at Austin, † Meta AI.

ABSTRACT

Interactive voice assistants have been widely used as input interfaces in various scenarios, e.g. on smart home devices, wearables and on AR devices. Detecting the end of a speech query, i.e. speech end-pointing, is an important task for voice assistants to interact with users. Traditionally, speech end-pointing is based on pure classification methods along with arbitrary binary targets. In this paper, we propose a novel regression-based speech end-pointing model, which enables an end-pointer to adjust its detection behavior based on the context of user queries. Specifically, we present a pause modeling method and show its effectiveness for dynamic end-pointing. Based on our experiments with vendor-collected smartphone and wearables speech queries, our strategy shows a better trade-off between end-pointing latency and accuracy, compared to the traditional classification-based method. We further discuss the benefits of this model and generalization of the framework in the paper.

Index Terms**—** end-pointing, end-of-query, interactive voice assistant.

1. INTRODUCTION

With the rapid development of speech technologies in recent years, interactive voice assistants have been widely adopted as a mainstream for intelligent user interface [1, 2, 3, 4]. By taking users' speech queries, these systems are able to perform a variety of tasks from basic question answering, music playing, calling and messaging, to device control. As an initial step, a voice assistant needs to determine the time point when a user finishes the query so that it knows when to close the microphone and continue downstream processing, e.g., automated speech recognition (ASR) and taking action. This process is often referred to as speech *endpoint detection* or *end-pointing*. In practice, the challenge for speech end-pointing lies in the conflicting goals of response latency and accuracy (avoiding early cuts of user speech). On the one hand, a rapid close of the microphone and a timely response to user queries brings a better user experience. On the other hand, however, this inevitably increases the risk of early cuts of user queries. The performance of an end-pointing system is thus evaluated on how this conflict is resolved in practical cases.

Canonically, voice activity detection (VAD) has been used for speech end-pointing [5, 6]. A VAD model can be used to distinguish speech from non-speech segments, including silence and background noises [7, 8, 9]. The end of the query can then be determined if a fixed duration of silence is observed by the VAD system. However, this approach is not reliable enough [10, 11]. The fact that a VAD system is not typically trained to distinguish within-query pauses and end-of-query pauses prevents the system from capturing enough acoustic cues related to end-of-query detection, such as speaking rhythm or filler sounds [11].

Recent research on speech end-pointing mostly focuses on classification methods, where a dedicated end-pointer is developed to classify audio frames as end-of-query frames and others [11, 12]. The success of the Long Short-Term Memory (LSTM) [13] architecture contributes to this method. Some other efforts leverage additional text decoding features [14] or user personalized i-vectors [15] to better fit the end-pointer for specific acoustic environments. In an end-to-end ASR system, the end-pointer may also be jointly optimized with the recognition model [16]. Despite the promising results, all the above work focuses on binary detection of speech endpoints with hard labels (i.e. 0 and 1). In real scenarios, however, an end-pointer should adjust its end-pointing aggressiveness based on semantics, prosody or other speaking patterns in the query. The traditional binary targets for classification can be less flexible in this respect.

In this paper, we study a novel speech end-pointing strategy based on regression. Specifically, an end-pointer is optimized to fit soft-coded targets during training, and the training targets are adjusted with the expected pause given the semantic context of queries. By testing on 14.4M smartphone speech queries and 467K queries from wearables, we show that our proposed method effectively reduces the response delay of end-pointing while maintaining a comparable accuracy performance as the conventional classification-based method.

2. REGRESSION-BASED ENDPOINT MODELING

2.1. Principle

In traditional classification, a speech end-pointing model is trained against targets of hard-coded values (0 and 1). During

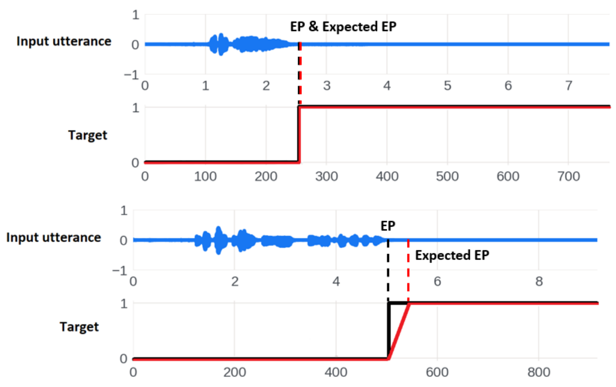

Fig. 1. Sample targets for end-pointer training based on hard-coded binary values (target in black) and soft-coded float values (target in red). EP: endpoint.

inference, a decision threshold is chosen to adjust the model's end-pointing confidence. In a regression setting, the inference stage remains the same, but the training targets are changed to be continuous float values from 0 to 1, representing the confidence of a frame being part of the endpoint. Specifically, we apply a transition curve to the area between the actual and expected (i.e. position where we hope to perform end-pointing) endpoint position (Figure 1). The motivation of our regression-based modeling includes: 1) the model should have an increased confidence in end-pointing as more silence is observed; 2) the aggressiveness of the end-pointing behavior should be adjustable, which corresponds to the slope of the transition region. As an extreme case, a regression model with a zero transition region would have the same targets as a classification model. The top plot of Figure 1 is an example where the training targets of a regression model and a classification model become the same. At the bottom, the regression target enforces the model to be less aggressive, with a smoother slope in the transition region.

Figure 2 presents the overall pipeline of end-pointing with regression. To enable a dynamic output behavior of the end-pointer, the transition slope of the training targets can be quantified based on unique patterns (e.g., expected pause) of individual input queries. In this paper, we propose a method to compute and leverage the pause statistics of queries for this purpose (Section 3). During inference, the regression model shares the same setup as the classification model, i.e., taking speech features as inputs and applying a decision threshold to the model outputs for endpoint decision.

2.2. Loss Function

In our study, we used the mean-square-error (MSE) as the loss function for end-pointer training. The loss function is defined as below:

$$l = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n} \tag{1}$$

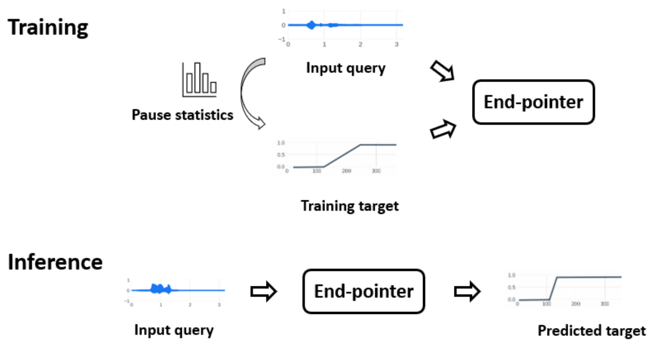

Fig. 2. The overall pipeline of our proposed method.

where i is the ith frame of a query in the dataset, n is the total number of frames in the dataset, $\hat{y}_i$ and y_i are the output of the end-pointer and the true target, respectively. In our experiments, we found that regression with the binary targets yields the same performance as the traditional classification setup. Furthermore, the MSE loss tends to be more stable for model convergence than the L1 loss.

3. SPEAKER PAUSE MODELING

To enable dynamic adjustments of end-pointing behaviors, we explore the expected pause following a speech segment. Our intuition is that ideal end-pointing aggressiveness should be controlled by the expected pause following a query, and the expected pause is related to semantics. For example, the chance of a follow-up speech after a query *"what's the weather"* should be bigger than that after *"what's the weather in Seattle next week"*. By calculating the expected pause duration following a speech segment, we can adjust the slope of the transition region to be smoother for the first query so that the end-pointer can learn to wait longer for the first query than for the second before responding. Hence, we hope to estimate this expected pause duration for each training query and use it to adjust the targets for these queries.

To this end, we first model the pause duration for a given text as a Gaussian variable:

$$T_T|C \sim \mathcal{N}(\mu, \sigma^2) \tag{2}$$

Here, T_T is the pause duration for the text of a query, and C is the context of the text. μ and σ are parameters for the Gaussian variable. The parameters of this model can be estimated by aggregating pause statistics of queries with a shared context (prefix). For example, the query "*what is the weather*" can be considered as a prefix for queries "*what is the weather*", "*what is the weather in Seattle*", and "*what is the weather today*", and all these queries contribute to the estimation of statistics for the query "*what is the weather*". Note that in this example, the query "*what is the weather*" has zero end-of-query pause duration, because it is a complete query itself. In this work, the prefix definition requires a strict match of transcriptions. Hence, queries "*what is the weather*" and "*how is the weather*" were considered distinct, though they are semantically similar and may ideally be grouped together. Also, we used the 95^{th} percentile statistics in our experiments for the expected pause estimation rather than the mean μ to reduce the risk of early-cutting.

We then model the expected pause for a given speech query as follows:

$$T_S = T_T \cdot R \tag{3}$$

Here, T_S is the expected pause duration for a speech query, and R is the speaking rate. The intuition here is that different queries may be spoken at different speaking rates, and the slower a speaker speaks, the longer the expected pause can be. The speaking rate factor can be estimated using the ratio between the duration of a speech query and the average duration of all those queries with the same prefix (duration of the prefix part only).

4. EXPERIMENTAL SETUP

4.1. Data

We used two datasets to evaluate our method - both were collected by third-party data vendors. The first one was collected using smartphones whereas the second one was collected using wearable devices (smart glasses). We refer to the two datasets as smartphone data and wearables data in this paper, respectively. The smartphone dataset contains clean user speech that is spoken in a fluent manner, with little background noise. In total, 14.4M queries were collected, categorized into 55 domain contexts based on the text transcription. The most common contexts are *music* (1.4M), *weather* (1.3M), and *device handling* (1.2M). The average word count per query is 6. The wearables dataset was collected when users interacted with a voice assistant on smart glasses, with real life noises in the background. A considerable amount of speech pauses and dis-fluency were identified in this data. It contains 447K queries, with an average word count of 11 per query. Both datasets contain speech and ground truth transcriptions. Ground truth speech endpoints were calculated by aligning speech to transcription and measuring the end time of the last words. From the alignments, we also collected pause duration (if any) following each word in the queries.

For both datasets, we randomly split the queries at a ratio of 95:5 for model development and testing. A training and a validation set was further split out at random during training for learning rate adjustment. The training set and testing set share the same set of speakers.

4.2. Model Configuration

Our end-pointer is an LSTM neural network inspired by prior work [11, 14]. Specifically, the model consists of three uni-directional LSTM layers with a hidden unit size of 128 per layer. For the baseline classification model, the output layer is a fully-connected layer of two neurons, transformed by log-softmax activation. For the regression model, the output layer includes only a single neuron with sigmoid activation. During training, the baseline model used the regular cross-entropy loss, while the regression model used MSE loss. The learning rate was 2×10^{-4} / 2×10^{-3} for each model, and reduced by a factor of 0.5 if no improvement was observed on the validation set. The Adam optimizer [17] was used with a Beta of (0.9, 0.999). The mini batch size was 128. Besides, a sliding window of 10 ms was applied to the output of the baseline model at the inference stage for score smoothing. Both models were trained for 15 epochs, given the learning rates dropped below a threshold. All development was conducted using the PyTorch toolkit [18].

We used 40-dim filter-bank features as inputs to the networks. We extracted the features based on a window of 25 ms and a stride of 10 ms. The sampling rate of the audio was 16 kHz.

5. RESULTS

5.1. Overall Results

In total, we obtained 106K and 18K unique prefixes respectively for the smartphone and the wearables training sets. Following the prior work [11, 14], we applied the early-cut rate as the model accuracy metric which reports the proportion of early-cut queries out of the entire test population. We did not employ common application metrics such as the word error rate (WER) here because we viewed end-pointing and its cascaded applications (e.g., ASR) as separate tasks, and the WER is not a direct indicator of the end-pointing accuracy and its resulting user experience. For example, an early cut at the head and tail of an utterance can both lead to interaction failure, but their resulting WER in ASR can differ greatly. For the latency measure, we used the 50^{th}, 75^{th}, 90^{th}, and 99^{th} percentile statistics of latency values out of the predicted results, noted as P50, P75, P90, and P99, respectively. For both metrics, a smaller value is more desirable. Tables 1 and 2 summarize the overall model performance of our study. The decision thresholds in the tables are reported in pairs such that the early-cut rates between the baseline and the regression models remain comparable at a given threshold. The model performance is then examined by the output latency. Due to space limit, we reported three featured pairs (baseline thresholds 0.5 - 0.7) in each table, and the observations are similar for the remaining. We can see that the prediction latency measured by P50, P75, and P90 tends to be improved by the regression model on both datasets. This demonstrates a better

Threshold	**Early-cut rate (%)**	**P50 (ms)**	**P75 (ms)**	**P90 (ms)**	**P99 (ms)**
0.50 / 0.63	3.39 / 3.38	160 / **120**	180 / **150**	230 / **210**	**480** / 530
0.60 / 0.74	2.45 / 2.38	170 / **120**	200 / **160**	250 / **240**	**530** / 600
0.70 / 0.82	1.74 / 1.67	180 / **130**	210 / **170**	280 / **260**	**590** / 660

Table 1. End-pointing performance with classification / regression-based models for the smartphone dataset.

Threshold	**Early-cut rate (%)**	**P50 (ms)**	**P75 (ms)**	**P90 (ms)**	**P99 (ms)**
0.56 / 0.50	14.83 / 14.81	420 / **350**	590 / **450**	860 / **810**	**1990** / 2500
0.60 / 0.59	12.82 / 12.75	470 / **430**	670 / **510**	970 / **870**	**2050** / 2500
0.70 / 0.67	10.53 / 10.37	580 / **500**	780 / **650**	**1100** / 1130	**2120** / 2500

Table 2. End-pointing performance with classification / regression-based models for the wearables dataset.

balance of end-pointing behavior achieved by the regression model. For P99, the classification model is consistently better. This is in fact reasonable, because P99 shows the very tail latency of prediction, and the regression model was enforced to have a very smooth performance for queries of potentially a high expectation of succeeding pause. For the typical user experience of latency, regression-based end-pointing is still more desirable, as demonstrated by P50 and P75.

5.2. Discussions

To study the effects of the design factors, we also examined the model performance based on a simpler setting. In this setting, the slope of the training targets was adjusted based on either the local pause or the total word length of a query. For some queries, we observed that the output of the regression model changed as expected, where its output remained as sharp as the classification baseline for a short query and became smoother for a longer one (Figure 3). However, we did not observe superior performance globally (e.g., the P50 / P75 at an early-cut rate of 2.0% was 180 / 300 ms for the local pause method on the smartphone data). This indicates that the regression method can be generalized to different slope adjustment strategies as well. However, the local pause or duration information alone may not be sufficient for optimal global performance, and including the speaking rate helps to further improve the model's performance empirically.

In an extra study, we also noticed that our proposed method helps to maintain the model stability for end-pointers of lower complexity. Specifically, we halved the hidden units of the LSTM layers for our neural end-pointer. Based on the wearables data, the early-cut rate of the regression model increased to 18.77% at a threshold of 0.5. The P50, P75, P90 and P99 also increased to 380 ms, 550 ms, 970 ms, and 2,370 ms, respectively. However, at the closest early-cut rate (27.95%, threshold = 0.81), the P50, P75, P90 and P99 of the classification model significantly increased to 1,770 ms, 2,260 ms, 2,480 ms, and 2,690 ms, respectively, much worse than the regression outputs. A possible explanation is that the

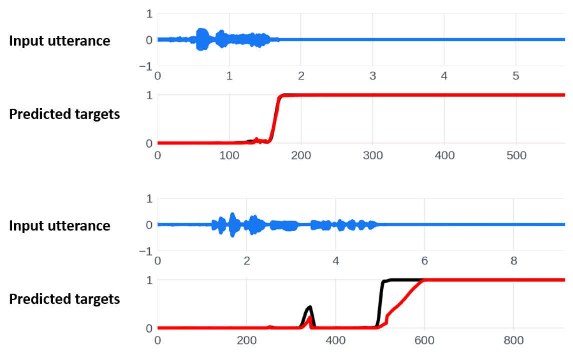

Fig. 3. Sample end-pointer outputs based on classification (black) and regression (red). The training targets of the regression model were adjusted by local pause of queries.

abstraction of the unique prefixes from the original queries helps to generalize the end-pointer on the queries. This benefit is meaningful for end-pointing on the edge, especially for devices with limited computational capabilities.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel regression-based speech end-pointing model, which points to a new direction for solving the early-cut issues in end-pointing without sacrificing latency. We describe how we utilize this model for dynamic end-pointing by adjusting training targets based on expected pause of speech queries. Based on experiments with speech data collected by smartphones and wearables, the proposed model shows a better performance on latency-accuracy tradeoff, with an improved P50, P75 and P90, specifically, while maintaining a comparable early-cut performance to the traditional method. This regression-based approach is quite flexible and can be used to incorporate further speaking behaviors. A better pause model can also be explored by grouping queries into semantics rather than pure prefixes in the future.

7. REFERENCES

[1] Martin Porcheron, Joel E Fischer, Stuart Reeves, and Sarah Sharples, "Voice interfaces in everyday life," in *proceedings of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–12.

[2] Alisha Pradhan, Kanika Mehta, and Leah Findlater, "" accessibility came by accident" use of voice-controlled intelligent personal assistants by people with disabilities," in *Proceedings of the 2018 CHI Conference on human factors in computing systems*, 2018, pp. 1–13.

[3] Radhika Garg and Subhasree Sengupta, "He is just like me: a study of the long-term use of smart speakers by parents and children," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–24, 2020.

[4] Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle Lottridge, "Understanding the long-term use of smart speaker assistants," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–24, 2018.

[5] Won-Ho Shin, Byoung-Soo Lee, Yun-Keun Lee, and Jong-Seok Lee, "Speech/non-speech classification using multiple features for robust endpoint detection," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*. IEEE, 2000, vol. 3, pp. 1399–1402.

[6] Ramalingam Hariharan, Jula Hakkinen, and Kari Laurila, "Robust end-of-utterance detection for real-time speech recognition applications," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*. IEEE, 2001, vol. 1, pp. 249–252.

[7] Rathinavelu Chengalvarayan, "Robust energy normalization using speech/nonspeech discriminator for german connected digit recognition," in *Sixth European conference on speech communication and technology*, 1999.

[8] Shuo-Yiin Chang, Bo Li, Gabor Simko, Tara N Sainath, Anshuman Tripathi, Aäron van den Oord, and Oriol Vinyals, "Temporal modeling using dilated convolution and gating for voice-activity-detection," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5549–5553.

[9] Dawei Liang, Zifan Xu, Yinuo Chen, Rebecca Adaimi, David Harwath, and Edison Thomaz, "Automated detection of foreground speech with wearable sensing in everyday home environments: A transfer learning approach," *arXiv preprint arXiv:2203.11294*, 2022.

[10] Luciana Ferrer, Elizabeth Shriberg, and Andreas Stolcke, "A prosody-based approach to end-of-utterance detection that does not require speech recognition," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)*. IEEE, 2003, vol. 1, pp. I–I.

[11] Matt Shannon, Gabor Simko, Shuo-Yiin Chang, and Carolina Parada, "Improved end-of-query detection for streaming speech recognition.," in *Interspeech*, 2017, pp. 1909–1913.

[12] Shuo-Yiin Chang, Bo Li, Tara N Sainath, Gabor Simko, and Carolina Parada, "Endpoint detection using grid long short-term memory networks for streaming speech recognition.," in *Interspeech*, 2017, pp. 3812–3816.

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Roland Maas, Ariya Rastrow, Chengyuan Ma, Guitang Lan, Kyle Goehner, Gautam Tiwari, Shaun Joseph, and Björn Hoffmeister, "Combining acoustic embeddings and decoding features for end-of-utterance detection in real-time far-field speech recognition systems," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5544–5548.

[15] Aditya Jayasimha and Periyasamy Paramasivam, "Personalizing speech start point and end point detection in asr systems from speaker embeddings," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 771–777.

[16] Shuo-Yiin Chang, Rohit Prabhavalkar, Yanzhang He, Tara N Sainath, and Gabor Simko, "Joint endpointing and decoding with end-to-end models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5626–5630.

[17] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.