Domain-Specific Utterance End-Point Detection for Speech Recognition

Roland Maas, Ariya Rastrow, Kyle Goehner
Gautam Tiwari, Shaun Joseph, Björn Hoffmeister

Amazon.com, USA

{rmaas,arastrow,kgoehner,tgautam,josshaun,bjornh}@amazon.com

Abstract

The task of automatically detecting the end of a device-directed user request is particularly challenging in case of switching short command and long free-form utterances. While low-latency end-pointing configurations typically lead to good user experiences in the case of short requests, such as "play music", it can be too aggressive in domains with longer free-form queries, where users tend to pause noticeably between words and hence are easily cut off prematurely. We previously proposed an approach for accurate end-pointing by continuously estimating pause duration features over all active recognition hypotheses. In this paper, we study the behavior of these pause duration features and infer domain-dependent parametrizations. We furthermore propose to adapt the end-pointer aggressiveness on-the-fly by comparing the Viterbi scores of active short command vs. long free-form decoding hypotheses. The experimental evaluation evidences a 18% relative reduction in word error rate on free-form requests while maintaining low latency on short queries.

Index Terms: end-pointing, end-of-utterance detection, pause, hesitation, online speech recognition

1. Introduction

Consumer products with voice-enabled interfaces are on the rise. Popular examples comprise smartphones, navigation devices, and personal assistants. Typically, the interaction with such devices is user initiated, e.g., by pressing a button or uttering a wake-word. In contrast, the end of an utterance is often to be inferred automatically by the automatic speech recognition (ASR) system. This task of end-point detection, or end-pointing, needs to address two major problems: late end-points due to background noise as well as early end-points in case of long within utterance pauses.

The classical end-pointing approaches are based on different types of voice-activity-detection (VAD), where the end-pointer would trigger if a non-speech region of a certain length is detected. While VAD systems can be based on engineered acoustic features [1–5], such as audio energy [6], pitch [7], zero-crossing rate [8, 9], and cortical features [10], superior performance is typically achieved with deep neural network (DNN) based approaches [11–14]. Besides detecting silence regions, it is furthermore important to differentiate within-sentence pauses from end-of-sentence pauses, which can be achieved by employing prosodic and semantic features [15–25].

All of the methods above are either based on frame-level speech/non-speech classification or speech/non-speech alignments from the 1-best recognition hypothesis. In the latter case, the pause duration is represented by the number of consecutive frames for which the hypothesis has been in a non-speech state. To obtain more robust and accurate pause duration estimates in far-field scenarios, we recently proposed the concept of "expected pause duration" and "expected final pause duration" [26]. The former is given by the weighted sum of pause durations over all active recognition hypotheses using the normalized probability of each hypothesis. The latter expectation is calculated considering only hypotheses in a language model end-state, hence representing an estimate for the end-of-sentence pause.

In this work, we investigate a new level of complexity, which arises in case of domain switching, i.e., when both short commands (such as "play music") as well as long free-form utterances (such as "remind me that I need to go to the hairdresser tomorrow at 3pm and grocery shopping afterwards") are to be supported at the same time. The challenge persists in maintaining low end-pointing latency for commands while not prematurely cutting off longer free-form utterances. After reviewing the expected pause duration-based end-pointer in Section 2, we study the employed end-pointing features and infer parametrizations for short commands in Section 3 and free-form interactions in Section 4. We furthermore propose, in Section 5, to adapt the end-pointer aggressiveness on-the-fly by comparing the Viterbi scores of active short command vs. long free-form decoding hypotheses. The experimental evaluation in Section 6 evidences a 18% relative reduction in word error rate on free-form requests while maintaining low latency on short queries.

2. Decoder-Integrated End-Pointing

In this section, we review the decoder-integrated end-pointer [26], which is based on different pause duration estimates derived from the runtime decoding graph.

Let $X_t = \{x_1, x_2, x_3..., x_t\}$ be the sequence of audio frames until t, and let $S_t^i = \{s_1^i, s_2^i, s_3^i, ..., s_t^i\}, i = [1, N_t]$ be the state sequence of the ith active hypothesis at time t. For any given time t, N_t is the number of active hypotheses. The posterior of the hypothesis is denoted by $p(S_t^i|X_t)$. By L_t^i, we denote the pause duration for the i-th hypothesis. Formally, we can define L_t^i as the largest integer N such that $s_{t-N+1}^i \in S_{\text{NS}} \wedge \cdots \wedge s_t^i \in S_{\text{NS}}$ holds, where S_{NS} denotes the set of all non-speech states [26]. The *best path pause duration* is then given by $L_t^{i_{\max}}$ with $i_{\max} := \arg\max_i p(S_t^i|X_t)$. An estimate of the within-sentence pause length is represented by the *expected pause duration* $\mathbb{D}(L_t) := \sum_i L_t^i p(S_t^i|X_t)$. Furthermore, the end-of-sentence pause can be estimated by the *the expected final pause duration* $\mathbb{D}_{\text{end}}(L_t) := \sum_{i, s_t^i \in S_{\text{end}}} L_t^i p(S_t^i|X_t)$ with S_{end} denoting the set of end-states. With these features, we define the following end-pointer logic, which is to end-point an utterance, if

$$\mathbb{D}(L_t) > T_1 \text{ or } \mathbb{D}_{\text{end}}(L_t) > T_2, \tag{1}$$

where T_1 and T_2 are typically tuned thresholds for within-sentence and end-of-sentence pauses, respectively. Even with rule (1) in place, we still need to safeguard against two bound-

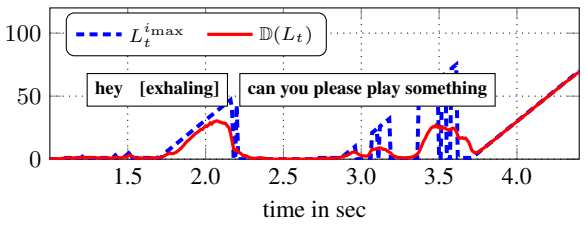

Figure 1: *Feature value comparison of the best path pause duration $L_t^{i\max}$ and the expected pause duration $\mathbb{D}(L_t)$ for an example utterance.*

ary cases [26]: Since the best decoding hypotheses are likely to be empty at the onset of a word, rule (1) may only be applied if a DNN-VAD has indicated silence for a minimum period of time T_3. Furthermore, to shield against confusion in the search graph due to background speech, the end-pointer may exceptionally trigger if $L_t^{i\max} > T_4$ with T_4 being a very large threshold. We will discuss the meaning and behavior of all of these features in more detail in the next section.

3. End-Pointing for Command Domains

In this section, we turn to an in-depth analysis of the decoder-integrated end-pointer. Specifically, we discuss the purpose and behavior of the different types of decoder pause duration features and infer thresholds for domains with short commands, such as "play music", "what's the weather", or "turn on the lights".

3.1. Best path pause duration vs. expected pause duration

To gain a better intuition for the best path pause duration and the expected pause duration features, Fig. 1 depicts their values over time for an example utterance. We can see that the best path pause duration mostly exhibits a linear increase. However, it also shows discontinuities since the locally best hypothesis can change rapidly between frames, which is mainly due to two reasons: The language model "corrects" the hypothesis throughout the decoding run and the decoding graph optimization causes the language model score distribution along a path to not be smooth [26]. We observe, for example, that the 1-best path partly hypothesizes non-speech after the word "play" as long as the word "something" is not fully pronounced. We can also see that the best path pause duration is less affected by noise, such as exhaling sounds. In the absence of noise, the expected pause duration also exhibits a rather linear increase over time, much like the best path pause duration, as shown in Fig. 1 after 3.7sec. However, the expected pause duration value tends to be much more sensitive to speech and noise due to the fact that many sounds can trigger competing hypotheses to transition into speech states. It is also interesting to note that the expected pause duration is a (mostly) continuous features.

In summary, the best path pause duration is generally the more "aggressive" feature, which can spike discontinuously and has the tendency to ignore noise sources and word onsets. The expected pause duration feature, in contrast, is sensitive to competing hypotheses. These observations motivate the choice of different thresholds for both features: the expected pause duration is the default feature used for end-pointing during within-sentence pauses while the best path pause duration is only used as safety valve against cases where background noise is picked up by speech states at decoding. For short requests, preliminary experiments showed that a reasonable range for the within-sentence pause threshold (governed by T_1) is 700-900msec. In contrast, the safety feature against background speech, i.e., the best path pause duration threshold, needs to be much larger (due to its aggressive nature), for example $T_4 = 2T_1$.

3.2. Expected pause duration vs. expected final pause duration

As shown in the previous subsection, the expected pause duration feature is an (approximately) linear measure of pause duration, meaning that a feature value of x corresponds to x frames of pause (at least in the absence of noise). The expected final pause duration, however, has a fundamentally different meaning. Instead of being a measure of pause duration, correlation analyses revealed the feature to be a measure of the ratio of the probability mass of tokens in final state over the probability mass of all active tokens. The distribution of final states is typically highly dependent on the structure of both the acoustic and the language model. As a consequence, the threshold value T_2 for the expected final pause duration feature typically needs to be re-tuned whenever the ASR model changes. In cases of typical short requests, preliminary analyses suggested that a pause duration threshold in the range of 400-700msec after a sentence end represents a reasonable trade-off between latency and accuracy. We illustrate the different behavior with two example utterances, depicted in Fig. 2. It can be observed that the expected final pause duration hardly varies in the case of "play music by", which is considered an in-sentence pause causing the end-pointer to trigger (after approximately 750msec) as the expected pause duration reaches its threshold. In contrast, the subsequent pause after "play music" is considered an end-of-sentence pause, which is why the expected final pause duration spikes allowing the end-pointer to trigger after approximately 500msec.

It is important to point out that the expected final pause duration represents a crucial element in decoder-based end-pointing: This feature not only allows to significantly reduce the end-pointing latency in case of sentence ends, our studies also proved it to be the most noise-robust of all features (relative to the best path pause duration and the expected pause duration). A possible explanation may be that a token can typically only transition out of a language model end-state into a speech state if the next word in the language model is either a likely sentence beginning or if a (potentially high) back-off penalty to a uni-gram state is consumed.

4. End-Pointing for Free-Form Domains

In this section, we describe the end-pointing requirements in case of free-form applications, where users tend to pause longer between words [25], such as setting multiple reminders ("remind me that I need to go to the hairdresser tomorrow at 3pm and grocery shopping afterwards").

Predicting whether the user finished speaking a free-form utterance is a hard problem. One way of approaching this task is to simply relax the end-pointing thresholds and hence allow for a longer pause/silence period before the end-pointer triggers. This, of course, comes at the cost of increased latency. However, we can generally assume latency expectations to differ in free-form and command-type interactions. As an example, if a user asks their device to turn on the lights, they likely desire an immediate reaction. In contrast, after uttering a 30 second-long todo list, it appears acceptable to experience an increased latency before the todo list is being updated.

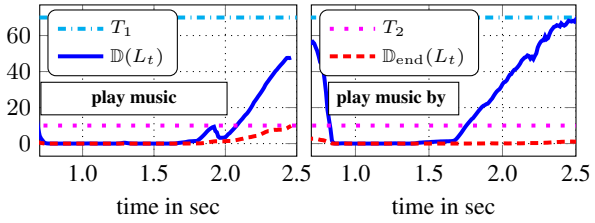

Figure 2: *Feature value comparison of the expected pause duration $\mathbb{D}(L_t)$ and the expected final pause duration $\mathbb{D}_{\text{end}}(L_t)$ for two example utterances.*

4.1. Relaxed thresholds

A direct implication on the end-pointing features discussed in the previous sections is that the thresholds need to be adjusted. Specifically, we found to be beneficial to increase the expected pause duration threshold T_1 to allow for longer pauses in the range of 800msec to 2sec (depending on the application, type and average utterance length).

As for the expected final pause duration, it appears reasonable for free-form utterances to allow the user to pause/hesitate irrespectively of whether the language model is in an end-state or not. In fact, a user might hesitate even longer after the end of a sentence before continuing with the second sentence. This is fundamentally different from the typical short commands, where a language model end-state represents a strong indicator that the entire interaction is completed. In consequence, if we want the end-pointer to always tolerate an increased pause length independent of the language model state, the expected final pause duration needs to be turned off ($T_2 = \infty$).

4.2. Unique challenges

As in the case of short commands, the two major challenges for end-pointing in free-form domains are early and late end-points. However, they exhibit different characteristics.

The problem of late end-points is much more pronounced in free-form interactions, which has multiple reasons: Relaxing the end-pointing thresholds to allow for longer pauses between words and sentences also necessarily increases the window where background speakers and noise can barge in, potentially causing the end-pointer to pick up on these interfering signals. We also found the noise robustness to be further reduced due to the deactivated expected final pause duration feature ($T_2 = \infty$). In addition, since the language models for free-form domain are typically of higher perplexity than language models for short commands, a greater variety of word sequences can be hypothesized and matched against background speech. In our experiments, we noticed that the problem of late end-points specifically occurs in cases, where the background speaker is well intelligible. In case of babble noise-like background speech, the 1-best path in the ASR decoder will typically hypothesize non-speech allowing the end-pointer to trigger.

The tendency of far-field ASR systems to be robust against babble-type background noise by hypothesizing non-speech can, however, be counter-productive. While it is certainly beneficial for the aforementioned sake of noise-robustness, we found it to also cause early end-points. If a user utters an, e.g., 30 second-long query, a segment of unintelligible or mumbled words or words masked by strong background noise can lead to the non-speech hypothesis being ranked first and hence trigger the end-pointer. This problem seems more pronounced in free-form than in short command interactions for two reasons: Short commands, such as "turn on the lights", seem to be less likely to contain mumbled segments (potentially because users concentrate more). Furthermore, the likelihood for an unseen n-gram to occur is higher in free-form domains, in which case, the decoder would have to back off to an m-gram ($m < n$) in order to generate the next word. Since backing off is typically associated with higher decoder penalties, the non-speech hypothesis may be favored.

This represents an inherent conflict for end-pointing in free-form domains: On the one hand, the end-point detection should be robust to background noise, on the other hand, users should not be cut off early in case of partly unintelligible segments. In this contribution, we approached this problem by carefully tuning the thresholds T_3 and T_4 of the two previously mentioned "safety guards" on data sets with both background speech as well as partly unintelligible device-directed speech. A more robust and principled approach may, however, be to incorporate speaker change detection information [27–30] into the end-pointing decision.

5. Domain detection

In the previous sections, we detailed how end-pointing requirements can fundamentally differ for short commands and long free-form utterances. We also motivated different parameter ranges to address these requirements. In case of multi-turn interactions, the domain is typically known before-hand. For example, if the previous device prompt was "What should I remind you of?", then the domain of the next user utterance is known and the according end-pointer configuration can be preloaded. In cases of first-turn interaction, however, the domain is unknown a priori. After the wake-word, a user can either ask to "play music" or "add to my todo list to go grocery shopping, to pick up Aaron from school, and to buy a new dress". In the following, we therefore investigate the task of detecting on-the-fly, i.e., during decoding, which domain-dependent end-pointing configuration is to be used.

To this end, we organize the full language model, which covers all domain types, into two separate groups that are decoded in parallel threads: a) a default language model for typical short queries, b) a free-form language model to handle utterances with certain carrier phrases (such as "remind me of"), followed by a free-form payload (such as "my hairdresser appointment tomorrow at 3pm and ..."). We propose to perform online domain detection by continuously comparing the Viterbi scores from both decoders. Specifically, we switch to the relaxed free-form end-pointer configuration whenever the 1-best decoding hypothesis stems from the free-form language model. Fig. 3 depicts the difference in Viterbi costs of the 1-best hypotheses from the default and the free-form language model, respectively. We see that for short commands, the free-form model quickly diverges (since the utterance does not match the set-reminder carrier phrase). In rare cases, however, the free-form model may spike at first (here: since "can you" may be compatible with the beginning of a set-reminder carrier phrase). We also observe that during the set-reminder carrier phrase, the free-form hypothesis is the most likely one. However, during the free-form payload, the 1-best hypotheses from both language models can have similar scores.

This motivates the design of the following binary state model for this domain detection task with state 0 corresponding to the regular short command end-pointing configuration and state 1 to the long free-form utterance end-pointing configura-

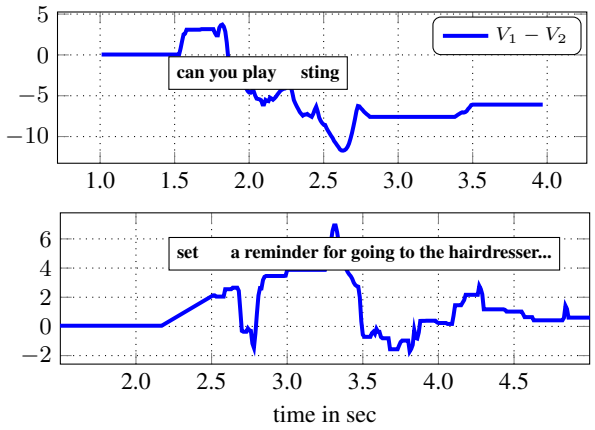

Figure 3: *Difference of the Viterbi costs V_1 and V_2 of the 1-best hypothesis from the default and the free-form language model, respectively. A positive number indicates that the "winning" hypothesis stems from the free-form language model.*

tion: a) transition to state 1 if $V_1 - V_2 > R_1$ for at least K out of the last M frames, b) transition to state 0 if $V_1 - V_2 < R_2$ for at least K out of the last M frames, where V_1 and V_2 denote the Viterbi costs of the 1-best hypothesis from the default and free-form language model, respectively, state 0 is the initial state, and R_1, R_2, K, and M are parameters tuned on a dev set. Preliminary investigations showed that this basic binary sequence classifier achieves false alarm and reject rates of less than 2%.

6. Experiments

In this section, we turn an experimental analysis of the proposed end-pointing configurations for short commands and long free-form utterances. We also assess the proposed on-the-fly domain detection algorithm.

6.1. Experimental setup

For the experimental evaluation, we use an in-house ASR system with a language model organized in multiple decoders, as detailed in Section 5. As dataset, we use an in-house dataset of natural human interactions with voice-controlled far-field devices. We separated the data by domain into two sub-sets: The *command* dataset comprises 3k utterances of typical short intents, such as "play music", "what's the weather", or "turn on the lights". The *free-form* dataset consists of 1k utterances of type "carrier phrase + free-form payload", e.g., "set a reminder for..." or "create a todo list with...". Each free-form utterance contains at least 10 words.

For end-pointing, three different configurations are investigated with the specific parameter values being derived from dev sets. The *regular* configuration is tailored to short commands, as motivated in Section 3, and aggressively tuned for low latency: $(T_1, T_2) = (70, 10)$. The *relaxed* configuration is tailored to free-form interactions, as described in Section 4, and allows for longer pauses: $(T_1, T_2) = (75, \infty)$. The *adaptive* end-pointing mode switches on-the-fly between the regular and the relaxed configuration using the Viterbi cost-based detection mechanism proposed in Section 5.

We evaluate the ASR and end-pointing performance using the following metrics: a) Word Error Rate (WER), b) Early Endpoint Rate (EEPR) describing how often the end-point detector triggers before the end of the last word in an utterance is reached (which usually results in cutting off speech), c) end-pointing latency in msec at P50 and P90 quantiles describing the gap between the end of utterance and the moment the end-point detector triggers.

Table 1: *Comparison of end-pointing accuracy and latency in msec. The WER and EEPR represent relative changes to the respective baseline regular configuration.*

data	config	WER	EEPR	P50	P90
command	regular	-	-	**500**	750
command	relaxed	$\pm 0\%$	-35%	**780**	900
command	adaptive	$\pm 0\%$	$\pm 0\%$	**500**	760
free-form	regular	-	-	730	870
free-form	relaxed	-21%	-62%	830	1020
free-form	adaptive	-18%	-53%	820	1020

6.2. Experimental results

Table 1 shows the accuracy and latency for the different end-pointing configurations and dataset types. We observe that for short commands, the change of the end-pointer from "regular" to "relaxed" does not affect the WER (up to one significant digit). The end-pointing latency, however, increases substantially from 500msec to 780msec at P50, which is mainly due to the deactivation of the expected final pause duration feature. It can also be seen that the adaptive end-pointer achieves the same low latency performance as the regular end-pointer with only a very marginal increase in P90 latency, from 750msec to 760msec. In contrast, for free-form interactions, the regular and the relaxed end-pointer yield significantly different results: Switching to the relaxed end-pointer reduces the WER by 21% relative. The reduction in WER comes, of course, at the cost of an increased latency, from 730msec to 830msec. The adaptive end-pointer performs slightly worse than the relaxed end-pointer on free-form data, achieving an 18% rel. reduction in WER over the baseline. However, it is able to preserve the low latency on command data. A deeper analysis revealed that the cases, where the proposed on-the-fly end-pointer adaptation failed to detect a free-form domain are predominantly caused by invalid or misrecognized key words. For example, if the ASR system recognizes "rewind" instead of "remind", then the free-form decoder may be outperformed by default decoder, similarly to the upper case in Fig. 3. These scenarios are, however, quite rare. In summary, we can state that the regular end-pointer is optimized for typical short commands, the relaxed end-pointer for free-form interactions, and the Viterbi cost-based adaptation mechanism effectively switches between the two configurations.

7. Conclusions

We presented detailed analyses of decoder-integrated pause duration features for robust end-pointing in far-field applications and inferred guidelines for the according parameter values in domains with short commands as well as domains with long free-form utterances. We also proposed an approach to on-the-fly domain detection in order to switch between both configurations. The experimental results show a relative reduction in WER of 18% on free-form interactions while maintaining low latency on short commands.

8. References

[1] W.-H. Shin, B.-S. Lee, Y.-K. Lee, and J.-S. Lee, "Speech/non-speech classification using multiple features for robust endpoint detection," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, 2000, pp. 1399–1402.

[2] T. T. Kristjansson, S. Deligne, and P. A. Olsen, "Voicing features for robust speech detection," in *Proceedings Interspeech*, 2005, pp. 369–372.

[3] X. Li, H. Liu, Y. Zheng, and B. Xu, "Robust speech endpoint detection based on improved adaptive band-partitioning spectral entropy," in *Proceedings of the Life System Modeling and Simulation International Conference on Bio-Inspired Computational Intelligence and Applications*, 2007, pp. 36–45.

[4] M. Fujimoto, K. Ishizuka, and T. Nakatani, "Study of integration of statistical model-based voice activity detection and noise suppression." in *Proceedings Interspeech*, 2008, pp. 2008–2011.

[5] S. O. Sadjadi and J. H. L. Hansen, "Unsupervised speech activity detection using voicing measures and perceptual spectral flux," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 197–200, 2013.

[6] K.-H. Woo, T.-Y. Yang, K.-J. Park, and C. Lee, "Robust voice activity detection algorithm for estimating noise spectrum," *Electronics Letters*, vol. 36, no. 2, pp. 180–181, 2000.

[7] R. Chengalvarayan, "Robust energy normalization using speech/nonspeech discriminator for German connected digit recognition," in *Procedings Sixth European Conference on Speech Communication and Technology*, 1999.

[8] A. ITU, "Silence compression scheme for G. 729 optimized for terminals conforming to recommendation V. 70," *ITU-T Recommendation G*, vol. 729, 1996.

[9] L. Lu, H.-J. Zhang, and H. Jiang, "Content analysis for audio classification and segmentation," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 7, pp. 504–516, 2002.

[10] S. Thomas, S. H. R. Mallidi, T. Janu, H. Hermansky, N. Mesgarani, X. Zhou, S. A. Shamma, T. Ng, B. Zhang, L. Nguyen, and S. Matsoukas, "Acoustic and data-driven features for robust speech activity detection." in *Proceedings Interspeech*, 2012.

[11] N. Ryant, M. Liberman, and J. Yuan, "Speech activity detection on youtube using deep neural networks." in *Proceedings Interspeech*, 2013, pp. 728–731.

[12] Xiao-Lei Zhang and Ji Wu, "Deep belief networks based voice activity detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 697–710, 2013.

[13] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7378–7382.

[14] R. Maas, S. H. K. Parthasarathi, B. King, R. Huang, and B. Hoffmeister, "Anchored speech detection," in *Proceedings Interspeech*, 2016, pp. 2963–2967.

[15] D. O'Shaughnessy, "Recognition of hesitations in spontaneous speech," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992, pp. 521–524.

[16] E. Shriberg, R. A. Bates, and A. Stolcke, "A prosody only decision-tree model for disfluency detection," in *Proceedings Fifth European Conference on Speech Communication and Technology, EUROSPEECH*, 1997.

[17] R. Hariharan, J. Häkkinen, and K. Laurila, "Robust end-of-utterance detection for real-time speech recognition applications," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001*, 2001, pp. 249–252.

[18] L. Ferrer, E. Shriberg, and A. Stolcke, "Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody," in *Proceedings Seventh International Conference on Spoken Language Processing*, 2002.

[19] J. Edlund, M. Heldner, and J. Gustafson, "Utterance segmentation and turn-taking in spoken dialogue systems," *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen*, pp. 576–587, 2005.

[20] L. Ferrer, E. Shriberg, and A. Stolcke, "A prosody-based approach to end-of-utterance detection that does not require speech recognition," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2003, pp. 605–608.

[21] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper, "Enriching speech recognition with automatic detection of sentence boundaries and disfluencies," *Trans. Audio, Speech and Lang. Proc.*, vol. 14, no. 5, pp. 1526–1540, Sep. 2006.

[22] H. Arsikere, E. Shriberg, and U. Ozertem, "Computationally-efficient endpointing features for natural spoken interaction with personal-assistant systems," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3241–3245.

[23] A. Raux and M. Eskenazi, "Optimizing endpointing thresholds using dialogue features in a spoken dialogue system," in *Proceedings 9th SIGdial Workshop on Discourse and Dialogue*. Association for Computational Linguistics, 2008, pp. 1–10.

[24] K. Audhkhasi, K. Kandhway, O. D. Deshmukh, and A. Verma, "Formant-based technique for automatic filled-pause detection in spontaneous spoken English," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 4857–4860.

[25] H. Arsikere, E. Shriberg, and U. Ozertem, "Enhanced end-of-turn detection for speech to a personal assistant," in *Proceedings AAAI Spring Symposium on Turn-taking and Coordination in Human-Machine Interaction*, 2015.

[26] B. Liu, B. Hoffmeister, and A. Rastrow, "Accurate endpointing with expected pause duration." in *Proceedings Interspeech*, 2015, pp. 2912–2916.

[27] S. Chen and P. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," in *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[28] D. Lilt and F. Kubala, "Online speaker clustering," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[29] S. E. Tranter and D. A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, 2006.

[30] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A review of recent research," *IEEE Transactions On Acoustics Speech and Language Processing (TASLP), special issue on New Frontiers in Rich Transcription*, 2011.