Keyword Spotting for Google Assistant Using Contextual Speech Recognition

Assaf Hurwitz Michaely, Xuedong Zhang, Gabor Simko, Carolina Parada, Petar Aleksic

Google Inc.

{amichaely,xuedong,gsimko,carolinap,apetar}@google.com

Abstract

We present a novel keyword spotting (KWS) system that uses contextual automatic speech recognition (ASR). For voice-activated devices, it is common that a KWS system is run on the device in order to quickly detect a trigger phrase (e.g. "Ok Google"). After the trigger phrase is detected, the audio corresponding to the voice command that follows is streamed to the server. The audio is transcribed by the server-side ASR system and semantically processed to generate a response which is sent back to the device. Due to limited resources on the device, the device KWS system might introduce false accepts (FA) and false rejects (FR) that can cause an unsatisfactory user experience.

We describe a system that uses server-side contextual ASR and trigger phrase non-terminals to improve overall KWS accuracy. We show that this approach can significantly reduce the FA rate (by 89%) while minimally increasing the FR rate (by 0.2%). Furthermore, we show that this system significantly improves the ASR quality, reducing Word Error Rate (WER) (by 10% to 50% relative), and allows the user to speak seamlessly, without pausing between the trigger phrase and the voice command.

Index Terms: speech recognition, keyword spotting

1. Introduction

Keyword spotting systems are a common component in speech-enabled devices. KWS systems enable hands-free speech recognition experience by detecting a trigger phrase used to initiate the interaction with a device. KWS system accuracy is critical for any voice input device, especially for devices that have only voice input. For example, a user may say "Ok Google, play some music", resulting in music playing without the user ever touching the device. Once a trigger phrase is detected on device, typically the connection is opened to the server and the audio corresponding to the rest of the query (e.g. "play some music") is sent for transcription using a server-side ASR system. Other common trigger phrases used in commercial voice enabled products are: "Alexa", "Hey Siri" and "Hey Cortana". KWS systems usually run on devices that have constraints in memory footprint and computational power. These constraints can often result in limited KWS system accuracy [1, 2, 3, 4].

To measure the accuracy of a KWS system, we use false accept rate and false reject rate as metrics. A false accept occurs when the system detects a trigger phrase although none was said. False accepts can result in the device responding to random commands and interrupting users, hence negatively affecting the user experience. A false reject occurs when the user says the trigger phrase, but the system does not detect it and rejects the query. A good KWS system has very small FA and FR rates.

In this paper we describe a system that uses two KWS systems. The primary KWS system runs on-device and determines which queries to send to the server. The secondary KWS system runs on the server side and is used to suppress queries that were falsely accepted by the primary on-device KWS system. The secondary KWS system takes advantage of the server-side ASR system, including larger models and more resources, to provide more accurate trigger phrase detection and eliminate a large percentage of false accepts. In the proposed system, once the on-device KWS triggers, the audio corresponding to both the trigger phrase and the query is sent to the server. The server-side ASR system decodes the concatenated audio using a context-aware recognizer for improved KWS performance. Once the ASR result is complete, the server-side KWS task is reduced to string matching: If the ASR result does not contain the trigger-phrase, the query is suppressed. Otherwise, the query is considered a true accept and is processed further.

We evaluate the proposed system on several human-transcribed test sets. We show that adding the server-side KWS system leads to a significant reduction of FA rates, with only a minimal increase of FR rates. Furthermore, we show that our system improves speech recognition accuracy, significantly reducing word error rate (WER). It does so by continuously decoding the audio corresponding to both trigger phrase and the query, thus:

1. Avoiding the negative effect of potentially incorrect segmentation of the trigger phrase audio and the query audio.
2. Improving the modelling of coarticulation effects, and improving the handling of noise, thanks to the additional audio decoded.

We organize the paper as follows. In section 2 we describe the main components of our KWS system. In section 3 we describe the adjustments made to the first pass language model (LM). In section 4 we describe the endpointing system. In section 5 we describe the final system modifications, including second pass rescoring (section 5.1), and trigger phrase detection and removal (section 5.2). In section 6 we describe the experimental setup and analyze the results. Finally, in section 7 we present our conclusions.

2. Client-Server KWS System Design

In this section we describe the main components of our combined KWS system consisting of both on-device and server-side KWS systems.

The on-device KWS system continuously listens and initiates a connection to the server when it detects a trigger phrase. Once a trigger phrase is detected, the audio corresponding to the trigger phrase and the query are sent to the server in a streaming fashion as shown in Figure 1. In this paper we do not modify the on-device KWS system behavior.

The server-side contextual ASR system is used to provide server-side KWS (trigger phrase detection). If the resulting

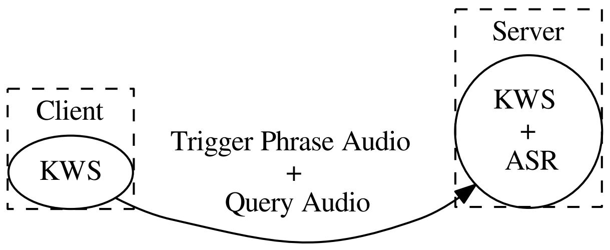

Figure 1: *Client-Server keyword spotting system.*

transcript does not contain the trigger phrase, the query is rejected (the device does not respond to the command). If the transcript contains the trigger phrase, the query is accepted and processed. Finally, the trigger phrase is removed from the final transcript as part of denormalization (e.g. “Ok Google stop” $\rightarrow$ “stop”), before being further processed.

The following sections describe the approaches we considered and solutions designed to enable using server side contextual ASR system for improved KWS performance and transcription.

3. Language Model (LM) Adaptation

The ASR system used in this work is a 2-pass ASR system utilizing a smaller first pass LM and a larger second pass LM. The LMs are trained on anonymized query examples, such as “what time is it”, most of which do not include trigger phrases. Therefore, any sentence starting with the trigger phrase (e.g. “Ok Google what time is it”) will usually have a high cost in the baseline LM. Using such an LM for KWS and for speech recognition of queries that include trigger phrases will results in both increased FR rates and increased WER. In order to avoid this degradation, the LM must be adapted to better model queries that do start with a trigger phrase. This section describes several adaptation approaches we considered.

3.1. LM Retraining

The first approach we considered was retraining the LMs with data consisting of queries that do include trigger phrases. This approach has two problems:

1. Changing the set of supported trigger phrases would require retraining of the LM.
2. The retrained LM would no longer match well recognition tasks in which the utterances do not start with a trigger phrase.

3.2. Contextual ASR

On-the-fly rescoring (biasing) [5, 6] is a framework for on-the-fly changing of LM costs corresponding to certain n-grams using rescoring (biasing) models. A biasing model consists of n-grams that need to be biased and the weights corresponding to each of the n-grams. Biasing is applied during decoding and is triggered based on particular conditions. The biasing n-grams can include any word, that is, words in the LM’s vocabulary as well as out of vocabulary words. Class based symbols [7] can also be biased (e.g. “$TIME”, “$DATE”).

During decoding, the cost from the LM, G, is combined with the cost from the biasing model, $\mathcal{B}$, as follows:

$$s(w|H) = \begin{cases} s_G(w|H) & \text{if } (w|H) \notin \mathcal{B} \\ C(s_G(w|H), s_B(w|H)) & \text{if } (w|H) \in \mathcal{B} \end{cases}, \quad (1)$$

where $s_G(w|H)$ is the raw score from the main model G for the word w leaving history state H and $s_B(w|H)$ is the raw score for the biasing model. C is some combination function, e.g. $min(x, y)$. Note that this approach only modifies the LM scores of n-grams Hw for which the biasing model provides an explicit weight. This differs from regular LM interpolation and is motivated by the fact that the support of the biasing model is much sparser than that of the main LM.

In a KWS system, rescoring can be used on the server side to boost the probability of the trigger-phrases at the start of the sentence. This rescoring would be activated only when the streaming audio includes both trigger-phrase and query audio. With this approach the contextual ASR system will still provide good recognition accuracy on utterances that do not contain trigger phrase audio. This approach is also modular and flexible: supporting a new trigger phrase would require a simple addition of the corresponding n-grams to the biasing model. This approach solves both problems associated with LM retraining approach.

Rescoring will ensure that the conditional probability of the trigger phrase at the beginning of the sentence is sufficiently high:

$$\mathbb{P}(\text{“Ok Google”}|\text{”}{<}S{>}\text{”})$$

However, the conditional probability of the word following the trigger phrase, w:

$$\mathbb{P}(w|\text{”}{<}S{>}\text{ Ok Google”})$$

will not be well estimated, because the history “$<S>$ Ok Google” is not well represented in the LM training data. This can result in a system with sufficiently low FR but increased WER. In order to solve this problem we introduce a trigger phrase non-terminal, described in the following section.

3.3. Trigger Phrase Non-Terminal

A non-terminal is a symbol in a finite-state-transducer (FST) [8] which can be replaced at recognition time by another FST. One common use of non-terminals is for class based language models [7], in which a non-terminal symbol, e.g. $DATE, is replaced with an FST representing the class grammar. The non-terminal symbol can also be replaced dynamically at recognition time [9], using contextual or personalized information.

In order to dynamically adapt the first pass LM to the task of decoding audio with the trigger phrase included, we introduce a trigger phrase non-terminal, $TRIGGER_PHRASE, as an arc at the LM FST’s start state (Figure 2).

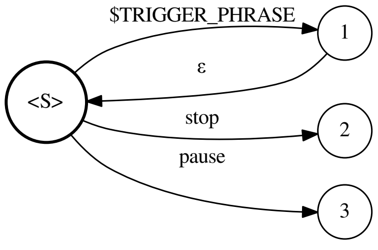

Figure 2: *$TRIGGER_PHRASE non-terminal in the first pass LM FST.*

At recognition time, if the trigger phrase audio is sent to the server, \$TRIGGER_PHRASE is dynamically replaced with an FST containing the relevant trigger phrases (Figure 3). If the trigger phrase audio is not sent to the server, \$TRIGGER_PHRASE is replaced with an empty FST, resulting in an undadapted LM.

We adjust the LM cost of \$TRIGGER_PHRASE using on-the-fly rescoring. We choose a cost that optimizes the FR and FA rates, by tuning on a dev set. To prevent multiple looping through \$TRIGGER_PHRASE, we set a low cost for its first occurrence only ("\$TRIGGER_PHRASE |<S>") and a higher cost for any additional occurrences (e.g. "\$TRIGGER_PHRASE |<S> \$TRIGGER_PHRASE").

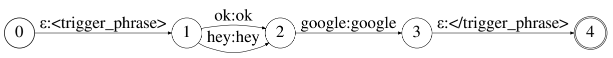

Figure 3: *The trigger phrase FST.*

The opening and closing tags ("<trigger_phrase>" and "</trigger_phrase>") mark the span of the trigger phrase. These tags are used for trigger phrase detection and removal (see 5.2).

4. Endpointing System

The quality of the overall system significantly depends on the endpointer system used. In this section we describe the endpointer system used in our system and various modifications applied.

4.1. Background

The task of endpointing is to detect quickly and accurately when the user started and finished speaking their query. A typical setup [10] for endpointing is to use a probabilistic voice activity classifier (or end of query classifier) that classifies frames as speech/non-speech (or query-complete vs query-not-complete) and threshold the framewise posteriors to reach hard decisions.

For simplicity it is often assumed that the output labels y_t of the classifier are conditionally independent. Then, the conditional probabilistic model expresses the posterior probabilities $\mathbb{P}(y_t|x_t, \lambda)$ given an acoustic feature vector sequence $x_T = x_1, \dots, x_T$ and model parameters λ.

In order to turn these posteriors into a hard decision, the posterior scores are thresholded. The formula for the final decision can be expressed in the form of

$$D = \underset{t}{\operatorname{argmin}} \left(t > \sigma_w \wedge \forall_{t-\sigma_w}^{t} (\mathbb{P}\left(y_t|x_t, \lambda\right) > \sigma_p) \right)$$

where D is the frame at which the endpointing decision is made, σ_p is the threshold used with the posterior scores and σ_w is the wait-time threshold (the number of consecutive frames where the same decisions were made).

4.2. Endpointer System Modifications Overview

Sending and decoding the entire audio segment can cause problems for the endpointer. For example, users typically pause after saying the trigger phrase, which might trigger the endpointer. In order to solve this problem, we can modify the formula for endpoint-decision to factor in the recognition results when choosing the thresholds as follows:

$$D = \underset{t}{\operatorname{argmin}} \left(t > \sigma_w(R) \wedge \forall_{t-\sigma_w(R)}^{t} (\mathbb{P}\left(y_t|x_t, \lambda\right) > \sigma_p) \right)$$

Here $\sigma_w(R)$ signifies that the threshold is dependent on the recognition result R. Note that we could similarly introduce a recognition-based threshold $\sigma_p(R)$, however, we did not pursue the idea in this work.

There are three modifications that we had to make for endpointing to work as expected:

1. Do not endpoint aggressively when the recognition consists solely of the trigger phrase.
2. Do not endpoint aggressively when the recognition of the trigger phrase is in flux.
3. Endpoint early if the ASR does not detect the trigger phrase after decoding the entire trigger-phrase audio.

4.3. User Pauses After Trigger Phrase

Users might pause between saying the trigger phrase and the main query. To prevent cutting off the user during this pause, the endpointer should have a longer-wait time if the recognition consists of only the trigger phrase. This detection can be done by matching the recognition text, R, in which case the endpointer's wait-time is increased.

$$\sigma_w(R) = \begin{cases} \pi_L, & \text{if } R = \text{"TP"} \\ \pi_S, & \text{otherwise} \end{cases} \tag{2}$$

where π_L is the (long) wait-time for incomplete queries (e.g., 800 frames), π_S for (short) complete queries (e.g., 30 frames), and "TP" is the textual form of the trigger phrase.

4.4. Trigger Phrase Recognition In Flux

Recognition in flux is a harder problem to tackle, since we do not have prior information about what the partial recognitions might look like. However, we know how long the trigger phrase audio is, and we can force the endpointer to not make any endpointing decisions before having decoded the entire trigger phrase audio.

$$\sigma_w(R) = \begin{cases} \pi_L, & \text{if } Fr(R) < Fr(\text{TP}_{audio}) \\ \pi_S, & \text{otherwise} \end{cases} \tag{3}$$

where $Fr(R)$ is number of frames processed to reach recognition R and $Fr(\text{TP}_{audio})$ is the length in frames of the trigger-phrase audio.

4.5. Trigger Phrase Not Detected

Finally, the endpointer can alleviate the problem in the low-probability but important case of false accepts. The endpointer can make an early end of the utterance decision if there is no trigger phrase detected after processing the trigger phrase audio, which saves bandwidth and server load.

$$\sigma_w(R) = \begin{cases} 0, & \text{if } R \neq \text{".*TP.*"} \wedge Fr(R) \geq Fr(\text{TP}_{audio}) \\ \pi_S, & \text{otherwise} \end{cases} \tag{4}$$

where the inequality of R expresses that the right-hand side regular expression does not match R.

The final equation that combines these decisions can be written as follows:

$$\sigma_w(R) = \begin{cases} \pi_L, & \text{if } R = \text{"TP"} \vee Fr(R) < Fr(\text{TP}_{audio}) \\ 0, & \text{if } R \neq \text{".*TP.*"} \wedge Fr(R) \geq Fr(\text{TP}_{audio}) \\ \pi_S, & \text{otherwise.} \end{cases} \tag{5}$$

5. ASR System Modifications

5.1. Second Pass Rescoring

In section 3 we described how the first pass LM is dynamically modified to enable recognition of trigger phrases. However, in a two-pass scoring system, the second pass LM must also be adapted. Similar to the first pass LM, if the second pass LM assigns a low probability to sentences starting with a trigger phrase, this will lead to increased FR rate and WER.

We solve this problem by modifying the 2nd pass LM to ignore the trigger phrase portion of the transcript and only rescore the part of the transcript corresponding to the query. The trigger phrase tokens retain the LM cost that was assigned to them in the first pass. For example, if the transcript is "<trigger_phrase> Ok Google </trigger_phrase> volume up", then the 2nd pass LM will only rescore "volume up", while "<trigger_phrase> Ok Google </trigger_phrase>" will retain the score assigned to it in the first pass.

5.2. Trigger Phrase Detection and Removal

The server side KWS system considers that the trigger phrase is detected if the trigger phrase tags are present in the top hypothesis (e.g. "<trigger_phrase> Ok Google </trigger_phrase> stop"). Otherwise, the server system discards the audio, and the query is rejected.

After detection, the trigger phrase tags and enclosed text are removed as part of the denormalization process, resulting in the transcription of the query without the trigger phrase. For example, the hypothesis "<trigger_phrase> Ok Google </trigger_phrase> stop" would be denormalized to "stop". Removing the trigger phrase text enables seamless integration of the server-side KWS functionality into the ASR pipeline in which only the query audio is decoded. Otherwise, if the trigger phrase text were kept in the hypothesis, further modification would be required in the semantic processing of the query to ignore the trigger phrase.

5.3. Summary of system adaptation changes

1. A $TRIGGER_PHRASE non-terminal arc is added to the first pass LM. The arc's cost is adjusted using contextual biasing (section 3.3).
2. Endpointer (section 4)

   Less-aggressive endpointing when the recognition consists solely of the trigger phrase (section 4.3).

   Less-aggressive endpointing when the recognition of the trigger phrase is in flux (section 4.4).

   Early endpointing if the trigger phrase is not detected after decoding the entire trigger-phrase audio (section 4.5).
3. The trigger phrase text is ignored during second pass rescoring (section 5.1).
4. The trigger-phrase text is detected and removed during denormalization (section 5.2).

6. Experimental Results

In this section we describe the experimental setup and analyze the results.

6.1. Experimental Setup

The baseline system consists of a long short-term memory acoustic model [11, 12] and a Katz smoothed [13] 5-gram FST [8, 14] LM, pruned to 100M n-grams and trained using Bayesian interpolation [15]. The system also includes a larger second pass LM, trained on the same data [16]. The training data does not include transcripts with trigger phrases. The baseline system does not decode the trigger-phrase audio of the utterance.

We ran experiments on the baseline system and the two experimental systems:

- SYS-A. This system decodes the trigger phrase audio and the query audio, and includes the modifications described to the first pass LM (section 3), the second pass LM (section 5.1) and the trigger phrase detection and removal (section 5.2). SYS-A does not include endpointer modifications described in section 4.
- SYS-B. This system includes all the modification described in the paper. That is, it is the same as SYS-A with the added modifications for the endpointer system, describes in section 4.

To evaluate our models, we ran two types of experiments:

1. Offline experiments on human-transcribed test sets.
2. Live traffic side-by-side experiments rated by humans.

6.2. Offline Experiments

Our offline experiments included two types of test sets, all consisting of anonymized utterances in British and American English.

1. Positive. Utterances that start with a trigger-phrase (e.g. "Ok Google"). These utterances are transcribed by human raters, and used for measuring WER and FR rate.
2. Negative. Utterances that were accepted by a client-side trigger-phrase detector, but determined (by human raters) to not include a trigger-phrase. This type of test set is used for measuring the system's effect on the FA rate. Note that because we are not changing the on-device KWS system, we do not measure FA or FR rate of the device, and our testing data consists of utterances that were accepted by the on-device KWS system. Accordingly, we consider the baseline system's FA rate to be 100%, its FR rate to be 0%, and we measure the experimental system's FA and FR rates relative to the baseline.

6.2.1. Ground Truth

The utterances were classified by human raters as positive (contain a trigger-phrase) or negative (do not contain a trigger phrase). The positive utterances were then transcribed by human raters, omitting the trigger-phrase from the transcription. For example, the utterance "hey google stop", would be transcribed as "stop". This allows fair comparison of the baseline system (which does not decode the trigger phrase audio) with the experimental systems (which do decode the trigger phrase audio but remove the trigger-phrase from the final transcript, as described in section 5.2).

6.3. Live traffic side-by-side (SxS) experiments

In these experiments each utterance is automatically transcribed by the two live systems, baseline and experimental. If the two resulting ASR transcripts are not identical, they are kept for rating. The experiment continues until a certain number of different transcripts is collected. The transcripts are then, together with corresponding audio, sent for rating. Each utterance is rated by two humans (and a third in case of a tie). For each of the SxS experiments, we present the following:

% Changed: The percentage of utterances in which the two systems produced different transcripts.

Wins/Losses: The ratio of wins to losses, comparing the experimental system to the baseline. A win occurs when the transcript of the experimental system is rated higher, a loss occurs when the transcript of the baseline is rated higher.

6.4. Results

In this section we present the results obtained by running experiments for each of the three systems on the positive and negative test sets.

Note that we do not report false accept or false reject rates for the baseline system. This is because the baseline system does not decode the trigger-phrase audio, only the main query audio, and does not suppress any requests. It can be seen as having 0% FR rate and 100% FA rate.

6.4.1. Negative Test Set Experiments

As this test set was created by collecting false client-side detections, the baseline system (with no server-side KWS) would have 100% FA. The two experimental systems (SYS-A and SYS-B) obtained an identical FA rate on this test set, 10.58% (Table 1), a decrease of 89.42% compared to the baseline system.

Name	Size	FA [%]	FA decrease [%]
NEGATIVE	832	10.58	89.42

Table 1: *False accepts rates obtained by SYS-A and SYS-B on the negative test set in American English. Note that the baseline system does not decode the trigger-phrase audio, and hence does not suppress any requests (its FA rate is 100%).*

6.4.2. Positive Test Set Experiments

We ran experiments for the three systems on six test sets in British English and three test sets in American English. Table 2 shows the results obtained for the British English test sets. These test sets were split by trigger-phrase (”Hey Google” or ”Ok Google”) and noise level (clean, noisy, very noisy). Table 3 shows the results for the three American English test sets. These test sets were split by query type:

1. “timer” and “alarm” test sets consist of utterances corresponding to timer and alarm control commands, respectively.
2. “general” test set consists of all types of voice queries.

In all of these test sets, the baseline WER (BASE) is reduced by decoding the trigger-phrase audio (SYS-A) and further reduced by adding the endpointer modifications (SYS-B). For example, on the test set “general”, the WER is reduced from 10.3% (BASE) to 7% when adding trigger-phrase audio decoding (SYS-A), and further decreases to 6.8% when adding the endpointer modifications (SYS-B). The two experimental systems obtained an identical FR rate on these test sets. The FR rate on the general American English test set is .15%, while “timer” and “alarm” test sets show higher FR rates, 2.7% and 3.5%. This is likely due to high levels of noise caused by the alarm and timer firing. Similarly on the test sets in British English, the FR increases with the amount of noise, ranging from .07% to .83%.

Test Set	Size	WER [%]			FR [%]
		BASE	SYS-A	SYS-B	SYS-A,B
hey clean	2102	15.5	10.9	9.7	.19
hey_noisy	3329	17.9	14.7	13.4	.24
hey_v_noisy	841	26.8	24.2	23.1	.83
ok_clean	5870	14.7	10.5	9.7	.07
ok_noisy	5832	19.2	14.1	12.7	.22
ok_v_noisy	1463	28.8	20.1	18.8	.61

Table 2: *Results on the positive test sets in British English. “BASE” is a system that does not decode the trigger-phrase audio. “SYS-A” is a system that does decode the trigger-phrase audio but does not have the endpointer adjustments described in 4. “SYS-B” is the full system described in this paper, including the endpointer adjustments. The FR rate for SYS-A and SYS-B are identical. Note that the baseline system does not decode the trigger-phrase audio, so it does not suppress any requests, and its FR rate is 0 %.*

Test Set	Size	WER [%]			FR [%]
		BASE	SYS-A	SYS-B	SYS-A,B
general	82987	10.3	7	6.8	.15
timer	1808	21	11.4	11.3	3.5
alarm	1631	16.6	8.2	7.8	2.7

Table 3: *Results on the positive test sets in American English. “BASE” is a system that does not decode the trigger-phrase audio. “SYS-A” is a system that does decode the trigger-phrase audio but does not have the endpointer adjustments described in 4. “SYS-B” is the full system described in this paper, including the endpointer adjustments. The FR rate for SYS-A and SYS-B are identical. Note that the baseline system does not decode the trigger-phrase audio, so it does not suppress any requests, and its FR rate is 0 %.*

6.5. SxS Experiments

Table 4 shows the results of live Side By Side (SxS) experiments, comparing the ASR transcriptions produced by the baseline system (which does not decode the trigger-phrase audio) to those of SYS-B (which does decode the trigger-phrase audio). These experiments all show a significant improvement gained by SYS-B compared to the baseline, with between 17% and 42% of the transcripts changed, and with a win/loss ratio between 2.6 and 11.6.

6.6. Improvements Analysis

In this section we further analyze specific problems solved by our system, leading to improved overall performance.

Exp	Changed [%]	Wins/Losses
en-us	42.47	11.667
en-gb	26.86	2.29
en-au	17.62	2.87
de-de	25.57	2.37
fr-fr	25.61	2.61

Table 4: *SxS results comparing SYS-B to the baseline system. The experiments were run in English(USA), English(Britain), English(Australia), German(Germany) and French(France). All languages show a significant improvement, with over 12% of the transcripts changed, and a win to loss ratio of over 2 for all experiments.*

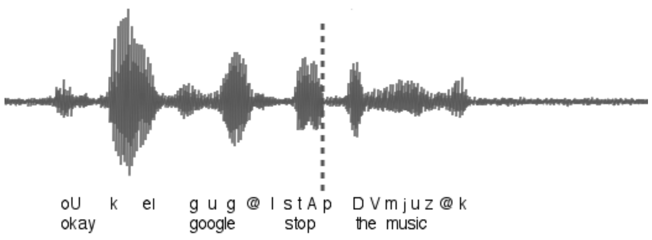

Figure 4: *An example where decoding the trigger-phrase audio and the main audio together improves the recognition. The vertical gray line represents the end of the trigger-phrase audio as recorded by the on-device KWS system. Due to background noise and a short pause between "google" and "stop", the audio for "stop" ends before the main audio.*

6.6.1. *Speech Onset Problem*

Since the baseline system does not decode the trigger-phrase audio, it must determine the speech onset after the trigger-phrase, at which it will start decoding. When the user makes a pause between the trigger-phrase and the command (e.g. "Ok Google...stop"), it is easier to detect where the trigger-phrase ends and the query begins. However, when the user speaks more naturally, without the pause, incorrect segmentation of the audio often results in the beginning of the query being misrecognized (see Figure 4). At times this results in an empty transcription (especially with short utterances). Other times it results in an incomplete or incorrect transcription, leading to a failed voice action. By decoding the trigger-phrase audio, the need for segmentation is eliminated, and the system can correctly model the coarticulation effect.

Examples of recognition improvements:

- "new ad" $\rightarrow$ "can you add"
- "which off alarm" $\rightarrow$ "switch off alarm"
- "soft rock" $\rightarrow$ "play soft rock"

6.6.2. *Early Cutoff Problem*

The endpointer modification (described in section 4.3) improves handling of the complementary problem to the one described above: the case where the user makes a lengthy pause between the trigger-phrase and the command. In system SYS-A (decoding trigger-phrase audio without adjusting the endpointing), endpointing would trigger during the pause, ending the decoding before the audio of the actual query is processed, and resulting in an empty transcript. Adjusting the endpointing to expect a pause after the trigger phrase addresses this problem.

6.6.3. *Second Speaker Problem*

Since the baseline system transcribes only the query audio, a second speaker (e.g. another person, TV) speaking between the trigger-phrase and the query can cause unwanted insertions or misrecognitions. The proposed system solves this problem since the decoder sees the trigger-phrase audio from the user, learns to ignore the background speech from additional speakers, and only transcribes user's query.

Some examples of improved recognition quality by using the trigger phrase audio to better eliminate the noise coming from a second speaker:

- First speaker: "play some latest English songs".

  Second speaker (human): "15 minutes later".

  Correction: "55th of latest English songs" $\rightarrow$ "play some latest English songs".
- First speaker: "dim the living room light".

  Second speaker (television): "instead of Renee".

  Correction: "better brunette dim the living room light" $\rightarrow$ "dim the living room light".
- First speaker: "turn down".

  Second speaker (television): "An oath to United States".

  Correction: "United States" $\rightarrow$ "turn down".

7. Conclusion

We described a system for improving keyword spotting quality that uses server-side contextual ASR. This system decodes the trigger phrase audio sent from the device for improved performance. Our system significantly reduces the rate of FAs by 89 %, with a small increase in the FR rate of 0.2%. In addition, our system significantly improves the ASR quality, reducing WER by 10-50% and showing significant improvements in SxS experiments in several languages. As future work, we plan to further expand support to other languages and evaluate our system on various voice activated devices.

8. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4087–4091.

[2] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5236–5240.

[3] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting." in *INTERSPEECH*, 2015, pp. 1478–1482.

[4] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *arXiv preprint arXiv:1701.04313*, 2017.

[5] K. B. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Interspeech 2015*, 2015.

[6] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to google speech recognition," in *Interspeech 2015*, 2015.

[7] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, pp. 467–479, 1992.

[8] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.

[9] L. Vasserman, B. Haynor, and P. Aleksic, "Contextual language model adaptation using dynamic classes," in *Spoken Language Technology Workshop (SLT), 2016 IEEE.* IEEE, 2016, pp. 441–446.

[10] M. Shannon, G. Simko, S. yiin Chang, and C. Parada, "Improved end-of-query detection for streaming speech recognition," in *Interspeech 2017*, 2017.

[11] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, 2014.

[12] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Interspeech*, 2014.

[13] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987, pp. 400–401.

[14] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *CIAA 2007*, ser. LNCS, vol. 4783, 2007, pp. 11–23, `http://www.openfst.org`.

[15] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *INTERSPEECH*, 2011, pp. 1429–1432.

[16] P. Jyothi, L. Johnson, C. Chelba, and B. Strope, "Distributed discriminative language models for google voice search," in *Proceedings of ICASSP 2012*, 2012, pp. 5017–5021.