Avestia Publishing
Journal of Machine Intelligence and Data Science (JMIDS)
Volume 5, Year 2024
ISSN: 2564-3282
DOI: 10.11159/jmids.2024.017

Response Delay Reduction in Large Language Model-Based Dialogue Systems

Hikaru Kamioka[1], Satoshi Maeda[1], Masayuki Hashimoto[1]
[1]Toyo University, Department of Electrical, Electronic and Communications Engineering
/ 2100 Kujirai, Kawagoe-shi, Saitama, Japan 350-8585
{s16C02100470, maeda518, hashimoto065} @ toyo.jp

Abstract - *In typical LLM-based voice dialogue systems, the system waits for the user to finish speaking before processing and generating a response, causing a delay. We propose an approach where the LLM generates response text for an incomplete utterance without waiting for the user to finish speaking. This method allows for earlier initiation of response generation, potentially reducing delays. However, LLMs may not always produce appropriate responses when generating replies to incomplete utterances, despite their predictive capabilities. If voice data acquisition is terminated too early, the likelihood of inappropriate responses from the LLM increases. On the other hand, delaying the termination reduces the effectiveness of latency reduction. Therefore, it is crucial to identify the optimal timing for terminating voice data acquisition.*

To determine the optimal timing for stopping audio capture and initiating response generation, we use changes in Sentence-BERT embedding representations of the dialogue history up to that point. We investigate changes in the similarity measure, $S_t(X_0, X_t)$, between the embedding representation X_0 (the embedding at the start of the user's utterance) and X_t (the embedding at midpoint t during the user's utterance).

It is suggested that the changes in similarity may be correlated with the validity of the LLM's responses. Therefore, we proposed some methods to determine the cutoff point based on these changes and conducted simulation experiments to evaluate their effectiveness specifically in Japanese conversations. As a result, we demonstrated that our proposed method can successfully stop audio capture 14.6 characters before the end of the user's utterance, achieving a 0.80 probability of generating a valid response. This reduction of 14.6 characters corresponds to approximately 2.4 seconds in Japanese speech.

Keywords**:** Response delay, Dialogue systems, Large Language Models.

1. Introduction

Voice dialogue systems [1] have been put into practical use and are utilized in various services. The systems can be classified into task-oriented dialogue systems, which are designed to accomplish specific tasks, and non-task-oriented systems, which do not aim to accomplish a particular task. For example, the former is used for purposes such as asking about tomorrow's weather and purchasing something on online sites. On the other hand, the latter is a chit-chat system which engages in conversations without a specific goal. Chit-chat systems are not only for enjoying casual conversations but are also expected to have psychological benefits, such as maintaining or improving mental health and reducing feelings of loneliness through conversation.

In these voice dialogue systems, it is common for a time-lag to occur between the user's utterance and the system's response. In task-oriented systems, users unconsciously tolerate some delay in responses because they generally hope to obtain information from the system or convey information to it. However, in chit-chat systems, response delay becomes a problem [2]. In Japanese conversations between humans, it is common for the next speaker to start speaking within one second after the previous speaker finishes. Therefore, if the purpose of the system is casual conversations like those between humans, unless its response occurs in a short time, subjective satisfaction will deteriorate significantly. Thus, this study focuses on chit-chat systems and attempts to reduce its response time.

Date Received: 2024-03-28
Date Revised: 2024-09-22
Date Accepted: 2024-10-19
Date Published: 2024-12-10

In the following section, we explain the mechanism by which response delay occurs in a voice dialogue system. Then, we introduce our approach for reducing the delay and explain the issues that arise in this approach in Section 4. In Section 5, we present our proposed method to solve these issues. In Section 6, we evaluate the performance of the proposed method using Japanese conversation corpora and present the optimal parameters of the method.

2. Response Delay in Dialogue Systems

Figure 1 illustrates a process flow of a typical voice dialogue system. The user's utterance is captured as an audio signal through a microphone [3]. In general, the end of a user's utterance is detected by margin, which means passing a certain time from no voice signal [4]. After detected it, the audio signal is processed through speech recognition to convert it into text [5], [6]. Following this, based on the user's utterance text, a large language model (LLM [7]) generates the system's response text. Then, the system converts it into system-generated speech [8]-[10].

By the above processes (the margin for detecting the end of the utterance, speech recognition, generating system response text and converting it into system response voice), "response delay" occurs from the time the user finishes speaking to when the system begins to respond.

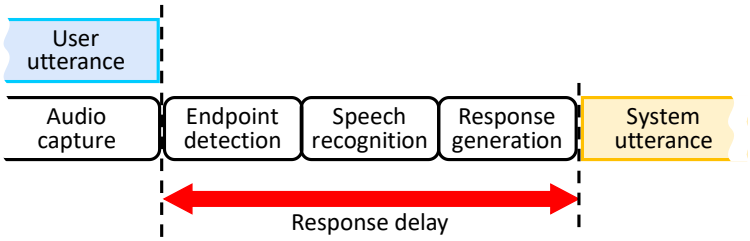

Figure 1. Response delay in a voice dialogue system

3. Related Work

To determine the timing for the system to start speaking (turn-taking), it is necessary to detect the end of the user's utterance [11]. A common approach is to wait for a certain margin of time after the user's utterance input has ceased. If this margin is too short, the system may mistakenly determine that the user has finished speaking due to short silent intervals caused by breathing or getting stuck for words. Generally, a margin of several hundred milliseconds to a few seconds is used (the exact value needs to be confirmed). However, this margin becomes significant for achieving system responses within one second, because in Japanese daily conversations, the next speaker usually starts speaking within one second after the previous speaker finishes [12], [13].

There is a method that can be used that predicts the end of speech based on changes in the prosody of the user's utterance to determine the end of speech without waiting for a margin. In this approach, features of changes in fundamental frequency and sound pressure at the end of speech are utilized to estimate the end of speech based on real-time measurements of the user's voice's fundamental frequency and sound pressure [14].

4. Approach and Issues

Regarding the response delay described above, this paper proposes a method where, instead of waiting for the user's utterance to be completed (by stopping audio capture on the way), speech recognition is performed on the incomplete utterance audio data, and a response is generated using a large language model based on the incomplete utterance text. This approach allows the speech recognition process to start earlier because this system does not wait for the user's utterance to be completed, which in turn enables the system to generate a response more quickly. (Of course, when the system generates a response, it will confirm that the user's utterance has finished at that time before proceeding with the response.) Figure 2 illustrates this proposed approach.

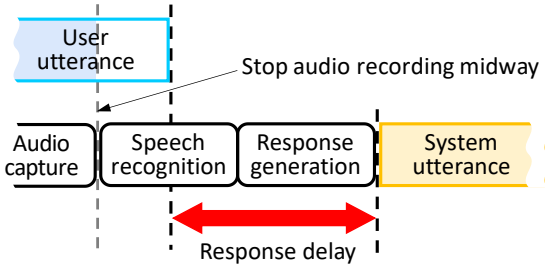

Figure 2. Proposed approach for reducing response delay

However, the proposed approach has a significant issue. Specifically, since responses are generated based on incomplete utterance text, there is a possibility that inappropriate responses may be created. When using large language models like ChatGPT for response generation, these models are designed to predict after

next word in a sequence based on partially completed sentences. While they have the capability to predict the missing parts (such as the end of an incomplete utterance) and generate coherent responses, if the termination which audio capture is stopped is too early, the likelihood of generating meaningful responses decreases.

On the other hand, delaying the termination reduces the effectiveness of latency reduction. Therefore, it is crucial to identify the optimal timing for terminating voice data acquisition. Therefore, it is crucial to identify the optimal timing for terminating voice data acquisition.

5. Proposed Method

In this section, we explain the proposed method to determine the cutoff point based on changes in similarity [15] calculated using embedding representations.

5. 1. Embedding of Dialogue History

Embedding is a method of converting text, images, or other types of data into numerical vectors, enabling computers to efficiently process and analyze them [16], [17]. In the context of text data, embedding converts words or sentences into low-dimensional dense vectors that capture various semantic features. Unlike simple vectorization, embeddings provide more advanced representations by considering the semantic relationships between words [1]. In this context, we use Sentence-BERT [18], a representative method for generating sentence embeddings, to create the embedding representation, denoted as X, of the conversation-history text up to a certain point.

To observe changes in the embedding representation, we track the similarity between a reference embedding and the most recent embedding at the point of interest. Figure 3 shows the concept of the similarity of the two embeddings. We set the initial point t_0, which is the beginning of the target turn (user's utterance), as the reference point. Moreover, we define the embedding representation of the entire conversation history up to the reference point as X_0. As the user's utterance progresses, we define the embedding representation of the entire conversation history up to the most recent point, t, as X_t. As a metric to represent the change in X_t, we use the cosine similarity $S_t(X_0, X_t)$ between X_0 and X_t. It can be written as:

$$S_t(X_0, X_t) = \frac{X_0 \cdot X_t}{\|X_0\|\|X_t\|} \quad (1)$$

Here, $X_0 \cdot X_t$ represents the dot product of X_0 and X_t, while $\|X_0\|$ and $\|X_t\|$ represent the magnitudes of each vector. $S_t(X_0, X_t)$ takes a value between 0.0 and 1.0, and the closer the value is to 1.0, the more similar X_0 and X_t are.

5. 2. Changes in Embedding

Figures 4 through 6 illustrate the changes in S_t for three example conversation sessions, along with the evaluation of the validity of the LLM's responses at each cutoff point in each session. In these graphs, the vertical axis represents similarity, the top horizontal axis shows the number of characters that could be deleted, the middle axis shows the deleted one character (user's utterances in Japanese), and the bottom axis represents

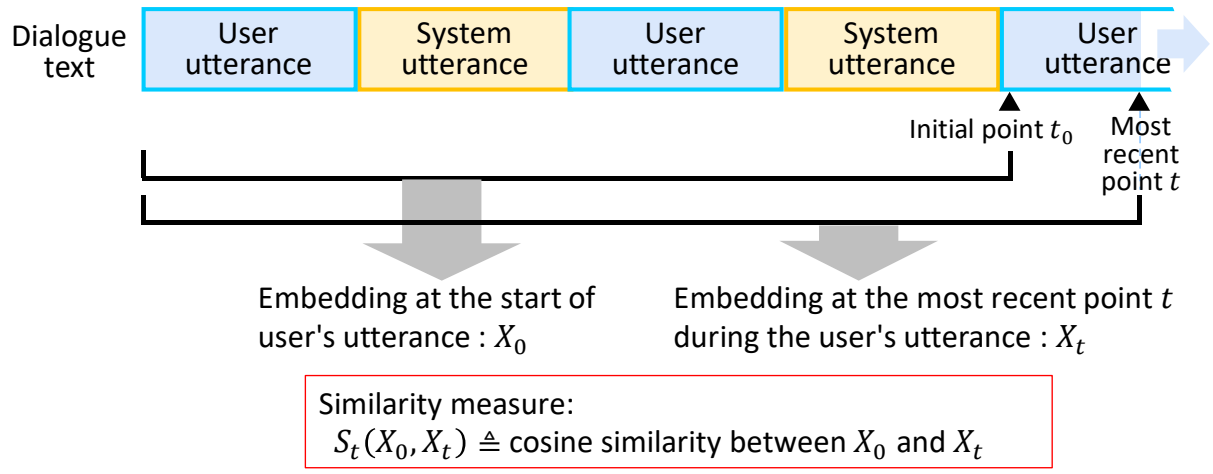

Figure 3. Similarity between a reference embedding and the most recent embedding

the human subjective evaluation of the validity of the LLM's responses. The subjective evaluation uses circles (◯) for Good, triangles (△) for Subpar, and crosses (×) for Bad.

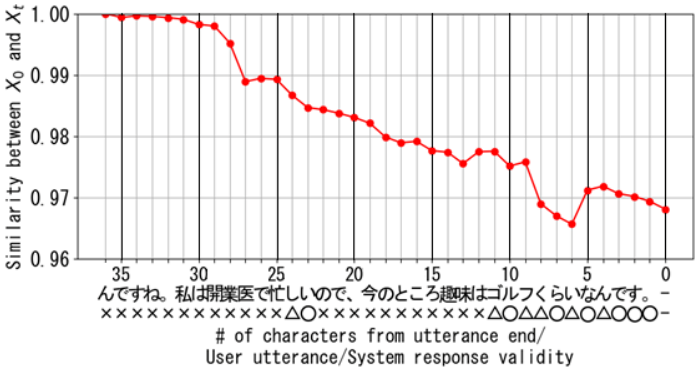

Figure 4. Changes of S_t in conversation session example 1

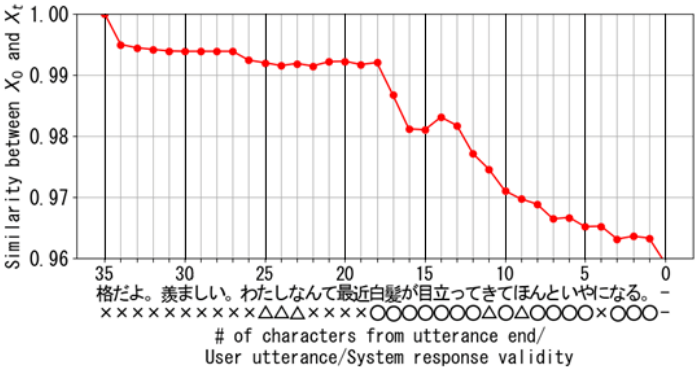

Figure 5. Changes of S_t in conversation session example 2

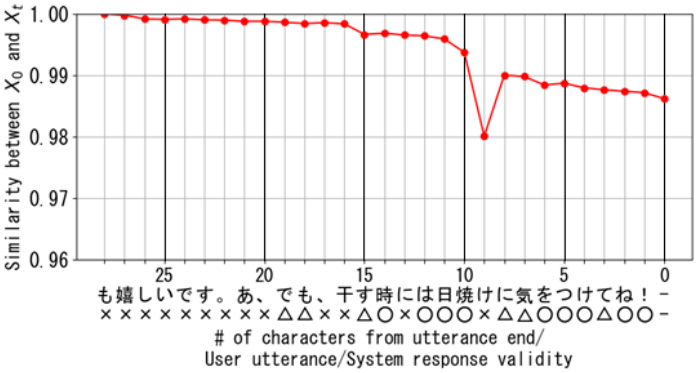

Figure 6. Changes of S_t in conversation session example 3

The relationship between changes in S_t and the variations in the validity of the LLM's responses at each cutoff point differs across conversation sessions. However, as illustrated in the example conversation in Figure 4, cutting off the dialogue too early typically results in low validity of the LLM's responses (×), while a later cutoff point tends to improve the validity (◯). Additionally, as shown in Figures 5 and 6, a common case is that cutting off the dialogue near the point where the similarity drops significantly often results in high validity of the LLM's responses (◯)." Therefore, these graphs suggest that changes in S_t could be used as a metric for determining cutoff points.

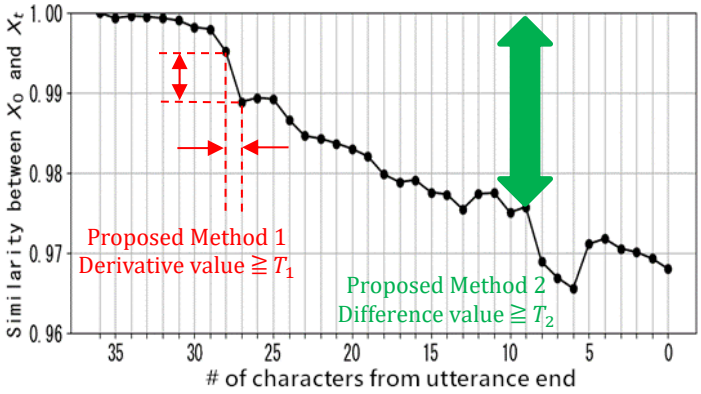

Figure 7. Concepts of Method 1 and Method 2

5. 3. Cutoff Point Determination Method

Three methods (Method 1, Method 2, and Method 3) are proposed to determine the cutoff point based on changes in embedding representation similarity. Figure 7 illustrates the concepts of Method 1 and Method 2.

5. 3. 1. Method 1 : Determination based on the derivative of S_t

In Method 1, the cutoff point is determined where the change in S_t (the derivative of S_t) is large over. The similarity S_t is calculated by the conversation history obtained from the reference point t_0 to a later point t. Compare the values of $S_{t-1} - S_t$ and T_1, and if $S_{t-1} - S_t$ is greater than or equal to T_1, it is determined as the cutoff point (Equation 2).

$$S_{t-1} - S_t \geq T_1 \quad (t = 1, 2, 3 \cdots) \tag{2}$$

Repeat the same evaluation as the user's utterances progress. Further, it continues to capture the conversation until a cutoff point is detected. Once the cutoff point is detected, the conversation capture stops, and the captured audio data up to that point is converted to text via speech recognition. The text is then input into the LLM to generate a response.

If the cutoff point is not detected before the user's utterance ends, it means that no characters can be omitted. In this case, the time shortened by the cutoff method would be zero.

5. 3. 2. Method 2 : Determination based on the decay of S_t

In Method 2, the cutoff point is determined based on the total amount of decay in S_t. Specifically, compare the values of $S_0 - S_t$ and T_1, and if $S_0 - S_t$ is greater than or equal to T_2, it is determined as the cutoff point (Equation 3).

$$S_0 - S_t \geq T_2 \quad (t = 1, 2, 3 \cdots) \tag{3}$$

The process of repetition and the procedure after detecting the cutoff point are the same as in Method 1.

5. 3. 3. Method 3 : Determination with n margin characters

Figure 8 illustrates the concept of Method 3. Similarly to Method 1, the cutoff point is detected based on the derivative of S_t. After detecting the cutoff point in Method 1, the cutoff point for Method 3 is detected after n margin characters. In other words, the cutoff point $t_{method3}$ for Method 3 is detected using the following equation:

$$t_{method3} = t_{method1} + n \tag{4}$$

Here, $t_{method1}$ is the cutoff point detected by Method 1. The process of repetition and the procedure after detecting the cutoff point are the same as in Method 1.

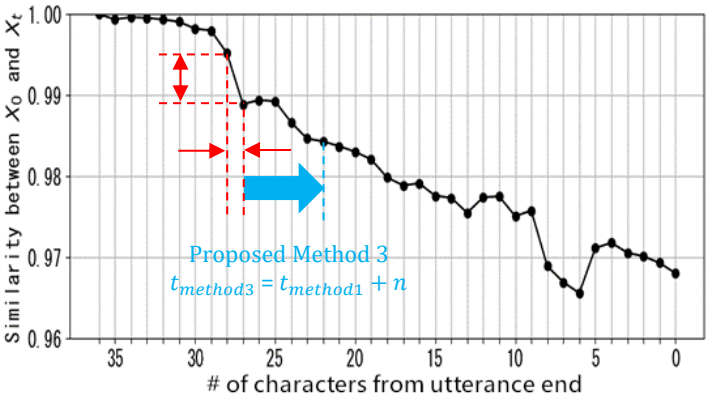

Figure 8. Concept of Method 3

6. Evaluation experiment

In this section, the effectiveness of each proposed method was evaluated. In particular, the evaluation methods, evaluation data, and evaluation results are described below.

6. 1. Evaluation method and Evaluation data

For each method, the following two metrics are evaluated. The first metric is Precision (P_r). Generally, Precision refers to the proportion of positive predictions that are actually correct [19]. In this study, Precision is defined as the proportion of LLM's responses deemed valid (◯) when a cutoff was applied by each method. Since a cutoff is always executed somewhere in each method (if cutoff point is not detected, it is interpreted as a cutoff at the end of the sentence), the number of "positive predictions" in the above definition corresponds to the total number of conversation sessions used in the experiment. Therefore, Precision (P_r) is calculated using the following formula:

$$P_r = \frac{N_{good}}{N_{sessions}} \tag{5}$$

Here, $N_{sessions}$ denotes the total number of sessions, and N_{good} denotes the number of sessions in which the LLM's response was subjectively evaluated as Good (◯) after applying the cutoff.

The second metric is the average number of deleted characters (N_r). It indicates how many characters from the user's utterance were omitted by applying each cutoff method, compared to capturing the entire utterance. Therefore, a higher N_r value is associated with a greater potential to shorten response delays.

The corpus used in this study was collected from dialogues in Japanese between two parties, with A representing the user and B representing the dialogue system. We focused on the 5th utterance of A (which is the user's turn) from the start of the dialogue, applying each cutoff method. To establish a basic approach, we used a text-based corpus [20], which is considered to have less ambiguity compared to colloquial dialogue corpus. From the top 100 data samples in the target corpus, we extracted 56 dialogue sessions where the number of characters in the focused turn was between 30 and 45 characters or between 60 and 100 characters.

6. 2. Evaluation results and Discussion

In Figure 9, the changes in Precision (P_r) for the threshold T_1 of Method 1 and the average number of deleted characters (N_r) are shown by the blue and orange lines, respectively. To see the Figure 9, as T_1

increases, P_r also increases. This can be interpreted as follows: as T_1 increases, the cutoff point is not detected until a larger differential value occurs. As a result, the cutoff point will be shifting later. Consequently, the text input to the LLM becomes longer. This increases P_r, which represents the proportion of correct answers provided by the LLM. Additionally, Figure 9 shows that as T_1 increases, N_r decreases. This is because as T_1 increases, the cutoff point shifts later, leaving fewer characters at the end of the text that can be deleted. As explained above, there is a trade-off between P_r and N_r.

In Method 1, as shown in Figure 9, to achieve P_r of approximately 0.8, a threshold of 0.75×10^{-2} resulted in P_r of 0.79 and N_r of 10.6 (as indicated by the red dashed line in Figure 9).

In Figure 10, the changes in P_r and N_r for the threshold T_2 of Method 2 are shown by the blue and orange lines, respectively. To see the Figure 10, as T_2 increases, P_r increases and N_r decreases. This arises for the same reason as explained in the case of Method 1. In Method 2, as shown in Figure 10, to achieve P_r of approximately 0.8, a threshold of 3.0×10^{-2} resulted in P_r of 0.84 and N_r of 9.0 (as indicated by the red dashed line in Figure 10).

In Figure 11, the changes in P_r and N_r for the detection time margin (n) of Method 3 are shown by the blue and orange lines, respectively. In this case, threshold parameter T_1 in the baseline Method 1 is equal to 0.50×10^{-2}. Theoretically, as n increases, the cutoff point shifts backward. Therefore, P_r should gradually increase from the precision (0.70) observed in the baseline Method 1 when $T_1 = 0.50\times10^{-2}$. However, in reality, to see Figure 11, we observe P_r fluctuates between approximately 0.65 and 0.8 as n increases. This indicates that, depending on the value of n, it is possible to achieve P_r of 0.8 or higher in Method 3.

In Method 3, as shown in Figure 11, setting the threshold T_1 to 0.50×10^{-2} and deciding a cutoff after a margin (n) of 5 achieves P_r of 0.80 and N_r of 14.6 (as indicated by the red dashed line in Figure 11). This result significantly exceeds the N_r of 10.6 observed in Method 1 when P_r was approximately 0.8 (actually 0.79) at T_1 equal to 0.75×10^{-2}.

In case of achieving a Precision of approximately 0.8, the best-performing method was Method 3 in the three methods. This method was able to delete an average of 14.6 characters in Japanese. In Japanese, people speak at a pace of about 6 characters per second. Therefore, this method would shorten the response delay by approximately 2.4 seconds.

Additionally, lowering the target Precision criterion below 0.8 allows for an increase in the average number of deleted characters. For example, when allowing Precision to drop to 0.7, the evaluation results for each method are as follows. In Method 1, with a threshold of 0.5×10^{-2}, Precision decreases to 0.70, while the average number of deleted characters increases to 17.5 (as indicated by the green dashed line in Figure 9). In method 2, with a threshold of 2.5×10^{-2}, Precision is 0.73 and the average number of deleted characters increases to 10.7 (as indicated by the green dashed line in Figure 10). Finally, for Method 3, with a threshold of 3.0×10^{-2} and a margin of 4 characters, Precision is 0.77, and the average number of deleted characters improves to 15.2 (as indicated by the green dashed line in Figure 11).

In case of achieving a Precision of 0.7, the best-performing method was also Method 3 in the three methods. This method was able to delete an average of 15.2 characters in Japanese. In this case, the response delay can be shortened by approximately 2.5 seconds.

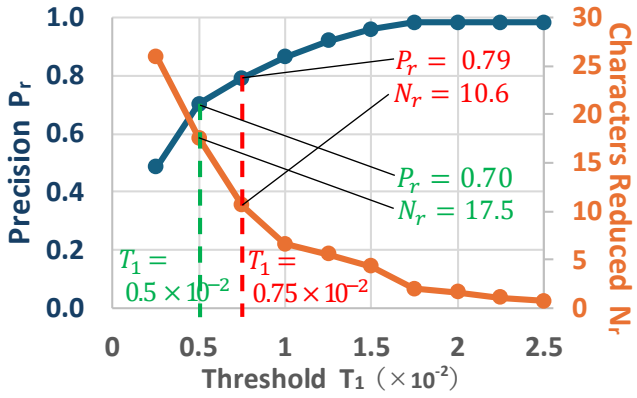

Figure 9. Evaluation result of Method 1

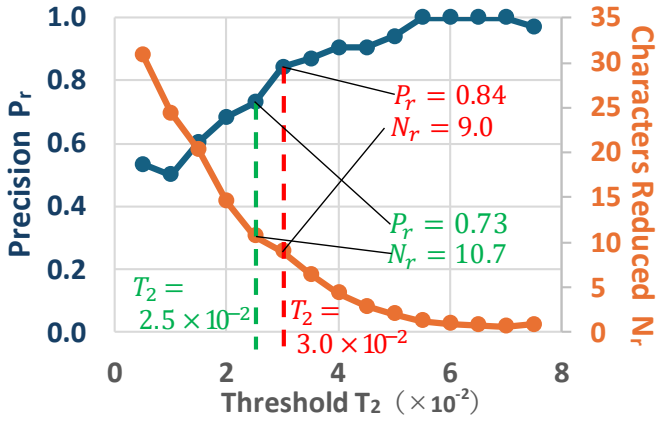

Figure 10. Evaluation result of Method 2

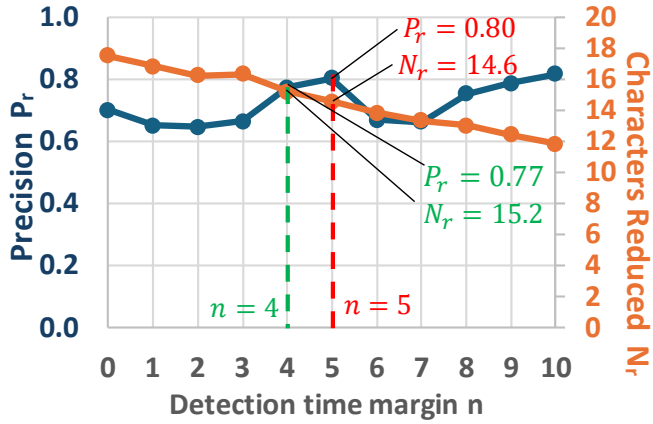

Figure 11. Evaluation result of Method 3 (T_1 = 0.5×10^{-2})

7. Conclusion

We examined a method which is generating responses with a Large Language Model (LLM) based on incomplete utterance to reduce response delay in voice dialogue systems. We have concluded that changes in text similarity may have been related to the appropriateness of the LLM's responses. Therefore, we proposed three methods to determine the cutoff point based on these changes and demonstrate their effectiveness. As a result, in Method 3 (determination based on $t_{method3}$, which is proceeded by n points from the derivative of S_t), we were able to reduce an average of 14.6 characters in Japanese text while maintaining more than 0.8 performance in LLM's responses. This corresponds to approximately 2.4 seconds in spoken Japanese. Therefore, we considered to reduce response delays by the same amount using Method 3.

Nevertheless, the corpus used in this experiment was text-based and comprised more formal expressions. In real-world applications, where human speech will be used, more colloquial expressions are likely to occur. Therefore, future work will focus on evaluating the LLM's performance under such conditions and further refining the proposed methods.

References

[1] Michael Mctear, Conversational AI Dialogue Systems, Conversational Agents, and Chatbots. Springer Nature, 2021.

[2] Kevin Zagalo, Olena Verbytska, Liliana Cucu-Grosjean, and Avner Bar-Hen, "Response Times Parametric Estimation of Real-Time Systems," *arXiv preprint*, arXiv:2211.01720, 2022.

[3] Carlos Arriaga, Alejandro Pozo, Javier Conde, Alvaro Alonso, "Evaluation of real-time transcriptions using end-to-end ASR models," *arXiv preprint*, arXiv:2409.05674, 2024.

[4] Antoine Raux, "Flexible Turn-Taking for Spoken Dialog Systems," Ph.D. dissertation, Computer Science, Carnegie Mellon Univ., Pittsburgh, PA.

[5] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," *arXiv preprint*, arXiv:1412.5567, 2014.

[6] V. M. Reddy, T. Vaishnavi and K. P. Kumar, "Speech-to-Text and Text-to-Speech Recognition Using Deep Learning," *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, Namakkal, India, 2023, pp. 657-666.

[7] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, Ryan Lowe, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27730-27744, 2022.

[8] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," *arXiv preprint*, arXiv:1703.10135, 2017.

[9] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv preprint*, arXiv:1609.03499, 2016.

[10] Jinchuan Tian, Chunlei Zhang, Jiatong Shi, Hao Zhang, Jianwei Yu, Shinji Watanabe, Dong Yu, "Preference Alignment Improves Language Model-Based TTS," *arXiv*, vol. abs/2409. 12403, 2024.

[11] Ryo Masumura, Tomohiro Tanaka, Atsushi Ando, Ryo Ishii, Ryuichiro Higashinaka, Yushi Aono, "Neural Dialogue Context Online End-of-Turn Detection," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, Melbourne, Australia, 2018, pp. 224–228

[12] Shuhei Asaka, Kenji Nishida, Katsutoshi Itoyama, and Kazuhiro Nakadai, "Towards Natural Spoken Dialogue Systems Based on AI Services," *SICE SI2023*, Tokyo Tech Univ., Tokyo, Japan, 3B2-01, 2023, (in Japanese).

[13] Derek Jacoby, Tianyi Zhang, Aanchan Mohan, Yvonne Coady, "Human Latency Conversational Turns for Spoken Avatar Systems," *arXiv preprint*, arXiv:2404.16053, 2024.

[14] Tirza Biron, Daniel Baum,Dominik Freche,Nadav Matalon,Netanel Ehrmann,Eyal Weinreb,David Biron,Elisha Moses, "Automatic detection of prosodic boundaries in spontaneous speech," *PLoS One*, vol. 16, no. 5, 2021.

[15] Fangjia Shen, Timothy G. Rogers, "A Hardware Ray Tracer Datapath with Generalized Features," *arXiv preprint*, arXiv:2409.06000, 2024.

[16] Erhan Sezerer, Selma Tekir, "A Survey On Neural Word Embeddings," *arXiv*, vol. abs/2110.01804, 2021.

[17] Mourad Mars, "From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough," *Applied Sciences*, vol. 12, no. 17, p. 8805, 2022.

[18] Nils Reimers, Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 3982–3992.

[19] Ameet V. Joshi, "Machime Learning and Artificial Intelligence 2nd Edition," 2023, pp. 216-217.

[20] Hiroaki Sugiyama, Masahiro Mizukami, Tsunehiro Arimoto, Hiromi Narimatsu, Yuya Chiba, Hideharu Nakajima, and Toyomi Meguro. Empirical analysis of training strategies of transformer-based Japanese chit-chat systems. In Proceedings of 2022 IEEE Spoken Language Technology Workshop, pp. 685–691, 2023.