November 2023

In-Conference Tool Virtual Assistant with Real Time Speaking Latency

Dongeek Shin

In-Conference Tool Virtual Assistant with Real Time Speaking Latency

Speech-based interaction models in virtual assistants can enable users to verbally interact with their devices in real time. A user can provide a verbal query to the user device, which the speech-based interaction model processes and responds to in audio or text format. To process the audio input of a user in real time, some conventional speech-based interaction models utilize a transcription engine and a large language model (LLM). The transcription engine transcribes the audio input of the user and sends the transcribed query to the LLM, which analyzes the transcription and generates a response.

In some conventional systems, the transcription engine includes a buffer that serves as a temporary storage area for the audio data before it is processed and transcribed. Once the buffer reaches a specific size (e.g., the size allocated to the buffer) or once a specific time interval has passed, the transcription engine transcribes the audio data in the buffer. However, waiting until the buffer reaches a specific size or until a specific time interval expires, before processing the audio data, can introduce latency in real-time speech-based models because the user can finish speaking before the buffer reaches the specific size or the specific time interval expires. The latency caused during this period is often the bottleneck in real time speech-based models. While reducing the specific size for the buffer or the specific time period might reduce latency, it can also negatively impact the quality of transcriptions due to a lack of context if the buffer is filled before the user has finished speaking. Accordingly, these conventional methods do not solve the concern of creating high quality transcriptions with minimal latency.

The response time of real time speech-based models can be calculated using the following equation:

t_i(response start) – t_i(query end),

where t_i(response start) represents the time when the speech-based interaction model begins responding to the user's query, and

t_i(query end) represents the time when the user finishes speaking.

However, as described above, in most conventional systems the response time is highly dependent on the transcription engine because it plays the largest role in determining when a query ends and when the resulting transcription can be sent to the LLM for processing. The lag in response time can break the realism of conversations with virtual assistants for users.

To address the above and other shortcomings, a framework is proposed that includes a hybrid scheme utilizing a combination of a LLM and a small sound model (SSM). The SSM can be used to timestamp the end of a user's verbal query, which can then be used to notify the transcription engine that the query has ended. The transcription engine can then abort and send the existing set of transcriptions to the LLM for processing without waiting for the buffer to reach the specific size or for the specific time interval to expire. As a result, the latency caused by waiting for the buffer to fill or for the specific time interval to expire is reduced. The above framework can be integrated in virtual assistants and other real time speaking applications or devices, such as language translation devices, accessibility devices, voice controlled smart devices, and other voice assistive technologies.

Figure 1 illustrates a data flow diagram of a method 100 for processing a user's audio input 110 to send to a LLM 140. When a user is speaking, the audio input 110 can be received and provided as input to both a transcription engine 120 and SSM 130 concurrently. The SSM 130 can be associated with a mechanism for communicating with the transcription engine 120. The SSM 130 can determines when a verbal query has ended and generate a message or a flag that is sent to the transcription engine 120 to immediately abort and use the existing set of

transcriptions to provide as input to the LLM 140. The LLM 140 then processes the transcribed query and generates a response.

In one example, the SSM 130 can be a direct speech-based model trained to find patterns in a user's tone that indicate the end of a verbal query. The SSM 130 can be trained to output a segmentation score for received audio input using raw sound data. For instance, the SSM 130 can take a user's raw speech as input and output a query segmentation vector. When an output segmentation score exceeds a predefined threshold, the SSM 130 can determine that the query has ended. For example, the SSM 130 can determine that a verbal query has meaningfully ended within milliseconds of when the user stops speaking.

In an illustrative example, the raw speech input fed into the SSM 130 can be the phrase "how is the weather today?" The SSM 130 can output a query segmentation vector for the raw speech input and analyze the segmentation scores of the query. The segmentation score for the end of the word "today" may exceed the predefined threshold score for the end of the query. As a result, the SSM 130 can determine that the user's query has ended after the word "today" and generate a message or a flag, that can be used to notify the transcription engine 120 to abort and to provide the transcription to the LLM 140 for processing.

The SSM 130 can be composed of, e.g., a single level of linear or non-linear operations (e.g., a support vector machine (SVM)) or a deep network, such as a machine learning model that is composed of multiple levels of non-linear operations. An example of a deep network is a neural network with one or more hidden layers, and such a machine learning model may be trained by, for example, adjusting weights of a neural network in accordance with a back propagation learning algorithm or the like. In some instances, the SSM 130 can be composed of convolutional networks. In other instances, the SSM 130 can be composed of transformer

decoders, similar to the architecture of an LLM. Once the SSM 130 is trained, it can be used to analyze the segmentation scores of a user query to determine when the query has ended.

The transcription engine 120 that is provided with the raw audio input 110 in parallel with the SSM 130 can convert the user's raw speech input into written text. Much like the SSM 130, the transcription engine 120 can be composed of, e.g., a single level of linear or non-linear operations (e.g., a support vector machine (SVM)) or a deep network, such as a machine learning model that is composed of multiple levels of non-linear operations. In some instances, the transcription engine 120 can be trained using datasets of paired audio and transcriptions. The transcription engine 120 can include an acoustic model that uses the audio input to map acoustic features into linguistic units and a language model that estimates the probability of word sequences in a language. A decoder can be used in the transcription engine 120 to take the output of the two models and search for the word sequence that best fits the audio input with the goal of reducing the difference between the predicted transcription and the actual transcription. Once the transcription engine 120 is trained, it can be used to process raw user audio input and generate a transcription of the audio input, which is then provided as input to the LLM 140.

The LLM 140 receives the transcription of the user query and generates a response to the user query. In some instances, the LLM 140 can be composed of, e.g., a single level of linear or non-linear operations (e.g., a support vector machine (SVM)) or a deep network, such as a machine learning model that is composed of multiple levels of non-linear operations. In other instances, the LLM 140 can use a transformer-based model architecture with a self-attention mechanism. The LLM 140 can comprise an artificial neural network, composed of artificial neurons or nodes connected by weights. A positive weight reflects a relevant connection, while a

negative weight reflects irrelevant connections. Through training, the LLM 140 can adjust the weights to minimize the difference between predicted and desired outputs.

The LLM 140 can be trained using datasets of question-answer pairs. The LLM 140 can learn the probabilities of question-answer pairs by using self-supervised and/or supervised learning to predict answers for input questions with the goal of reducing the difference between predicted answers and actual answers. Once the model is trained, it can be used to process transcriptions of user audio input 110 and generate a response to the user's query.

Figure 2 illustrates a diagram of a comparison 200 between a timeline of a user's verbal query 210 and a timeline of the buffer space used by the transcription engine 220 to transcribe the query. The amount of space needed to store the length of the user query 230 is equivalent to 2.5 buffers 240a-c in the transcription engine 120. The third buffer 240c is shorter than the other two buffers 240a-b because the transcription engine 120 had received a notification that the query had ended based on a message or a flag generated by the SSM 130, which causes the transcription engine 120 to abort and provide the transcription to the LLM 140 for processing without waiting for the third buffer 240c to reach the specific size or for the specific time interval to expire.

Figure 2 visually illustrates that the proposed framework reduces latency due to the SSM 130 determining when a user query has ended and generating a message or a flag, which is used to notify the transcription engine 120. In conventional systems, the transcription engine 120 would not receive a notification to abort and would not send the transcription for processing until the third buffer 240c has reached a specified size (is completely filled). Accordingly, the latency in the conventional system would be significantly greater than with the disclosed framework due to the time it would take to completely fill the third buffer 240c.

By using a hybrid scheme between LLMs and SSMs rather than determining sentence semantics with a transcription engine, the disclosed technique can generate more accurate query end times and reduce the net response time in real time speech-based models.

Abstract

A framework is proposed for reducing latency for real-time speaking applications, such as virtual assistants. The framework utilizes a hybrid scheme between a large language model (LLM) and a small sound model (SSM). The SSM timestamps the end of a user's verbal query and notifies the transcription engine that the query has ended. The transcription engine will then abort and send the existing set of transcriptions to the LLM for processing without waiting for the buffer to meet the specific value or time. This minimizes the net response time for real time speech-based models due to the reduced latency from the transcription engine.

Keywords: virtual meeting, video conference, virtual agent, virtual assistant, large language model, small sound model, speaking latency

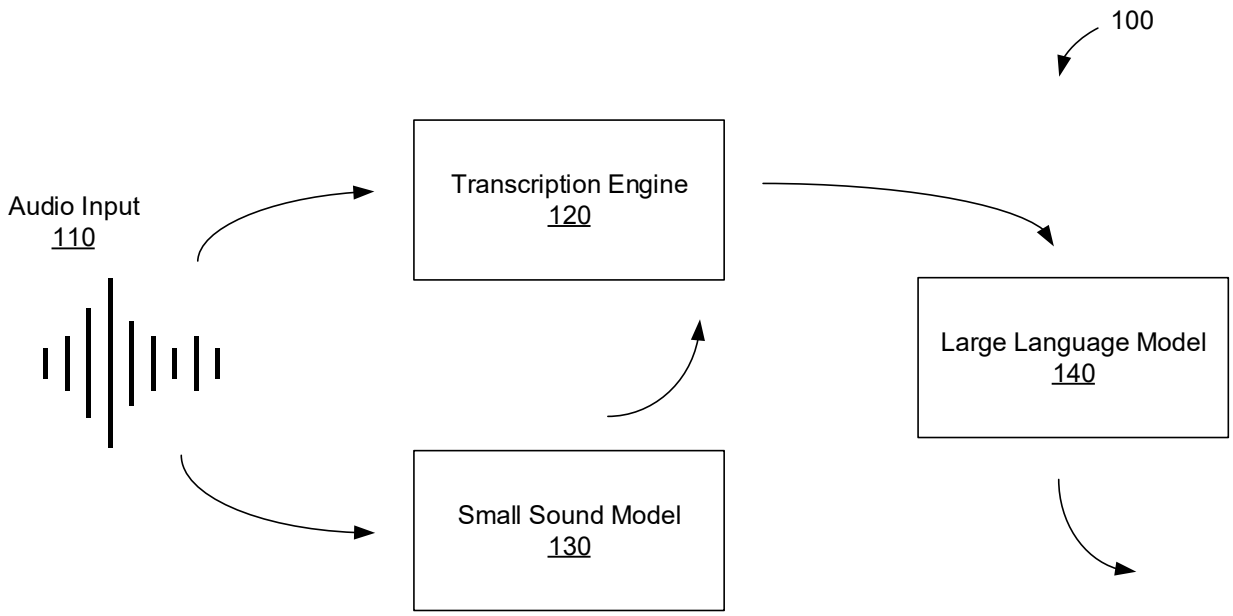

Figure 1

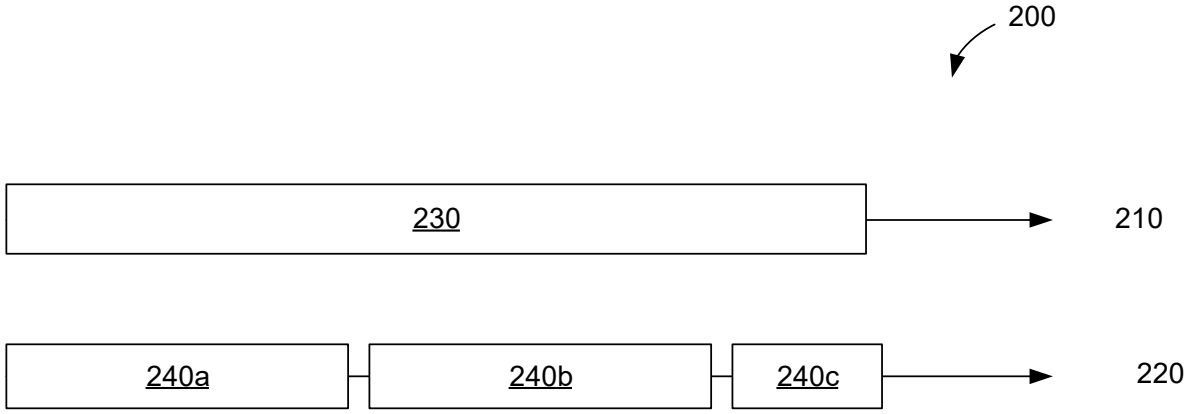

Figure 2