IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 17, 2021, accepted August 31, 2021, date of publication September 22, 2021, date of current version September 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3114371

On-Device System for Device Directed Speech Detection for Improving Human Computer Interaction

ABHISHEK SINGH[1,2], **RITURAJ KABRA**[1], **RAHUL KUMAR**[1], **MANJUNATH BELGOD LOKANATH**[1], **REETIKA GUPTA**[1], **AND SUMIT SHEKHAR**[1]
[1]Samsung Research Institute Bangalore, India, Bengaluru 560037, India
[2]University of California San Diego, La Jolla, CA 92093 USA

Corresponding author: Abhishek Singh (abhishek.s.eee15@iitbhu.ac.in)

This work was supported by the Samsung Research Institute Bangalore, India.

ABSTRACT Voice assistants (VA) are finding a place in many households, with increasing numbers. Nevertheless, for every interaction user invokes VA using a key or wake-up word, which is too common and hinders natural conversation. To solve this, we propose an On-Device solution that listens to the user continuously for **only** predefined period and classifies the utterance into device-directed or non-device-directed using a deep learning-based model. Since our solution is On-Device, the privacy of the user is maintained. We tried to solve false acceptance of utterance as a command and incorrect rejection of the command as background noise to improve user experience. We calculated the False Acceptance Rate (FAR) and False Rejection Rate (FRR) for different thresholds to evaluate our model. Then we calculated Equal Error Rate (EER) results with the FRR and FAR data. We achieved 3.6% of EER on our best optimized **Stage-1** model and EER of 5.1% on **Stage-2** model. We have also used accuracy as evaluation metrics during training and testing. We achieved an accuracy of 96.8% on testing data with our **Stage-1** model and 95.3% on **Stage-2** model.

INDEX TERMS Human–computer interaction, natural conversation, speech classification, voice-assistants.

I. INTRODUCTION

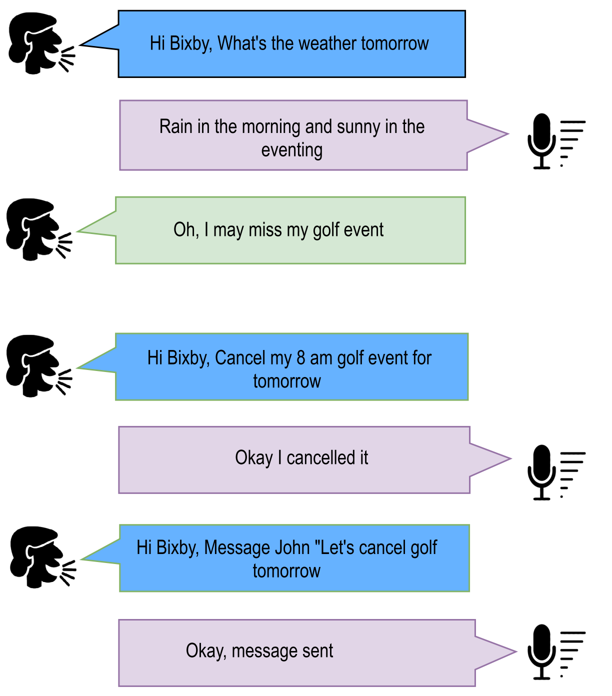
In recent years, voice assistants (VA) such as Bixby, Alexa, and Google-Home are becoming popular in smart home/phone environments. Current smartphones/devices are still dependent on the wake-up keyword for every utterance (or some external trigger), which hinders the natural way of interacting with devices. Hence we propose an on-device solution that listens to the user continuously for a predefined period and classifies the utterance as a device directed) or non-device directed.

We selected most frequently used domains in mobile phones, such as navigation, call, messaging, etc., to create the dataset for device directed through which we are able to distinguish from background speech. In this work, we have used both text and speech features to train a robust classifier to deal with false acceptance and false rejection. Our framework is based on two stages **Stage-1** and **Stage-2** which deals with root command and follow up command. We trained in two stages as root command does not require any context for classification, whereas follow-up utterances do need context. This two-stage framework is different from all the previous work done in this area. Computing on the cloud provides readily available and more resources, but at the same time, it gets more vulnerable as it provides more opportunities to hackers [1]. If the personal data of the user is breached, hackers will have access to valuable and sensitive information, which is the most common concern of the user [2]. As mentioned in [3], cloud-based personal assistants pose a challenge of security threats. We have optimized our framework so that it is deployed on-device; therefore user's utterance is processed on the device itself instead of sending it to the server. Our on-device solution contributes towards saving the privacy of the user.

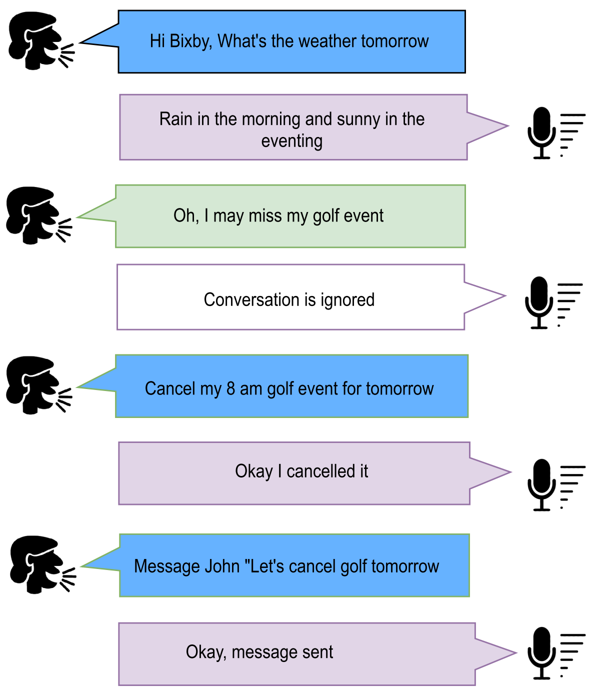
Throughout this paper, we have used terms device-directed utterance and non-device-directed utterance, which we define as:

Device-directed: A word or a sentence uttered by an interlocutor that is intended to be processed by computer(s) (Mobiles, home-assistant, robots).

The associate editor coordinating the review of this manuscript and approving it for publication was Giuseppe Desolda.


Hi Bixby, What's the weather tomorrow
Rain in the morning and sunny in the eventing
Oh, I may miss my golf event
Hi Bixby, Cancel my 8 am golf event for tomorrow
Okay I cancelled it
Hi Bixby, Message John "Let's cancel golf tomorrow
Okay, message sent

(a) Present System, wake-up is needed for every conversation.


Hi Bixby, What's the weather tomorrow
Rain in the morning and sunny in the eventing
Oh, I may miss my golf event
Conversation is ignored
Cancel my 8 am golf event for tomorrow
Okay I cancelled it
Message John "Let's cancel golf tomorrow
Okay, message sent

(b) Proposed system for seamless conversation.

FIGURE 1. **Present and proposed scenarios.**

Non-device directed: A word or a sentence uttered by an interlocutor that is **not** intended to be processed by computer(s), it can be a conversation between two people or a person speaking to himself.

We summarize the main contributions of our work in the following points.

- Collecting the anonymized user utterance data and then analyzing to create synthetic data points.
- Build a classifier for root utterances (**Stage-1**) on audio and text features extracted from utterances.
- Propose a context-aware hierarchical model (**Stage-2**) to classify continuation and follow-up utterance as device and non-device directed.
- Optimizing our models to reduce inference time and size of the model by quantizing nodes, merging duplicate nodes, stripping unused nodes, and fixing vocabulary size.

Figure 1 depicts two scenarios, figure 1a represents present system whereas figure 1b represents the proposed system.

II. RELATED WORK

The on-device continuous listening framework for voice assistants has not been extensively explored in the past. Device-directed speech by [4] proposed a classifier model using Acoustic Embedding and GloVe word embedding. These features are fed to LSTM (Long Short Term Memory) [5] based deep learning model. Their model can achieve an EER (Equal Error Rate) of 5.2% on their proprietary dataset. In a study to improve device-directed utterance classification by [6], they present the performance for different dialog types and decoder features. Learning when to listen by [7] proposes using Prosodic features along with lexical features for better detection of utterance; they report EER of 6.72 with their best model. [6], [8] present methods for improving device-directed speech detection and real-time speech command detector for the control room. Improving device directedness using semantic lexical features combined with lightweight acoustic features [9], applied transfer learning and semi-supervised learning to the model to improve the accuracy. Attention mechanism along with CNN (Convolutional Neural Network) followed by RNN (Recurrent Neural Network) on log-Mel filterbank, is explored in [10]. They claim that the attention mechanism always leads to improvement in performance on their datasets.

In [11] presented device-directed speech detection in the presentation domain by combining signal information and behavior features. Device directedness is also explored in multiple human and robots scenario in simulated spacesuits doing planetary geological exploration with the help of 1-2 robots [12]. Extracting domain level features to select a helpful context from noisy conversations and then incorporating this knowledge in understanding computer-directed utterances is proposed [13]. Since there is no publicly available data for this problem, we compare the performance of existing systems to that of our proposed method on our dataset.

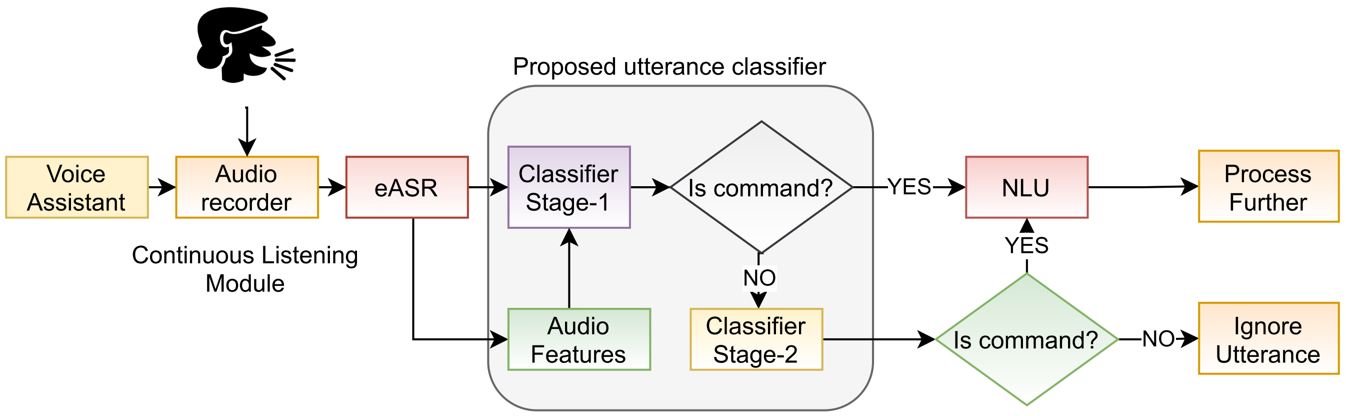


FIGURE 2. Complete system architecture.

III. SYSTEM ARCHITECTURE

In this section we discuss complete system diagram part wise. Complete system architecture diagram is shown in Fig:2.

A. AUDIO PROCESSOR

This module takes care of recording the audio continuously and decides whether to send it for further processing based on voice activity detection. This module also takes care of detecting the endpoint of user utterances so the embedded ASR (Automatic Speech Recognition) module can give the final result [14].

B. EMBEDDED ASR

This module is responsible for the on-device conversion of speech to text. The lightweight ASR model is embedded in the software rather than deployed in the server for faster execution.

C. COMMAND CLASSIFIER

To classify whether the utterance is device directed or non-device directed is supervised binary classification, with the labeled dataset containing data from most frequently used domains such as call, setting, weather, and so on, as a device directed while other than this is non-device directed data. Domains are the features supported by the respective device. We used two models to classify the utterance, the **Stage-1** model is to classify the *Root*[1] command, and the **Stage-2** model is to classify the *Follow-up* and *Continuation command*.[2] **Stage-2** model checks whether this continued utterance entails the previous utterance, which is further processed.

IV. DATASET DESCRIPTION

We have used two different completely anonymized English datasets in our experiments. The **Corpus-1** is only text data of the utterances. This data is comprised of device-directed utterances and non-device-directed utterances. Device-directed utterances are collected from our VA (Voice Assistant) corpus, which is anonymized by the IDs. These types of utterances are collected from fifteen most frequently used domains in voice assistants such as call, weather, settings, gallery, and so on. Whereas non-device directed root utterances are sampled from Cornell Lines [15] and Movie Subtitles.[3] Refer to Table 1:

TABLE 1. Corpus-1 data distribution(text only).

Utterance Type	Device Directed	Non-Device directed
Root Utterance	300K	350K
Contextual/ Follow up	60,911	72,563

The **Corpus-2** is (audio + text). Device directed utterances are collected from our VA corpus; it contains total of 100hrs speech data along with its text transcription. Non-device directed data is collected from Librispeech corpus [16], it is also 100hrs speech data along with text.

Root utterance for device-directed class can be "*What is the weather outside*" and for non-device directed "*Did you have a nice summer*". Current utterance relation with the previous utterance is not considered while classifying it as device or non-device directed. Here non-device-directed utterances are collected from movie subtitles.

Examples of continuation and contextual utterance can be referred from Table 2 Example 1. To classify current utterance as device and non-device-directed, we need to find its relation with the device's current context and previous utterance. Hence, the second model is trained to find a correlation between current utterance and previous utterance for these types of utterances. Table 2 example 2 presents a sample of utterances where the current utterance is not related to the

[1]Intent of these types of utterance can be determined independent of previous utterance or context

[2]Intent of these type utterances can be determined by taking context and information from previous utterance and device action

[3]https://github.com/PolyAI-LDN/conversational-datasets/tree/master/opensubtitles

TABLE 2. Device directed continuation and follow up commands.

Previous Utterance	Current Utterance
What is the weather in London?	How about New York?
Send a message to John, I am coming.	Start slideshow.
YouTube outlaws the game soundtrack	Play that week I have about 20 Witness come out of the dryer

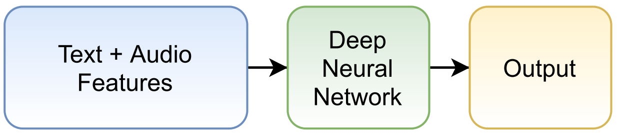


FIGURE 3. Stage-1 - Root Model architectur.

previous utterance. It cannot be considered as device directed in the present context of the device, but it can be regarded as in some other scenario. For example, "Start slideshow" can be considered as device-directed when the user is checking photos in the gallery app. While exploring the data, we found that 9% of utterances do not have a correct grammatical structure, making it challenging to comprehend meaning even to the human evaluator. In Table 2 example 3, we present examples of such types where it is challenging to derive meaning due to grammatical incorrectness and sentence structure. These errors are usually caused due to incorrect ASR results. Hence we cleaned the data before using it by spelling correction, contraction, and acronym expansion to train the models. This cleaning helped to increase performance by 1.9% in accuracy. Sample text data[4] only contains synthesized data maintaining the privacy of every user.

Acronyms and Contractions were replaced with their corresponding English words. We replace it by creating a dictionary of acronyms and contractions mapping to their expanded form. Acronyms such as 4ever are converted to forever, abt to about, cb to comeback, etc. These acronyms are commonly used in social media platforms. Contractions such as can't, aren't, i've, etc., were again converted to their corresponding text cannot, are not, and I have for this case.

We have used PySpellchecker,[5] it uses a Levenshtein Distance algorithm to find permutations within an edit distance of 2 from the original word. Then it compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results.

V. METHODOLOGY

We trained multiple classifier models, **Stage-1** to classify root utterances and **Stage-2** to classify continuation and follow-up utterances. We used audio as well as text features to classify root utterances. The reason to consider audio features is that we do not need to train separately for different domains, and ASR performance does not become a bottleneck.

A. FEATURE EXTRACTION AND PRE-PROCESSING

Text features: The input text sequence is capped to length 15 for root classifier and contextual/follow-up classifier. Raw text is pre-processed by first expanding contractions and acronyms, removing punctuation marks, etc. This text is then tokenized to the word level, and the embedding matrix is created using these words. **Audio features:** The audio is processed to extract the following prosodic features consisting of log-Mel-Spectrogram, MFCC (Mel-Frequency Cepstral Coefficients) [17], DCT (Discrete Cosine Transform) [18] bases to capture energy-related features, duration of speech, and silences to summarize rate of speech and features from ASR pipelines like likelihood over a frame, average likelihood over all the frames, list of phonemes and noise duration, and the logarithm of STFT (Short-Term Fourier Transform).

STFT provides the time-localized frequency information for situations in which frequency components of the signal vary over time. In contrast, the standard Fourier transform provides information averaged over the entire signal time interval [19], [20].

We also compute dynamic MFCC features, since static feature only contains information from the current domain. The extra information about the temporal dynamics of the signal is obtained by computing first and second derivatives of cepstral coefficients [20]–[22]. The first-order derivative is called delta coefficients, and second-order derivatives are called double delta coefficients. Delta coefficients tell us about speech rate, whereas double delta coefficients provide information similar to the acceleration of speech.

VI. DEEP LEARNING MODELS

Our **Stage-1** network has LSTM layers, dropout layer (dropout rate of 0.2), and this output is fed to self-attention [23] layer, and finally, we have a dense layer with sigmoid activation. Batch normalization is performed before dropout.

In the case of root model shown in Figure 4 (**Stage-1**), if the result is non-device directed, we pass the current utterance through the contextual/follow-up **Stage-2** model shown in (Figure 5) along with the features of the last context determined by the VA. We are using the 2nd stage model for follow-up and continuation commands.

Model for root-based commands – All the audio features MFCC, log-STFT, and DCT are passed through the LSTM layer separately. The final stage outputs of these features are concatenated with a ratio of speech in complete audio features and text features.

We have fixed the maximum length of the utterance to 15, as we found out that 98% of the utterances that are directed towards the device are equal to or below 15. The utterance is tokenized and then passed to the model. In this model, the single input of the current utterance is passed through

[4]Please contact the author for supplementary material
[5]https://github.com/barrust/pyspellchecker

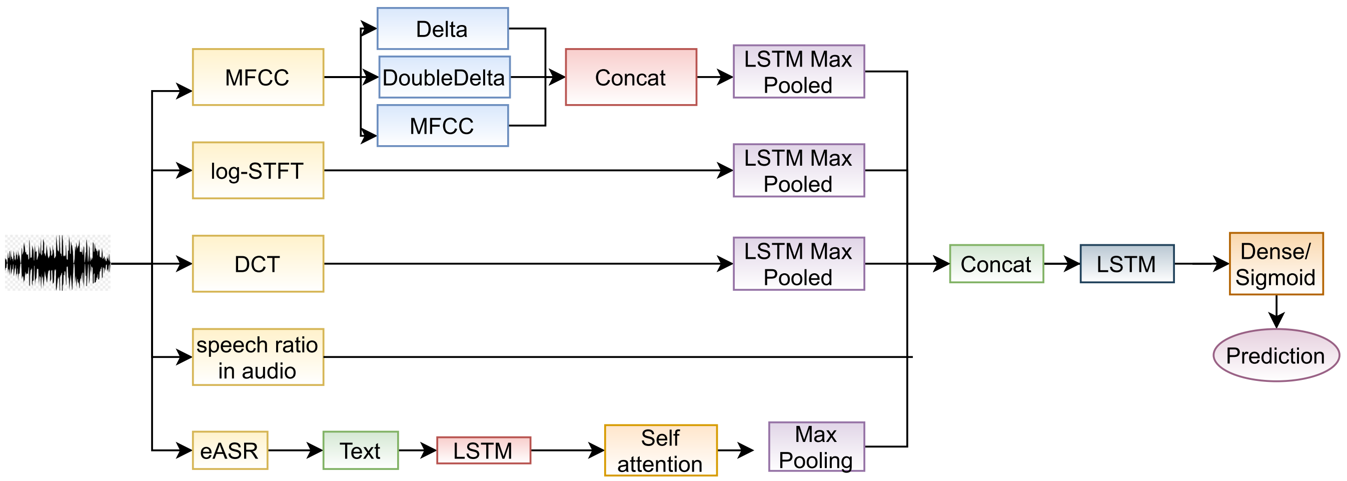


FIGURE 4. Architecture of root (Stage-1) model.

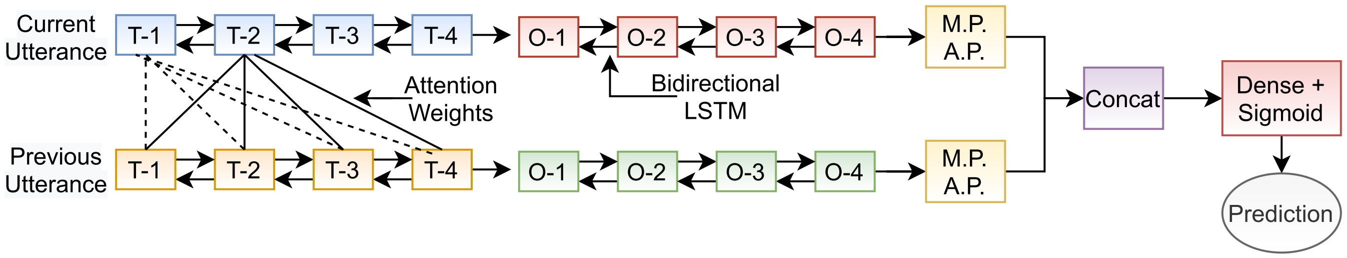


FIGURE 5. High level architecture diagram of contextual model. Here T-1, T-2, T-N represent tokens of current and previous utterance and O-1, O-2, O-N represents m_c and m_p of equation 4,5. M.P. and A.P. represent max-pooling and average pooling respectively.

the LSTM layer, and output is passed through self-attention. Self-attention allows each word to pay attention to other words in the sequence, independent of their position. The output of self-attention is max-pooled and concatenated with LSTM outputs of audio features and ratio of speech in complete audio, which is then passed through dense layers and then to sigmoid activation to determine the utterance as a device and non-device directed.

Model for contextual and follow-up commands – Follow-up and contextual commands depends upon their relationship with the previous utterance. Hence, in this model, we consider two inputs - current and previous utterances. We try to draw the entailment relation between current and previous utterances using the attention mechanism between these inputs. This model is inspired by Enhanced LSTM for Natural Language Inference [24]. Both the inputs are passed to BiLSTM [25] layers to represent words using both the left and right context. To compute inter-sentence alignment between current and previous utterances, we compute soft-alignments between hidden state outputs of each time step of both current and previous utterances. Consider, and, as hidden(output states) generated by BiLSTM at each time steps of current and previous utterances. Soft alignment attention weights is calculated as:

$$e_{ij} = \bar{c}_i^T \bar{p}_j \tag{1}$$

For the hidden state of word in current utterance i.e. $\bar{c}$ (encoding the word itself and intra-sentence context), relevant relation from previous utterance is computed using weight vector e_{ij} as:

$$\tilde{c}_\iota = \sum_{j=1}^{l_b} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{l_b} e_{ik}} \bar{p}_j, \quad \forall i \in [1, \ldots\ldots, l_a] \tag{2}$$

$$\tilde{p}_j = \sum_{i=1}^{l_a} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{l_a} e_{ik}} \bar{c}_\iota, \quad \forall j \in [1, \ldots\ldots, l_b] \tag{3}$$

Here $\tilde{c}_\iota$ is the weighted summation of $\{p\}_{j=1}^{l_b}$. With this context vector, we are able to derive information from the previous utterance that is relevant to each word in the current utterance. A similar thought can be put for the previous utterance context vector in Equation 3. Hence, this way, for each word in the current utterance, we are able to gather its relation with each word in the previous utterance to get the correlation, and its relation with other words from the current utterance can be collected using the BiLSTM layer.

Enhancement of Inference model: In our inference model for follow up and continuation utterances we enhance the model by taking element-wise difference and product for tuples $< \bar{c}, \tilde{c} >$, and $< \bar{p}, \tilde{p} >$. It is expected that from difference and multiplication, we can gather important features

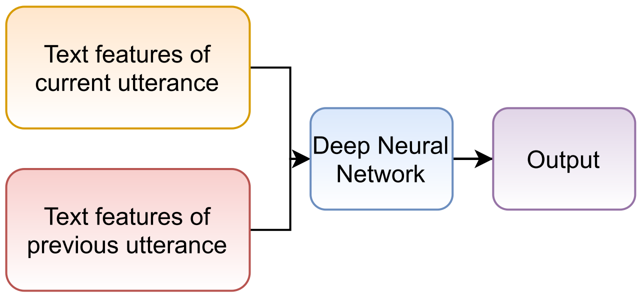

Text features of current utterance
Text features of previous utterance
Deep Neural Network
Output

FIGURE 6. **Stage-2 - Contextual model architecture.**

like entailment and contradiction between two sentences (in our case, relation between current and previous utterance). All these features are then concatenated as shown in equation (4, 5) and passed through a feed-forward network with sigmoid activation in the last layer to determine whether the command is device or non-device directed.

$$m_c = [\bar{c} : \tilde{c} : \bar{c} - \tilde{c} : \bar{c} \odot \tilde{c}] \tag{4}$$

$$m_p = [\bar{p} : \tilde{p} : \bar{p} - \tilde{p} : \bar{p} \odot \tilde{p}] \tag{5}$$

A. OPTIMIZING THE MODELS

Though our solution is potentially useful, users may feel insecure about their privacy since the device listens to the user continuously for a short duration [26]. Users have further concerns about whether this data is stored and who has the opportunity to review it.

While traditional computation paradigms rely on mobile sensing and cloud computing, deep learning implemented on mobile devices provides several advantages. These advantages are four-fold, which include low communication bandwidth, small cloud computing resource cost, quick response time, and improved data privacy [27].

Our raw model size without any optimizations for the root model is 98.5MBs, and for the contextual model is 196MBsmbs. Deploying these large-size models is not compatible with devices with less RAM and computation power. To deploy these models on devices such as smart speakers and mobile phones, it is necessary to reduce model size. Initially, we reduced and fixed the maximum number of words in the vocabulary for embedding matrix to 25,000 most frequently occurring words. We came up with this number by running our experiments on different vocabulary sizes. This size resulted in the best tradeoff between the accuracy and size of the model. With this step, we reduced our model sizes to one-third of the original ones. We reduced the sizes of embedding vectors to reduce the size of the model further. Reducing vector sizes did not bring down the accuracy of the models.

We applied multiple optimization techniques[6] to shrink the overall size of the model and improve prediction latency. These techniques include Freezing, Pruning, Constant folding, Folding batch norms, and Quantization. Freezing helps to convert variables to constants in the model graph. Pruning removes unused nodes and merges duplicate nodes. Constant folding looks for sub-graphs in the model that always return constant value and replace it with that constant. Folding batch norms merges the multiplication to weights of the previous layer. Quantization reduces floating-point of weights to lower precision, such as 16 or 8 bits. With these steps, our model sizes are reduced from 98.5MBs and 196MBs to 11.3MBs and 8.28MBs.

VII. FEATURE PARAMETERS AND MODEL HYPER-PARAMETER TUNING

Text feature parameters- The maximum number of words in the embedding matrix is capped to 25000. The embedding vector dimension in the model is 100. The common tokenizer is used for the root and contextual model. Punctuations and stop words are not removed while processing the text, as they provide meaning to utterances in our case. The maximum length of utterance in root and contextual models in kept as 15 tokens.

Audio feature parameters- We extract 39 MFCC coefficients (including delta and delta-delta) with a speech rate of 16 kHz. To extract the log-STFT feature, we use a window length of 30ms, hop length of 15ms. We compute DCT (type-1) as mentioned in SciPy [28]–[31] by taking frame size of 200 and hop size of 100, and speech rate of 16 kHz. We determined the voice activity of the utterance using WebRTC-VAD.[7] We applied VAD (Voice Activity Detection) with the aggressiveness of 2 on the audio to create a vector of speech and non-speech in the audio.

Dataset after preprocessing is split into three parts, training (72%), dev (8%), and test (20%). We use pre-trained Glove embedding [32] to initialize our embedding matrix; these embeddings are fine-tuned while training the model. Our models are trained with Binary Focal loss function [33] (a variant of binary cross-entropy) as it is well suited for imbalanced datasets. Focal loss is defined as:

$$FL(p_t) = -(1 - p_t)^{(\gamma)} log(p_t), \tag{6}$$

where γ is tunable focusing parameter.

We have used Adam [34] optimizer with a learning rate of 0.004, early stopping, and reducing the learning rate on plateau callbacks. Models are trained for 10 epochs and are validated after every epoch with the validation data as 10% of data after removing the test set. A batch size of 32 was used during training. Early stopping with a delta of 0.0001 and patience of 2 was applied on validation loss for a call back during training.

[6]https://github.com/tensorflow/model-optimization

[7]https://github.com/wiseman/py-webrtcvad

TABLE 3. EER and accuracy results on root model(Stage-1).

	EER	Accuracy	Size
Norouzian et al.	4.32	0.951	78.3MBs
Shriberg et al	6.91	0.93	56.3MBs
Text	3.2	0.97	98.5MBs
Audio	5.6	0.93	56.2MBs
Audio+Text (Proposed)	2.9	0.975	134.6MBs
Size Optimized (Proposed)	3.6	0.968	11.3MBs

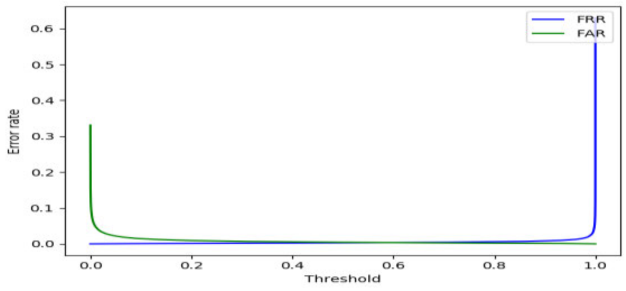


FIGURE 7. EER calculation of size optimized Root model.

VIII. EXPERIMENTS AND RESULTS

In this section, we discuss results and experiments in two parts. First part presents results on **Corpus-1** and second part presents results on **Corpus-2**.

A. EXPERIMENTS AND RESULTS ON ROOT MODEL STAGE-1

We performed experiments on **Stage-1** model using both audio and text features. Combination of both audio and text features performed better than when trained on individual features due to the fact that the model is able to understand utterance semantically, and audio features help to improve confidence score. There is a slight decrease in performance in the optimized model, but the size is drastically reduced, which is useful to port on the device. Inference time is reduced from 252ms to 102ms on optimized model when tested on mobile device. Table 3 presents evalution on **Corpus-1** using different methods.

B. EXPERIMENTS AND RESULTS ON CONTEXTUAL/ FOLLOW-UP MODEL (STAGE-2)

We performed experiments on **Stage-2** model using only text features for context as audio features are considered during **Stage-1** classification. For **Stage-2** model performance is intriguing since the model needs to understand the relation between the current utterance and the previous utterance. The size of the optimized model is reduced significantly without compromising much on the performance, which is suitable for on-device deployment. The contextual model gave an EER of 4.6, an accuracy of 97.1, and a size of 196MBs, and the optimized size model gave an EER of 5.1, the accuracy of 95.3, and the size drastically reduced to 8.2MBs.

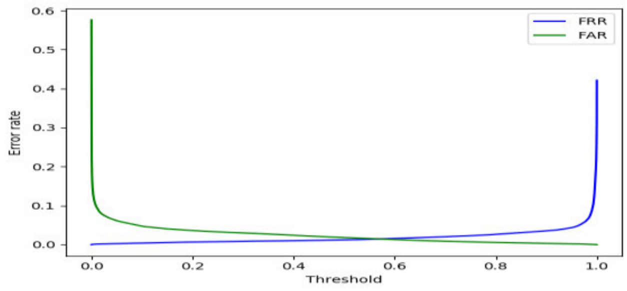


FIGURE 8. EER calculation of size optimized context model.

IX. CONCLUSION

This paper proposed a novel two-stage framework to classify an utterance as a device-directed and non-device-directed utterance for root and follow-up utterance. This architecture helped us to improve the performance of the framework by reducing false acceptance and false rejection rate of the utterance. This framework will help to improve the user experience when they interact with virtual assistants by making the conversation more natural; as the user does not require to explicitly invoke virtual assistant using a wake-up word or pressing a key, therefore making the conversation fluent and natural as depicted in figure 1b.

In our experiments, we have explored data **Corpus 1, 2** and performed separate experiments on both of them. The combination of audio and textual features improved the model's EER and accuracy performance. We achieved an EER of 3.6 on the size-optimized model and 2.9 without size optimization. We found from experiments that text alone model required a large corpus to generalize well in our case. Training and evaluating only on audio data has a drawback that when it is tested on the manually recorded dataset from different distribution, its performance went down by 5.6% EER. Hence to overcome these problems, we combined both text and audio models for **Stage-1** classifier. A combination of models improved the model's EER and accuracy performance. We have trained audio model on different combinations of features- log-MFCC (with delta and delta-delta features), ratio of speech in complete audio, DCT, and log-STFT.

We trained **Stage-1** (contextual/follow-up) model on text features only. We achieved an EER of 5.1% on the optimized contextual model. To classify current utterance as a device or non-device directed, its entailment with the previous utterance is computed using an attention mechanism as explained in the above sections. In this model, both current and previous utterance (text) is passed in the model. BiLSTM is used to get intra-sentence context, and an attention mechanism is used to get inter-sentence relation for each word in both the inputs.

Further improvements in the system can be made by collecting more robust data from different distributions. The contextual model needs further improvement since most of the utterances are similar to general conversation.

REFERENCES

[1] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *J. Supercomput.*, vol. 63, pp. 561–592, Oct. 2013.

[2] M. Tabassum, T. Kosinski, and H. R. Lipford, "'I don't own the data': End user perceptions of smart home device data practices and risks," in *Proc. 15th Symp. Usable Privacy Secur. (SOUPS)*, 2019, pp. 435–450.

[3] J. S. Edu, J. M. Such, and G. Suarez-Tangil, "Smart home personal assistants: A security and privacy review," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–36, 2020.

[4] S. Harish Mallidi, R. Maas, K. Goehner, A. Rastrow, S. Matsoukas, and B. Hoffmeister, "Device-directed utterance detection," 2018, *arXiv:1808.02504*. [Online]. Available: http://arxiv.org/abs/1808.02504

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] C.-W. Huang, R. Maas, S. H. Mallidi, and B. Hoffmeister, "A study for improving device-directed speech detection toward frictionless human-machine interaction," in *Proc. Interspeech*, Sep. 2019, pp. 3342–3346.

[7] E. Shriberg, A. Stolcke, D. Hakkani-Tür, and L. Heck, "Learning when to listen: Detecting system-addressed speech in human-human-computer dialog," in *Proc. Interspeech*. International Speech Communication Association, Sep. 2012, pp. 334–337. [Online]. Available: https://www.microsoft.com/en-us/research/publication/learning-when-to-listen-detecting-system-addressed-speech-in-human-human-computer-dialog/

[8] D. Reich, F. Putze, D. Heger, J. Ijsselmuiden, R. Stiefelhagen, and T. Schultz, "A real-time speech command detector for a smart control room," in *Proc. 12th Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*. Florence, Italy: International Speech Communication Association, Aug. 2011, pp. 2641–2644.

[9] K. Gillespie, I. C. Konstantakopoulos, X. Guo, V. T. Vasudevan, and A. Sethy, "Improving device directedness classification of utterances with semantic lexical features," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 7859–7863.

[10] A. Norouzian, B. Mazoure, D. Connolly, and D. Willett, "Exploring attention mechanism for acoustic-based classification of speech utterances into system-directed and Non-system-directed," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 7310–7314.

[11] T. Paek, E. Horvitz, and E. Ringger, "Continuous listening for unconstrained spoken dialog," in *Proc. 6th Int. Conf. Spoken Lang. Process. (ICSLP)*, Beijing, China, Nov. 2000, pp. 138–141. [Online]. Available: https://www.microsoft.com/en-us/research/publication/continuous-listening-unconstrained-spoken-dialog/

[12] J. Dowding, R. Alena, W. J. Clancey, M. Sierhuis, and J. Graham, "Are you talking to me? Dialogue systems supporting mixed teams of humans and robots," in *Proc. AAAI Fall Symp.: Aurally Informed Perform.*, 2006, pp. 22–27.

[13] D. Wang, D. Hakkani-Tur, and G. Tur, "Understanding computer-directed utterances in multi-user dialog systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8377–8381.

[14] R. Maas, A. Rastrow, K. Goehner, G. Tiwari, S. Joseph, and B. Hoffmeister, "Domain-specific utterance end-point detection for speech recognition," in *Proc. Interspeech*, Aug. 2017, pp. 1943–1947.

[15] C. Danescu-Niculescu-Mizil and L. Lee, "Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs," 2011, *arXiv:1106.3077*. [Online]. Available: http://arxiv.org/abs/1106.3077

[16] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[17] J. W. Picone, "Signal modeling techniques in speech recognition," *Proc. IEEE*, vol. 81, no. 9, pp. 1215–1247, Sep. 1993.

[18] N. Ahmed, "How I came up with the discrete cosine transform," *Digit. Signal Process.*, vol. 1, no. 1, pp. 4–5, Jan. 1991.

[19] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[20] J. B. Allen and L. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, Nov. 1977.

[21] S. Furui, "Comparison of speaker recognition methods using statistical features and dynamic features," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 3, pp. 342–350, Jun. 1981.

[22] J. S. Mason and X. Zhang, "Velocity and acceleration features in speaker recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 1991, pp. 3673–3674.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[24] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced LSTM for natural language inference," 2016, *arXiv:1609.06038*. [Online]. Available: http://arxiv.org/abs/1609.06038

[25] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[26] N. Malkin, S. Egelman, and D. Wagner, "Privacy controls for always-listening devices," in *Proc. New Secur. Paradigms Workshop*, Sep. 2019, pp. 78–91.

[27] Y. Deng, "Deep learning on mobile devices: A review," *Proc. SPIE*, vol. 10993, May 2019, Art. no. 109930A.

[28] K. J. Millman and M. Aivazis, "Python for scientists and engineers," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 9–12, Mar./Apr. 2011.

[29] F. Chollet, "Keras: The Python deep learning library," in *Proc. ASCL*, 2018, p. 1806.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[31] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop: Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.

[32] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

ABHISHEK SINGH was born in Satna, Madhya Pradesh, India, in 1997. He received the B.Tech. degree in electrical engineering from Indian Institute of Technology (BHU), Varanasi, in 2019. He is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. His research interests include machine learning and natural language processing.

RITURAJ KABRA was born in Dhansura, Gujarat, India, in 1989. He received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology (NIT), Silchar, in 2012. He is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. His research interests include machine learning and natural language processing.

RAHUL KUMAR was born in Madhubani, Bihar, India, in 1998. He received the B.Tech. degree in electronics and communication engineering from Indian Institute of Technology (IIT), Guwahati, in 2019. He is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. His research interests include machine learning and natural language processing.

MANJUNATH BELGOD LOKANATH was born in Davanagere, Karnataka, India, in 1981. He received the B.E. degree in computer science from Visvesvaraya Technological University, in 2003. He is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. His research interests include machine learning, AI, and natural language processing.

REETIKA GUPTA was born in Jhansi, Uttar Pradesh, India, in 1991. She received the M.C.A. degree from Birla Institute of Technology (BIT), Mesra, in 2014. She is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. Her research interest includes natural language processing.

SUMIT SHEKHAR was born in Uttar Pradesh, India, in 1993. He received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology (NIT), Silchar, in 2015. He is currently working as a Research Engineer with the Voice Intelligence Team, Samsung Research Institute Bangalore, India. His research interests include machine learning and natural language processing.