Rethinking Dialogue Disentanglement for LLMs via Dialogue-Level Assignment and Subsequent Context

Naoki Takada[1,*], Tatsunori Mori[1]

[1]Yokohama National University, 79-7 Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa, 240-8501, Japan

Abstract

In online chat platforms, multiple dialogues are often entangled in a complex manner. Disentangling them into coherent dialogues is essential for understanding conversations. Supervised models have long dominated this task. The application of large language models (LLMs) to this problem remains underexplored; however, preliminary studies have shown that their performance is substantially inferior to that of conventional non-LLMs methods. This raises fundamental questions: Do LLMs inherently lack the capability for dialogue disentanglement, and what approaches are necessary to enhance their performance? We answer these questions by introducing two novel methods: dialogue-level assignment (DLA), which tasks LLMs with assigning an utterance to a dialogue, and subsequent context (SC), which provides subsequent context as auxiliary evidence. Experiments on the benchmark IRC dataset show that our method, equipped with DLA+SC, achieves new state-of-the-art results across all evaluation metrics. These results prove that LLMs possess a strong capability for the studied task. Furthermore, an ablation study was conducted on open-source models to investigate the effectiveness of DLA and SC, revealing that it is model-dependent. Nevertheless, it also demonstrates that both methods are key factors in improving performance. The findings of this study mark a paradigm shift for dialogue disentanglement, transitioning from conventional non-LLMs approaches to applying LLMs.

Keywords

Dialogue Disentanglement, Multi-Party Chat, Large Language Models (LLMs)

1. Introduction

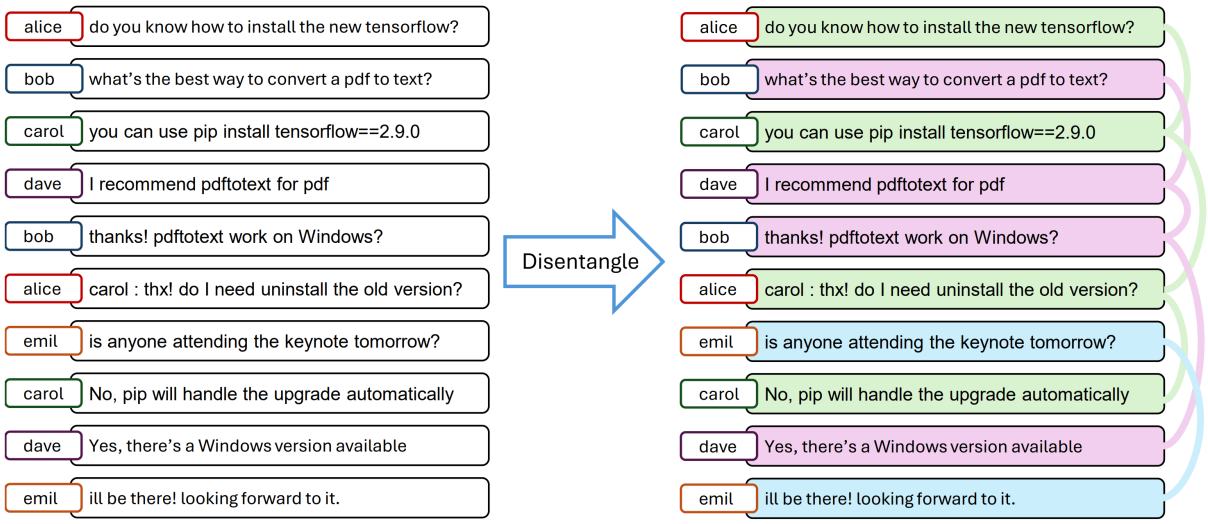

Figure 1: Dialogue disentanglement example: Same color indicates the same dialogue

In multi-party chat platforms such as Slack and Internet Relay Chat (IRC), multiple concurrent

The 10th Linguistic and Cognitive Approaches to Dialog Agents Workshop at the 40th AAAI conference, January 26, 2026, Singapore
*Corresponding author.
takada-naoki-xs@ynu.jp (N. Takada); tmori@ynu.ac.jp (T. Mori)
https://haniwara.github.io/ (N. Takada); https://forest.ynu.ac.jp/mori (T. Mori)
0009-0002-4221-9804 (N. Takada); 0000-0003-0656-6518 (T. Mori)

dialogues frequently become intertwined. This entanglement leads to a chaotic conversational flow. For human participants, following individual dialogues becomes challenging, imposing a substantial cognitive burden. For systems, these entangled dialogues introduce noise into the conversational context and complicate subsequent processing. To address this challenge, the concept of dialogue disentanglement has been proposed [1, 2]. Dialogue disentanglement aims to partition entangled utterances into coherent clusters connected through reply-to relations (Figure 1). This approach supports a wide range of downstream applications, including dialogue state tracking [3, 4] and response generation [5, 6, 7, 8]. This task has predominantly been addressed using pairwise relation classification based on supervised machine learning [2, 9, 10]. In this approach, a model scores every candidate utterance pair within a conversation and makes a local binary decision: is one utterance a direct reply to the other? A global dialogue structure is then assembled from these local links. This approach operationalizes disentanglement by atomizing decisions for machine learning.

Meanwhile, large language models (LLMs) offer strong contextual reasoning and adaptation, suggesting their potential to revolutionize dialogue disentanglement. However, initial attempts to apply LLMs to dialogue disentanglement resulted in substantially worse performance compared to that of conventional non-LLMs methods [11]. This performance gap motivates a re-examination of the task's formulation for LLMs. We propose two methods that recast dialogue disentanglement as a problem suited to LLMs. First, we propose the dialogue-level assignment (DLA) method. Rather than identifying one parent utterance, thereby replicating traditional pairwise relation classification, we apply a method identifying dialogues with reply-to relations. The LLM is provided with dialogue clusters that have been identified from previous assignments. The LLM must then assign the target utterance to one of these existing dialogues or classify it as the start of a new one. Second, we introduce the subsequent context (SC) method, in which the model receives subsequent utterances as auxiliary evidence to inform the assignment.

Our contributions are threefold. First, we focus explicitly on applying LLMs to dialogue disentanglement and develop methods that improve their disentanglement capability. Second, we show that using our method with LLMs achieves state-of-the-art (SOTA) performance on the IRC benchmark. Third, we conducted an ablation study regarding the effectiveness of DLA and SC. The analysis shows the optimal method is model-dependent, but establishes that DLA and SC are key components for improved accuracy.

2. Related Work

2.1. Dialogue Disentanglement

Dialogue disentanglement, also known as conversation disentanglement [1, 2], conversation management [12], or thread detection [13], is a crucial research topic for understanding entangled dialogues. Early approaches framed this task as a two-stage problem. First, a classifier assesses every potential utterance pair to determine whether one is a direct reply to the other. Second, based on this pairwise relation classification, a clustering algorithm assembles these pairs into a dialogue [1, 2]. These initial models relied heavily on handcrafted features, such as lexical overlap, time gaps, and explicit user mentions. The availability of large-scale annotated corpora, notably the Ubuntu IRC dataset [14], facilitated the development of data-driven, end-to-end neural models. A separate, subsequent advancement came with the adoption of fine-tuned pre-trained language models (PLMs), such as BERT [15], which substantially improved performance and set a new standard for these tasks [16, 9].

However, these powerful encoders typically treated conversations as simple, unstructured utterance sequences and thus failed to leverage the structure of the discourse. This prompted a subsequent line of research that focused on explicitly integrating discourse structure to better capture conversational dynamics. Models started to incorporate static, dialogue-specific features, such as speaker identity and user-mention dependencies [10]. Several key refinements have been introduced in the current SOTA model [11], enriching the model with dynamic discourse information, such as time gap and continuously updated reply chains. Hierarchical learning loss has also been integrated with an easy-first decoding

algorithm to capture global conversational properties.

2.2. LLMs for Dialogue Disentanglement

The rise of LLMs has shifted natural language processing paradigms, as these models have shown strong proficiency across many applications [17]. Their power stems from their ability to adapt to new problems through prompts. Foundational paradigms such as in-context learning and few-shot prompting, in which the model learns from a handful of examples provided directly in context, enable LLMs to perform tasks without task-specific training or fine-tuning [18].

Despite these powerful capabilities, the application of LLMs to dialogue disentanglement remains largely underexplored. To the best of our knowledge, the only notable investigation in this area is a preliminary experiment by Li et al. [11]. They framed the task as a zero-shot classification problem, asking an LLM to identify a parent utterance without examples. The task is performed through an approach similar to the pairwise relation classification method using conventional dialogue disentanglement. In this method, LLMs are asked to perform utterance-level assignments. This approach performed significantly worse than conventional non-LLMs methods. Consequently, the potential of LLMs to succeed in dialogue disentanglement has not been fully assessed to date. In this study, we reframed the task by introducing two new formulations: DLA and SC. To our knowledge, this is the first study to place LLMs at the center of dialogue disentanglement—a task long dominated by conventional non-LLMs approaches.

3. Methods

3.1. Task Definition

To formally define the dialogue disentanglement task addressed in this study, we first need to clarify the terminology. We refer to the entire observed sequence of utterances, where multiple dialogues are entangled, as conversation C. Set D represents the partitioned dialogues from C, where each d_j in D is a separated semantic unit comprising a chain of response-related utterances sharing a specific topic or purpose. Accordingly, the task is formalized as follows. The input is a conversation $C = (u_1, u_2, \dots, u_n)$ comprising n utterances in chronological order. Each utterance $u_i \in C$ is a tuple $u_i = (t_i, s_i, m_i)$, where t_i is the timestamp, s_i is the speaker ID, and m_i is the message content. The goal is to partition C into a set of mutually disjoint dialogues $D = \{d_1, d_2, \dots, d_p\}$. This partitioning must be a strict partition of C that satisfies the following two conditions: exhaustiveness ($C = \bigcup_{d_j \in D} d_j$) and exclusiveness ($\forall j \neq k, d_j \cap d_k = \emptyset$). Therefore, dialogue disentanglement is the problem of estimating the optimal set D that satisfies these conditions for a given C.

3.2. Dialogue Disentanglement Methods for LLMs

3.2.1. Core Framework

For the ablation study, we instantiated four methods. First, we describe the common core framework shared by all four methods. The core framework is an iterative greedy algorithm. The LLM processes each utterance chronologically, from u_2 to u_n. The local context for each target utterance u_{target} is constructed based on two fixed hyperparameters: the previous and subsequent window sizes, W_{prev} and W_{subs}, respectively. These values remain constant across all utterances within a single experimental run, rather than being dynamically adjusted. In this study, we selected these window sizes experimentally to evaluate the sensitivity of the model to context length. The LLM's task is to return a single assignment, indicating whether u_{target} belongs to an existing dialogue or initiates a new one. The system updates its state, the set of dialogues D, based on the LLM's decision. This process repeats until all utterances have been assigned, yielding the final partition. The prompts are simple, zero-shot prompts. They ask the model to determine whether the target utterance should connect to exactly one candidate or start a new dialogue, guided by a locality prior that suggests that replies are often temporally close. We did

not conduct an ablation study regarding this locality hint. We utilized the hint to align with the prompt formulation of Li et al. [11]. This ensures experimental consistency with their preliminary study. The LLM is constrained to output a JSON object containing three fields: is_new_dialogue (boolean), an assignment ID, and reason (a textual explanation). The DLA and DLA+SC methods task the LLMs with dialogue-level assignment, thus requiring a "dialogue_id" that links the utterance to an entire dialogue cluster. The Baseline and SC methods perform utterance-level assignment, which requires an "utterance_id" to specify a direct parent utterance. To handle malformed outputs, we implemented an error-handling and retry mechanism. If the generated JSON is invalid or its fields fail type or range validation, the query is re-sent with a corrective instruction appended to the prompt. We set a maximum of five retries for each assignment. Details of the prompts used in this study are listed in Appendix B.

3.2.2. Framework Variations

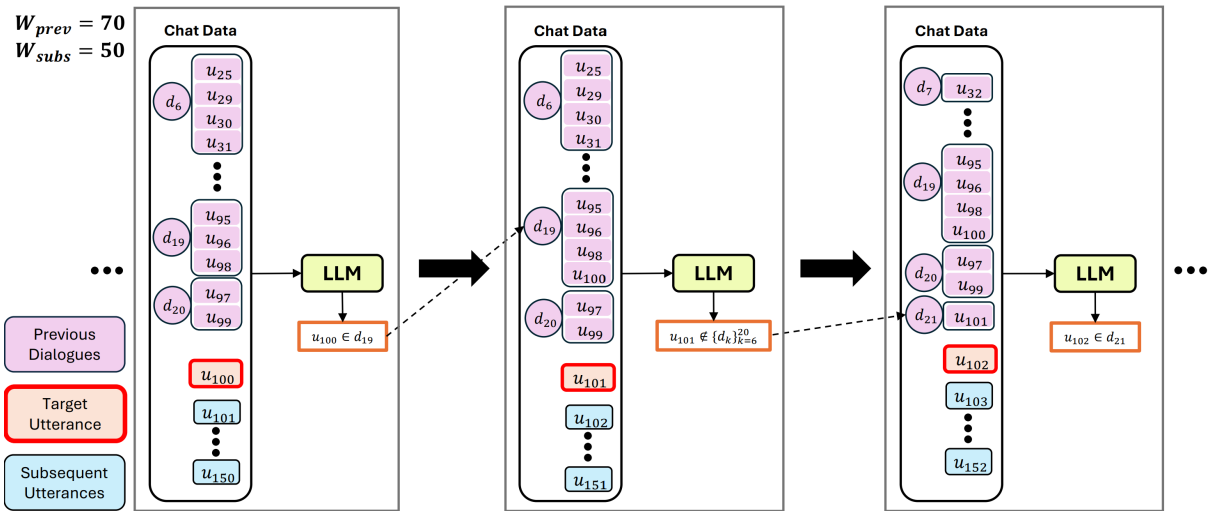

Figure 2: Snapshot of the DLA+SC disentanglement process. The algorithm operates greedily, processing utterances from the start. Based on the assignment and context window size, dynamically set chat data is provided to the LLM. For a target utterance, the LLM determines whether it belongs to an existing previous dialogue cluster or initiates a new dialogue.

Next, we define the four framework variations used in our experiments, which are distinguished by how they provide the LLMs with utterance sequences. The Baseline method provides LLMs with the previous context as a simple chronological sequence of utterances. The LLM's task is to perform utterance-level assignment by identifying which specific previous utterance the target is replying to. This formulation closely mirrors the approach in Li et al. [11]. However, our implementation modifies it by including not only the utterances within the immediate context window but also all other utterances previously assigned to the same dialogues represented in that window. This change ensures consistent information availability across methods to ensure a fair ablation. Further improvements are then applied based on this Baseline method. The DLA method organizes the previous context into coherent dialogue clusters. This structure is built dynamically, reflecting the assignment decisions made at each prior step. The LLM's task is thus transformed into assigning the target utterance to one of these previously established dialogues or initiating a new one. The SC method uses the same utterance-level assignment, but also provides subsequent utterances as auxiliary evidence. The DLA+SC method combines DLA and SC. A schematic workflow of the proposed method is shown in Figure 2. The complete DLA+SC definition is provided in Algorithm 1. Other framework variations are described in Appendix B.

Algorithm 1 DLA+SC (Dialogue-Level Assignment + Subsequent Context)

Input: $C = (u_1, u_2, \dots, u_n)$, where $u_i = (t_i, s_i, m_i)$ with t_i being the timestamp, s_i the speaker ID, and m_i the message content; window sizes W_{prev} and W_{subs}; max retries $N \leftarrow 5$
Output: A partition of C into dialogues $D = \{d_1, d_2, \dots, d_p\}$
function GreedyDisentangle(C, W_{prev}, W_{subs})
    $D \leftarrow \{\{u_1\}\}$
    for $i \leftarrow 2$ **to** n **do**
        $u_{target} \leftarrow u_i$; $C_{prev} \leftarrow \{u_k \mid \max(1, i - W_{prev}) \leq k < i\}$; $C_{subs} \leftarrow \{u_k \mid i < k \leq \min(n, i + W_{subs})\}$
        $D_{candidate} \leftarrow \{d \in D \mid d \cap C_{prev} \neq \emptyset\}$
        $Prompt \leftarrow$ FormatPrompt($D_{candidate}, u_{target}, C_{subs}$)
        $assignment \leftarrow$ "new" ▷ Assignment on persistent failure
        $Prompt^* \leftarrow Prompt$
        for $retry \leftarrow 1$ **to** N **do**
            $Output \leftarrow$ LLM($Prompt^*$) ▷ The LLM returns a reference to $d_j \in D_{candidate}$ or "new"
            $is\_valid \leftarrow$ IsJsonValid($Output$) ▷ Check if JSON is valid
            if is_valid **then**
                $assignment \leftarrow$ ParseJson($Output$)
                break ▷ Exit retry on valid json
            else
                $Prompt^* \leftarrow Prompt$ + warning_message ▷ Append warning for next attempt
        if *assignment* indicates "new" **then** $D \leftarrow D \cup \{\{u_{target}\}\}$ **else** $d_j \leftarrow d_j \cup \{u_{target}\}$
    return D
function FormatPrompt($D_{candidate}, u_{target}, C_{subs}$)
    $D^*_{candidate} \leftarrow \{\}$
    for each $d_k \in D_{candidate}$ **do**
        $S_k \leftarrow \{s \mid \exists u \in d_k, u = (t, s, m)\}$ ▷ Unique list of speakers participating in d_k
        $t_{last} \leftarrow \max\{t \mid \exists u \in d_k, u = (t, s, m)\}$ ▷ Timestamp of last utterance in d_k
        $dt_k \leftarrow t_{target} - t_{last}$ ▷ Time gap from target to last utterance in d_k
        $D^*_{candidate}[k] \leftarrow \{S_k, dt_k, d_k\}$
    $Prompt \leftarrow \text{instruct}_{dise} + D^*_{candidate} + u_{target} + C_{subs} + u_{target} + \text{instruct}_{out}$
        ▷ $instruct_{dise}$: instruction to return the most suitable existing dialogue d_j or "new"
        ▷ $instruct_{out}$: instruction to specify the output format (e.g., JSON)
    return $Prompt$

4. Experimental Setup

4.1. Dataset

The IRC chat dataset [14] has been widely adopted as a benchmark in disentanglement research. It comprises real chat logs for technical problem-solving related to the Ubuntu operating system. Its entangled, asynchronous, multi-participant dialogues make it suitable for evaluating disentanglement models. The dataset was split into train, dev, and test sets. To ensure high-quality ground truth, the dev and test sets were annotated by multiple annotators. All annotation discrepancies were subsequently resolved through discussion and adjudication, thereby creating more accurate annotations. The dev set contained 10 conversations with 250 utterances each, for a total of 2,500 utterances. The test set included 10 conversations with 500 utterances each, for 5,000 total utterances.

4.2. Evaluation Metrics

For direct comparison, we adopted the evaluation metrics from the current SOTA method, Li et al. [11]. These metrics evaluate dialogue disentanglement from three perspectives. First, we used the Variation of Information (VI) [19], Adjusted Rand Index (ARI) [20], and Normalized Mutual Information (NMI) [21] to measure the overall similarity between the gold standard data and the predicted utterance clusters. Second, we used Local_3 [2] to indicate the prediction accuracy when focusing on every three utterances as a local precision metric. Third, we used One-to-One (1-1) [2], Shen-F1 (S-F) [13], and exact matching Precision, Recall, and F1 score (P, R, and F1) [14] to show how many clusters match between the gold standard data and the predictions. Among these, the most stringent evaluation metrics are P, R, and F1, which measure the percentage of predicted dialogues that perfectly match the gold standard data.

4.3. Comparison Methods

We evaluated the proposed method against methods representing major paradigms in dialogue disentanglement. These comparators include a seminal statistical model reliant on handcrafted features [1], a neural model that treats the task as a sequence labeling problem [14], and a transition-based model that formulates disentanglement as an online state-transition process [22]. We also benchmarked against a generative model that predicts reply-to links [23], methods that leverage PLMs for contextual understanding [9], and those incorporating discourse structure [10]. Our primary baseline was the current SOTA method, which integrates additional discourse structure information [11].

4.4. Implementation Details

We evaluated the methods using both proprietary and open-source LLMs. For proprietary models, we utilized GPT4.1 and GPT4.1-mini via the Azure OpenAI Service (specifically version 2025-01-01-preview). We also employed Google's Gemini2.5-pro and Gemini2.5-flash from the stable June 2025 release. For all proprietary model experiments, we set the temperature to 0.0 to minimize variance. However, strict determinism is not guaranteed; therefore, we report results from a single trial. It is worth noting that rigorous reproducibility protocols for API-based models remain unestablished. The output token limit was set to 8,192. The previous context window was fixed to 70 utterances. This size was informed by a preliminary experiment, in which we observed reply dependencies spanning up to 66 utterances. The subsequent window was set to 50 utterances. For open source models, we used the Ollama framework as well as qwen3-32b (Ollama tag qwen3:32b; library ID 030ee887880f), gemma3-27b (Ollama tag gemma3:27b; library ID a418f5838eaf), gpt-oss-20b (Ollama tag gpt-oss:20b; library ID 17052f91a42e), and gpt-oss-120b (Ollama tag gpt-oss:120b; library ID a951a23b46a1). Additionally, we fixed the previous context window to 40 utterances. This reduction was necessary because larger contexts frequently resulted in malformed JSON outputs. We then systematically evaluated subsequent windows of 5, 10, 15, and 20 utterances. All open-source models were configured with a maximum input context (num_ctx) of 16384 and a maximum output token limit of 8192. Similar to the open-source models, the temperature was set to 0.0. For the gpt-oss-120b and gpt-oss-20b models specifically, we set the reasoning effort parameter to "high" in the system prompt.

5. Results

5.1. Preliminary Experiments

We compared our DLA+SC method against the preliminary experiments using DiHRL [11]. Although our Baseline method is conceptually related, it is not identical to the DiHRL method, as it incorporates modifications discussed in Section 3.2.2. We performed this evaluation on a 500-utterance subset of the IRC development set (IDs: 2005-06-27_12, 2005-08-08_01). Table 1 presents the results, with the superior method for each model underlined and the overall best value shown in bold. As these results confirm

Table 1
Comparative results of preliminary experiments applying LLMs to dialogue disentanglement on the IRC development set. The evaluation contrasts two methodologies. DiHRL represents the approach from previous research, which performed utterance-level assignment without access to subsequent context. DLA+SC denotes our proposed method, which executes dialogue-level assignment with subsequent context. All evaluation metrics are reported on a scale of 0 to 100, in which higher values indicate superior performance.

Method		VI	ARI	1-1	NMI	Local_3	S-F	P	R	F1
GPT4.1-mini	DiHRL[11]	57.39	7.84	26.40	70.08	60.48	32.78	0.00	0.00	0.00
	DLA+SC	86.02	62.65	72.60	89.29	84.14	79.59	8.11	8.82	8.45
GPT4.1	DiHRL[11]	58.40	7.65	30.60	70.85	61.76	36.15	0.00	0.00	0.00
	DLA+SC	92.48	78.68	84.40	94.00	93.95	88.74	30.56	32.35	31.43
Gemini2.5-flash	DiHRL[11]	73.51	35.41	50.80	77.31	76.41	55.44	11.11	11.76	11.43
	DLA+SC	**94.59**	**81.05**	**89.80**	**95.52**	96.44	**89.88**	44.12	44.12	44.12
Gemini2.5-pro	DiHRL[11]	85.28	59.57	72.00	86.82	86.16	69.98	33.33	29.41	31.25
	DLA+SC	93.82	70.05	84.80	94.78	**97.31**	86.36	**56.25**	**52.94**	**54.55**

the effectiveness of DLA+SC across all models, we selected this approach for the main experiments on the test set.

5.2. Comparison with SOTA Methods

Table 2
Comparison between traditional non-LLM methods and our proposed LLM-based method on the IRC test set. DLA+SC executes dialogue-level assignment with subsequent context. All evaluation metrics are scaled from 0 to 100. Higher values indicate superior performance.

Method		VI	ARI	1-1	NMI	Local_3	S-F	P	R	F1
Elsner [1]		82.10	—	51.40	—	—	—	12.10	21.50	15.50
BERT [22]		90.80	62.90	75.00	—	—	—	32.50	29.30	36.60
BERT+MF [24]		92.00	—	77.00	—	—	—	—	—	40.90
PtrNet [23]		94.20	—	80.10	—	—	—	44.90	44.20	44.50
DiaBERT [9]		93.20	72.80	79.70	—	—	—	42.10	47.90	44.80
Struct [10]		94.60	76.80	84.20	—	—	—	51.80	51.80	51.70
Struct [11]		93.30	73.60	81.18	89.04	94.08	84.64	41.46	44.23	42.90
DiHRL [11]		94.23	81.10	84.20	91.85	95.64	87.50	47.97	49.86	48.90
GPT4.1	DLA+SC	95.39	82.22	86.34	96.65	95.14	90.55	46.38	48.73	47.53
Gemini2.5-flash	DLA+SC	96.88	90.72	**90.88**	97.69	**97.63**	**92.45**	**60.39**	60.56	60.48
Gemini2.5-pro	DLA+SC	**97.16**	**92.23**	90.78	**97.88**	97.62	92.02	58.81	**63.94**	**61.27**

Table 2 presents the main experimental results on the 5,000-utterance IRC test set. In the table, the highest value among prior works and the highest value among the proprietary LLM model results are underlined, and the highest value across all results is shown in bold. Our proposed method substantially outperforms the SOTA methods. This confirms that LLMs guided by the DLA+SC formulations can decisively outperform conventional non-LLMs methods on this task.

5.3. Ablation Study

We ablated DLA and SC on four open-source LLMs, using exact-match $F1$ as the primary metric. The results are summarized in Figure 3, where *sub* indicates the number of utterances following the target utterance provided as subsequent context. These results show that the optimal formulation is model-dependent; each model required a different method configuration to achieve its peak performance.

Among all setups, qwen3-32b achieved the highest absolute $F1$, indicating that DLA and SC are effective components. Notably, its $F1$ score is comparable to that of the existing SOTA methods shown in Table 2, despite dataset differences. Full results for all experimental setups are provided in Appendix A.

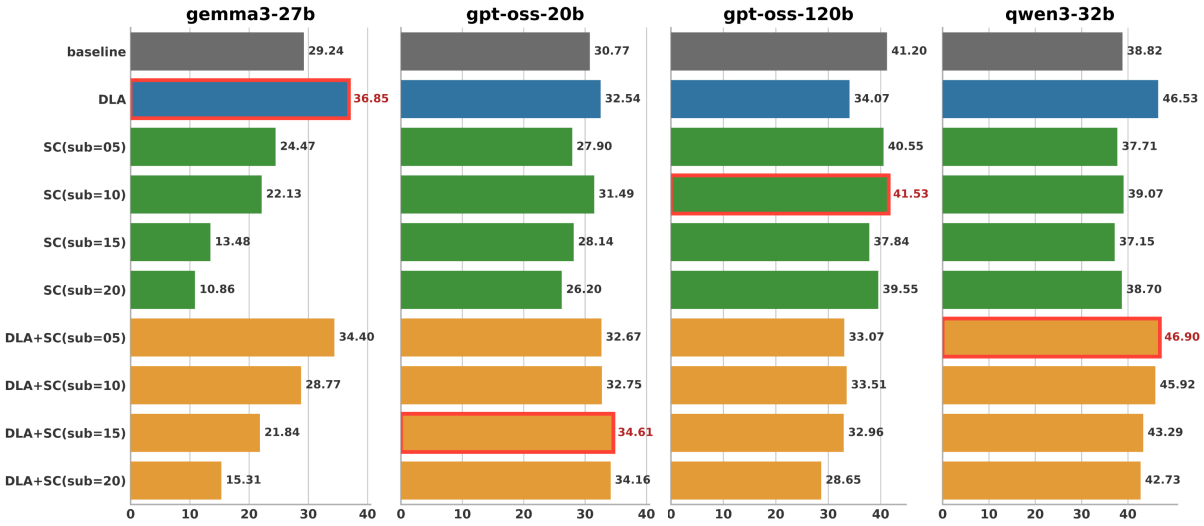

Figure 3: Exact-match $F1$ scores for the ablation study. All scores are reported on a scale of 0 to 100, where higher values indicate superior performance. Methods named DLA perform dialogue-level assignment, whereas those without this label perform utterance-level assignment. Methods named SC indicate the incorporation of subsequent context, where "sub" denotes the number of utterances provided.

6. Analysis and Discussion

The effectiveness of subsequent context is model-dependent, as demonstrated by gemma3-27b. This model's accuracy consistently deteriorated as its subsequent context window was enlarged. We hypothesize that the model misinterpreted these future utterances as valid parent candidates, which complicated the assignment task. This hypothesis is supported by our ablation study results. We initially configured the open-source models to use the same context window size as the proprietary models, but were compelled to reduce this value owing to a high rate of hallucinations. With longer input contexts, the model made errors; it incorrectly assigned the target utterance to a subsequent one. For certain LLMs, subsequent context may therefore act as a distractor rather than as auxiliary evidence, impairing performance. By contrast, the three models other than gemma3-27b demonstrated improved accuracy with the inclusion of subsequent context. This suggests that providing a limited window of subsequent utterances can be effective for dialogue disentanglement. However, its effectiveness is not universal and depends on the model's ability to differentiate evidence from assignment candidates.

We conducted a quantitative failure analysis on the results from all open-source models to identify conditions leading to assignment errors. We evaluated the average success rate by classifying each assignment decision as a binary success or failure based on dialogue cluster overlap. For a target utterance u_t, an assignment is considered successful only if the assigned dialogue cluster shares common utterances with the ground truth cluster within the previous context(u_i, where $i < t$). We investigated the correlation between the average success rate and three potential factors: the number of utterances in the prompt, the count of assignment candidates, and the reply-to relation positional distance. Specifically, we analyzed the trends in success rates relative to the numerical variations of these factors. This examination aimed to determine whether specific quantitative increases negatively impact performance. Our analysis revealed no significant correlation between these factors and assignment accuracy; for simplicity, detailed results are omitted here. However, this analysis is limited to the window sizes used in this study and should not be interpreted as a general characteristic for LLM's dialogue disentanglement.

In future work, large-scale experiments with varying window sizes, including larger contexts, are required to rigorously verify LLMs' dialogue disentanglement ability.

Additionally, we conducted a qualitative error analysis specifically on the results from our best-performing open-source configuration: qwen3-32b with DLA+SC (sub = 05). This analysis identified two primary factors contributing to errors. The first major failure mode involved the misclassification of short, non-substantive utterances such as reactions ("WTF!"), backchannels ("it is"), or greetings ("hi"). Correctly assigning these utterances requires identifying them as reactions within a dialogue the speaker is already participating in. An examination of the model's generated reasoning suggested that it often failed to properly recognize speaker identity. The model appeared to prioritize the semantic content of an utterance over speaker consistency, often defaulting to linking it with a temporally proximate utterance regardless of the speaker. This behavior may be exacerbated by the locality hint provided in our prompt, which instructs the model that temporally closer utterances are more likely to belong to the same dialogue. Misclassified short reactions were often temporally distant from the utterance they responded to. The model thus tended to make locally plausible but globally incorrect assignments. This finding suggests that the inclusion of a locality hint may hinder performance. The second type of common error was due to domain-specific terms, such as "hoary," the codename for Ubuntu 5.04. The model's generated reasoning showed that it understood that the term is related to Ubuntu but failed to correctly link it to the ongoing conversation. It frequently misinterpreted the jargon as the start of a new topic, thereby incorrectly fragmenting a single, coherent dialogue. For technical chats, retrieval-augmented generation could provide domain-specific knowledge and improve accuracy [25].

7. Conclusion

In this work, we demonstrated that LLMs can achieve SOTA performance in dialogue disentanglement. Applying our proposed methods to proprietary LLMs yielded accuracy surpassing that of conventional non-LLMs methods. We introduced two novel formulations to facilitate this achievement: DLA and SC. However, our ablation study with open-source models revealed that the effectiveness of DLA and SC is model-dependent. Although the optimal method varies by model, DLA and SC are proven to be valuable components for improving LLM-based dialogue disentanglement. The findings of this study provide a paradigm shift for dialogue disentanglement, moving from conventional non-LLMs approaches toward LLMs.

8. Future Work

There are three key avenues for future work. First, a more extensive analysis is necessary. Our ablation study was confined to a limited selection of LLMs. Future work should evaluate a wider range of open-source LLMs. Additionally, further investigation into context window sizes is necessary to identify optimal configurations for different models. Second, this study relied on simple, zero-shot prompts with a fixed locality hint. Consequently, comprehensive prompt optimization is required. Future research should investigate advanced strategies such as few-shot and chain-of-thought prompting. Furthermore, the impact of the locality hint must be rigorously ablated. Developing other appropriate hints and refining instruction methods are also essential steps. Finally, the domain dependency must be investigated. This research only focused on IRC data. It is crucial to assess the generalizability of our approach to other conversational contexts. Previous work has shown that supervised models trained on IRC data do not transfer well to Slack chat data without retraining [26].

9. Limitations

Our study has three primary limitations. First, our method is constrained by a context window. It cannot identify reply-to relations beyond the window's boundaries, which inherently limits its ability

to capture long-range relations. Although expanding the window size could mitigate this issue, larger window sizes increase prompt lengths, risking performance degradation and higher computational costs. Second, the use of proprietary LLMs introduces significant financial and temporal costs. The iterative, per-utterance assignment triggers many API calls, and large prompts increase token usage and fees, making the approach slow and costly. Third, deploying open-source models presents challenges regarding computational resources and processing time. Our experiments required a high-end GPU, such as an NVIDIA RTX 6000 Ada, and consumed over 23 GB of VRAM. Furthermore, processing 2,500 utterances took approximately 24 h. These hardware and time requirements may limit the scalability of our approach for larger datasets and may prove prohibitive for researchers without access to substantial computational resources.

Acknowledgments

This paper is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This research is also supported in part by JSPS KAKENHI Grant Numbers JP24K15084 and JP23H00491.

Declaration on Generative AI

The authors used Gemini2.5-pro and GPT5 for assistance with linguistic and technical formatting tasks. An initial draft was composed in our native language. We then utilized the specified generative AI to translate this draft into English. The tool was also used for grammar correction, style refinement, and assistance with formatting the LaTeX code. Following this process, the authors conducted a thorough review and edited the entire manuscript. We take full responsibility for this paper.

References

[1] M. Elsner, E. Charniak, You talking to me? a corpus and algorithm for conversation disentanglement, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL), Association for Computational Linguistics, 2008, pp. 834–842.

[2] M. Elsner, E. Charniak, Disentangling chat, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 117–126.

[3] Y. Ouyang, M. Chen, X. Dai, Y. Zhao, S. Huang, J. Chen, Dialogue state tracking with explicit slot connection modeling, in: Proceedings of the 58th annual meeting of the association for computational linguistics, 2020, pp. 34–40.

[4] Z. Zhang, L. Liao, M. Huang, X. Zhu, T.-S. Chua, Neural multimodal belief tracker with adaptive attention for dialogue systems, in: The world wide web conference, 2019, pp. 2401–2412.

[5] X. Cai, Y. Fu, H. Zhao, W. Jiang, S. Pu, Memory graph with message rehearsal for multi-turn dialogue generation, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), Association for Computing Machinery, Atlanta, GA, USA, 2022, pp. 108–117. doi:10.1145/3511808.3557392, cC BY-SA 4.0.

[6] J. Gu, C. Tan, C. Tao, Z. Ling, H. Hu, X. Geng, D. Jiang, HeterMPC: A heterogeneous graph neural network for response generation in multi-party conversations, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 5086–5097. doi:10.18653/v1/2022.acl-long.349.

[7] Y. Liang, F. Meng, Y. Zhang, Y. Chen, J. Xu, J. Zhou, Infusing multi-source knowledge with heterogeneous graph neural network for emotional conversation generation, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 13343–13352.

[8] P. Ren, Z. Chen, Z. Ren, E. Kanoulas, C. Monz, M. De Rijke, Conversations with search engines: Serp-based conversational response generation, ACM Transactions on Information Systems (TOIS) 39 (2021) 1–29.
[9] T. Li, J.-C. Gu, X. Zhu, Q. Liu, Z.-H. Ling, Z. Su, S. Wei, Dialbert: A hierarchical pre-trained model for conversation disentanglement, arXiv preprint arXiv:2004.03760 (2020).
[10] X. Ma, Z. Zhang, H. Zhao, Structural characterization for dialogue disentanglement, arXiv preprint arXiv:2110.08018 (2021).
[11] B. Li, H. Fei, F. Li, S. Wu, L. Liao, Y. Wei, T.-S. Chua, D. Ji, Revisiting conversation discourse for dialogue disentanglement, ACM Transactions on Information Systems 43 (2025) 1–34.
[12] D. R. Traum, S. Robinson, J. Stephan, Evaluation of multi-party virtual reality dialogue interaction., in: LREC, volume 4, 2004, pp. 1699–1702.
[13] D. Shen, Q. Yang, J.-T. Sun, Z. Chen, Thread detection in dynamic text message streams, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 35–42.
[14] J. K. Kummerfeld, S. R. Gouravajhala, J. Peper, V. Athreya, C. Gunasekara, J. Ganhotra, S. S. Patel, L. Polymenakos, W. S. Lasecki, A large-scale corpus for conversation disentanglement, arXiv preprint arXiv:1810.11118 (2018).
[15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186.
[16] H. Zhu, F. Nan, Z. Wang, R. Nallapati, B. Xiang, Who did they respond to? conversation structure modeling using masked hierarchical transformer, in: Proceedings of the AAAI conference on artificial intelligence, volume 34, 2020, pp. 9741–9748.
[17] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, Journal of Machine Learning Research 24 (2023) 1–113.
[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Advances in Neural Information Processing Systems, volume 33, Vancouver, Canada, 2020, pp. 1877–1901. ArXiv:2005.14165.
[19] M. Meilă, Comparing clusterings by the variation of information, in: Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings, Springer, 2003, pp. 173–187.
[20] L. Hubert, P. Arabie, Comparing partitions, Journal of classification 2 (1985) 193–218.
[21] A. F. McDaid, D. Greene, N. Hurley, Normalized mutual information to evaluate overlapping community finding algorithms, arXiv preprint arXiv:1110.2515 (2011).
[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
[23] T. Yu, S. Joty, Online conversation disentanglement with pointer networks, arXiv preprint arXiv:2010.11080 (2020).
[24] R. Zhu, J. H. Lau, J. Qi, Findings on conversation disentanglement, arXiv preprint arXiv:2112.05346 (2021).
[25] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.
[26] P. Chatterjee, K. Damevski, N. A. Kraft, L. Pollock, Software-related slack chats with disentangled conversations, in: Proceedings of the 17th international conference on mining software repositories, 2020, pp. 588–592.

A. Full Results of the Ablation Study

The highest value for each model is underlined, and the highest value across all results is shown in bold.

Table 3
Full Results of the Ablation Study

Method		VI	ARI	1-1	NMI	$Local_3$	S-F	P	R	F1
gemma3-27b	Baseline	89.84	64.95	75.12	91.13	89.96	78.75	27.71	30.94	29.24
	DLA	91.55	73.94	81.24	92.95	91.56	84.15	36.94	36.77	36.85
	SC(sub=05)	88.63	62.77	72.52	89.70	88.55	77.12	23.11	26.01	24.47
	SC(sub=10)	86.76	55.95	69.52	89.34	85.12	73.47	21.05	23.32	22.13
	SC(sub=15)	83.25	46.10	62.00	87.62	79.62	65.68	13.51	13.45	13.48
	SC(sub=20)	81.23	42.83	59.48	85.29	75.15	62.08	10.96	10.76	10.86
	DLA+SC(sub=05)	91.58	73.73	80.40	93.05	91.01	83.55	35.21	33.63	34.40
	DLA+SC(sub=10)	89.76	67.34	76.24	92.39	88.99	78.88	29.81	27.80	28.77
	DLA+SC(sub=15)	86.08	58.11	68.80	87.92	83.63	70.99	24.44	19.73	21.84
	DLA+SC(sub=20)	83.02	49.59	61.64	84.74	79.52	63.70	17.75	13.45	15.31
gpt-oss-20b	Baseline	90.82	69.49	78.04	92.30	90.42	81.45	29.39	32.29	30.77
	DLA	91.96	76.87	80.24	93.94	91.16	82.67	31.51	33.63	32.54
	SC(sub=05)	90.19	66.86	76.88	91.59	89.58	80.36	26.75	29.15	27.90
	SC(sub=10)	89.74	63.77	75.12	91.60	89.05	78.55	29.96	33.18	31.49
	SC(sub=15)	89.65	61.65	74.72	91.55	89.66	78.05	27.20	29.15	28.14
	SC(sub=20)	89.51	62.91	74.52	91.38	89.42	78.11	25.53	26.91	26.20
	DLA+SC(sub=05)	92.70	78.33	81.88	94.29	92.02	84.16	32.17	33.18	32.67
	DLA+SC(sub=10)	91.78	73.57	79.72	93.68	89.91	82.23	31.91	33.63	32.75
	DLA+SC(sub=15)	92.26	74.74	81.36	94.28	90.24	83.11	34.68	34.53	34.61
	DLA+SC(sub=20)	92.70	78.13	81.96	94.46	91.69	83.19	34.23	34.08	34.16
gpt-oss-120b	Baseline	92.93	76.56	82.64	94.24	92.78	84.18	42.58	39.91	41.20
	DLA	92.03	70.46	80.68	91.51	92.11	80.77	37.91	30.94	34.07
	SC(sub=05)	93.45	78.48	82.52	95.09	93.13	84.88	41.20	39.91	40.55
	SC(sub=10)	94.27	81.36	85.48	**95.51**	93.99	87.86	41.82	41.26	41.53
	SC(sub=15)	93.18	75.74	82.24	94.88	93.47	84.73	38.01	37.67	37.84
	SC(sub=20)	92.75	74.49	81.08	94.71	92.50	83.33	40.09	39.01	39.55
	DLA+SC(sub=05)	89.72	54.03	73.56	87.13	87.59	72.92	39.87	28.25	33.07
	DLA+SC(sub=10)	86.94	41.85	66.96	82.70	82.47	66.11	42.18	27.80	33.51
	DLA+SC(sub=15)	86.43	43.06	64.36	81.92	82.78	64.16	43.70	26.46	32.96
	DLA+SC(sub=20)	88.22	51.91	69.80	84.74	85.00	68.16	38.35	22.87	28.65
qwen3-32b	Baseline	93.44	76.90	81.76	93.13	92.82	86.44	36.65	41.26	38.82
	DLA	94.23	80.54	85.00	95.40	93.41	87.17	**46.43**	46.64	46.53
	SC(sub=05)	93.44	74.39	81.44	93.48	92.86	85.99	35.74	39.91	37.71
	SC(sub=10)	93.87	80.42	83.80	93.54	93.86	87.50	37.10	41.26	39.07
	SC(sub=15)	93.64	80.44	82.96	93.68	93.70	86.90	35.83	38.57	37.15
	SC(sub=20)	93.52	79.48	83.00	93.91	92.63	86.77	37.55	39.91	38.70
	DLA+SC(sub=05)	94.58	82.85	86.24	94.96	94.61	89.18	46.29	**47.53**	**46.90**
	DLA+SC(sub=10)	**94.91**	**83.78**	**86.76**	95.26	**95.15**	**89.49**	45.22	46.64	45.92
	DLA+SC(sub=15)	94.62	82.43	85.92	94.89	93.79	88.89	41.84	44.84	43.29
	DLA+SC(sub=20)	94.37	82.66	85.44	94.72	94.01	88.19	41.99	43.50	42.73

B. Method Details: Algorithms and Prompts

B.1. Baseline

Algorithm 2 Baseline (Utterance-Level-Assignment, No Subsequent Context)

Input: $C = (u_1, u_2, \ldots, u_n)$, where $u_i = (t_i, s_i, m_i)$ with t_i being the timestamp, s_i the speaker ID, and m_i the message content; window sizes W_{prev} and W_{subs}; max retries $N \leftarrow 5$
Output: A partition of C into dialogues $D = \{d_1, d_2, \ldots, d_p\}$

function GreedyDisentangle(C, W_{prev})
  $D \leftarrow \{\{u_1\}\}$ ▷ Initialize the first dialogue with u_1
  for $i \leftarrow 2$ **to** n **do**
    $u_{target} \leftarrow u_i$; $C_{prev} \leftarrow \{u_k \mid \max(1, i - W_{prev}) \leq k < i\}$
    $D_{candidate} \leftarrow \{d \in D \mid d \cap C_{prev} \neq \emptyset\}$
    $C_{candidate} \leftarrow \{u_k \mid 1 \leq k < i,\ \exists d \in D_{candidate} :\ u_k \in d\}$
    $Prompt \leftarrow$ FormatPrompt($C_{candidate}, u_{target}$)
    $assignment \leftarrow$ "new" ▷ Assignment on persistent failure
    $Prompt^* \leftarrow Prompt$
    for $retry \leftarrow 1$ **to** N **do**
      $Output \leftarrow \text{LLM}(Prompt^*)$ ▷ The LLM returns a reference to $u_j \in C_{candidate}$ or "new"
      $is\_valid \leftarrow$ IsJsonValid($Output$) ▷ Check if JSON is valid
      if is_valid **then**
        $assignment \leftarrow$ ParseJson($Output$)
        break ▷ Exit retry on valid json
      else
        $Prompt^* \leftarrow Prompt + \text{warning\_message}$ ▷ Append warning for next attempt
    if *assignment* indicates "new" **then**
      $D \leftarrow D \cup \{\{u_{target}\}\}$
    else
      $d^* \leftarrow d^* \cup u_{target}$ where $d^* \in D,\ u_j \in d^*$
  return D
function FormatPrompt($C_{candidate}, u_{target}$)
  $Prompt \leftarrow \text{instruct}_{dise} + C_{candidate} + u_{target} + u_{target} + \text{instruct}_{out}$
    ▷ $instruct_{dise}$: instruction to return the most suitable existing utterance u_j or "new"
    ▷ $instruct_{out}$: instruction to specify the output format (e.g., JSON)
  return $Prompt$

```
# Instruction
You are given a multi-user chat with each line labeled with an index number,
timestamp, speaker's name, and text message for an utterance. Your task is to identify
to which previous utterance the target utterance is responding. Assign the target
utterance to exactly one existing utterance from candidate utterance IDs, or determine
it starts a new dialogue. Note that the utterance is more likely to be responding to a
nearby one.

# Rule
chat contains system messages that are fundamentally different from user-generated
messages. The following rule should be followed when handling these messages.

## Definition of system message
- "speaker" is "system_message".
- "message" starts with "===".
```

```
Example:
{
  "utterance_id": 1,
  "timestamp": "2023-01-15T10:00:00.000000",
  "speaker": "system_message",
  "message": "=== jack33 [jack33@ca-29palms-cmts2d-189.losaca.adelphia.net] has
entered #channel "
}

## How to assign
Most system messages have no response relationship to other utterances. Generally, you
should set "is_new_dialogue" to true.

# Chat data

## Candidate utterance IDs
138
140
143

...

197
198
199

## Previous utterances
{
  "sequential_utterances": [
    {
      "utterance_id": 138,
      "timestamp": "2016-12-19T21:21:00.000000",
      "speaker": "worktoner",
      "message": "Did the 'top' program get replaced from ubuntu 10 to 14?"
    },

...

    {
      "utterance_id": 199,
      "timestamp": "2016-12-19T21:34:00.000000",
      "speaker": "nacc",
      "message": "figure002: can you pastebin the command and output you are
using/get?"
    }
  ]
}

## Target utterance
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Confirm
You assign this target utterance.
```

```
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Output (JSON ONLY)
Constraints:
- "is_new_dialogue": boolean(true if the target is determined to be the start of a new
dialogue, false if it is a continuation of an existing utterance).
- "utterance_id": If is_new_dialogue is true, set to null. If is_new_dialogue is
false, set the ID of the selected candidate utterance.
- "reason": Detailed explanation based as to why that decision (start of a new
dialogue/continuation of an existing utterance) was made.
Output example:
{
  "is_new_dialogue": false,
  "utterance_id": "17",
  "reason": "..."
}
```

B.2. DLA

Algorithm 3 DLA (Dialogue-Level Assignment)

Input: $C = (u_1, u_2, \ldots, u_n)$, where $u_i = (t_i, s_i, m_i)$ with t_i being the timestamp, s_i the speaker ID, and m_i the message content; window size W_{prev}; max retries $N \leftarrow 5$
Output: A partition of C into dialogues $D = \{d_1, d_2, \ldots, d_p\}$
function GreedyDisentangle(C, W_{prev})
    $D \leftarrow \{\{u_1\}\}$
    for $i \leftarrow 2$ **to** n **do**
        $u_{target} \leftarrow u_i$; $C_{prev} \leftarrow \{u_k \mid \max(1, i - W_{prev}) \leq k < i\}$
        $D_{candidate} \leftarrow \{d \in D \mid d \cap C_{prev} \neq \emptyset\}$
        $Prompt \leftarrow$ FormatPrompt($D_{candidate}, u_{target}$)
        $assignment \leftarrow$ "new" ▷ Assignment on persistent failure
        $Prompt^* \leftarrow Prompt$
        for $retry \leftarrow 1$ **to** N **do**
            $Output \leftarrow \text{LLM}(Prompt^*)$ ▷ The LLM returns a reference to $d_j \in D_{candidate}$ or "new"
            $is\_valid \leftarrow$ IsJsonValid($Output$) ▷ Check if JSON is valid
            if is_valid **then**
                $assignment \leftarrow$ ParseJson($Output$)
                break ▷ Exit retry on valid json
            else
                $Prompt^* \leftarrow Prompt +$ warning_message ▷ Append warning for next attempt
        if $assignment$ indicates "new" **then** $D \leftarrow D \cup \{\{u_{target}\}\}$ **else** $d_j \leftarrow d_j \cup \{u_{target}\}$
    return D
function FormatPrompt($D_{candidate}, u_{target}$)
    $D^*_{candidate} \leftarrow \{\}$
    for each $d_k \in D_{candidate}$ **do**
        $S_k \leftarrow \{s \mid \exists u \in d_k, u = (t, s, m)\}$ ▷ Unique list of speakers participating in d_k
        $t_{last} \leftarrow \max\{t \mid \exists u \in d_k, u = (t, s, m)\}$ ▷ Timestamp of last utterance in d_k
        $dt_k \leftarrow t_{target} - t_{last}$ ▷ Time gap from target to last utterance in d_k
        $D^*_{candidate}[k] \leftarrow \{S_k, dt_k, d_k\}$
    $Prompt \leftarrow \text{instruct}_{dise} + D^*_{candidate} + u_{target} + u_{target} + \text{instruct}_{out}$
        ▷ $instruct_{dise}$: instruction to return the most suitable existing dialogue d_j or "new"
        ▷ $instruct_{out}$: instruction to specify the output format (e.g., JSON)
    return $Prompt$

```
# Instruction
You are given a multi-user chat with each line labeled with an index number,
timestamp, speaker's name, and text message for an utterance. Your task is to identify
to which previous dialogue the target utterance is responding. Assign the target
utterance to exactly one existing dialogue from candidate dialogue IDs, or determine
it starts a new dialogue. Note that the utterance is more likely to be responding to a
nearby one.

# Rule
chat contains system messages that are fundamentally different from user-generated
messages. The following rule should be followed when handling these messages.

## Definition of system message
- "speaker" is "system_message".
- "message" starts with "===".
```

```
Example:
{
  "utterance_id": 1,
  "timestamp": "2023-01-15T10:00:00.000000",
  "speaker": "system_message",
  "message": "=== jack33 [jack33@ca-29palms-cmts2d-189.losaca.adelphia.net] has
entered #channel "
}

## How to assign
Most system messages have no response relationship to other utterances. Generally, you
should set "is_new_dialogue" to true.

# Chat data

## Candidate dialogue IDs
138
149
173
175
180
183
184
186
196

## Previous dialogues
[
  {
    "dialogue_id": "149",
    "participants": [
      "Elementalist",
      "MOUD",
      "MonkeyDust",
      "pavlos",
      "potatolord",
      "ppf"
    ],
    "time_difference_from_target_seconds": 60.0,
    "utterances": [
      {
        "utterance_id": 149,
        "timestamp": "2016-12-19T21:24:00.000000",
        "speaker": "Elementalist",
        "message": "dafuq"
      },

...

      {
        "utterance_id": 190,
        "timestamp": "2016-12-19T21:34:00.000000",
        "speaker": "Elementalist",
        "message": "pavlos here?"
      }
    ]
  },
```

```
...

  {
    "dialogue_id": "196",
    "participants": [
      "Elementalist"
    ],
    "time_difference_from_target_seconds": 60.0,
    "utterances": [
      {
        "utterance_id": 196,
        "timestamp": "2016-12-19T21:34:00.000000",
        "speaker": "Elementalist",
        "message": "ela re patrida"
      },
      {
        "utterance_id": 198,
        "timestamp": "2016-12-19T21:34:00.000000",
        "speaker": "Elementalist",
        "message": "ti programma xrhsimopoieis?"
      }
    ]
  }
]

## Target utterance
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Confirm
You assign this target utterance.
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Output (JSON ONLY)
Constraints:
- "is_new_dialogue": boolean(true if the target is determined to be the start of a new
dialogue, false if it is a continuation of an existing dialogue).
- "dialogue_id": If is_new_dialogue is true, set to null. If is_new_dialogue is false,
set the ID of the selected candidate dialogue.
- "reason": Detailed explanation based as to why that decision (start of a new
dialogue/continuation of an existing dialogue) was made.
Output example:
{
  "is_new_dialogue": false,
  "dialogue_id": "17",
  "reason": "..."
}
```

B.3. SC

Algorithm 4 SC (Subsequent Context)

Input: $C = (u_1, u_2, \ldots, u_n)$, where $u_i = (t_i, s_i, m_i)$ with t_i being the timestamp, s_i the speaker ID, and m_i the message content; window sizes W_{prev} and W_{subs}; max retries $N \leftarrow 5$
Output: A partition of C into dialogues $D = \{d_1, d_2, \ldots, d_p\}$
function GreedyDisentangle(C, W_{prev}, W_{subs})
    $D \leftarrow \{\{u_1\}\}$ ▷ Initialize the first dialogue with u_1
    for $i \leftarrow 2$ **to** n **do**
        $u_{target} \leftarrow u_i$; $C_{prev} \leftarrow \{u_k \mid \max(1, i - W_{prev}) \leq k < i\}$; $C_{subs} \leftarrow \{u_k \mid i < k \leq \min(n, i + W_{subs})\}$
        $D_{candidate} \leftarrow \{d \in D \mid d \cap C_{prev} \neq \emptyset\}$
        $C_{candidate} \leftarrow \{u_k \mid 1 \leq k < i, \exists d \in D_{candidate} : u_k \in d\}$
        $Prompt \leftarrow$ FormatPrompt($C_{candidate}, u_{target}, C_{subs}$)
        $assignment \leftarrow$ "new" ▷ Assignment on persistent failure
        $Prompt^* \leftarrow Prompt$
        for $retry \leftarrow 1$ **to** N **do**
            $Output \leftarrow$ LLM($Prompt^*$) ▷ The LLM returns a reference to $u_j \in C_{candidate}$ or "new"
            $is\_valid \leftarrow$ IsJsonValid($Output$) ▷ Check if JSON is valid
            if is_valid **then**
                $assignment \leftarrow$ ParseJson($Output$)
                break ▷ Exit retry on valid json
            else
                $Prompt^* \leftarrow Prompt$ + warning_message ▷ Append warning for next attempt
        if *assignment* indicates "new" **then**
            $D \leftarrow D \cup \{\{u_{target}\}\}$
        else
            $d^* \leftarrow d^* \cup u_{target}$ where $d^* \in D$, $u_j \in d^*$
    return D
function FormatPrompt($C_{candidate}, u_{target}, C_{subs}$)
    $Prompt \leftarrow \text{instruct}_{dise} + C_{candidate} + u_{target} + C_{subs} + u_{target} + \text{instruct}_{out}$
        ▷ $instruct_{dise}$: instruction to return the most suitable existing utterance u_j or "new"
        ▷ $instruct_{out}$: instruction to specify the output format (e.g., JSON)
    return $Prompt$

```
# Instruction
You are given a multi-user chat with each line labeled with an index number,
timestamp, speaker's name, and text message for an utterance. Your task is to identify
to which previous utterance the target utterance is responding. Assign the target
utterance to exactly one existing utterance from candidate utterance IDs, or determine
it starts a new dialogue. Note that the utterance is more likely to be responding to a
nearby one. Subsequent utterances are provided only as reference information - never
select the id from Subsequent utterances.

# Rule
chat contains system messages that are fundamentally different from user-generated
messages. The following rule should be followed when handling these messages.

## Definition of system message
- "speaker" is "system_message".
- "message" starts with "===".
Example:
```

```
{
  "utterance_id": 1,
  "timestamp": "2023-01-15T10:00:00.000000",
  "speaker": "system_message",
  "message": "=== jack33 [jack33@ca-29palms-cmts2d-189.losaca.adelphia.net] has
entered #channel "
}

## How to assign
Most system messages have no response relationship to other utterances. Generally, you
should set "is_new_dialogue" to true.

# Chat data

## Candidate utterance IDs
136
137
138

...

197
198
199

## Previous utterances
{
  "sequential_utterances": [
    {
      "utterance_id": 136,
      "timestamp": "2016-12-19T21:18:00.000000",
      "speaker": "MOUD",
      "message": "Hey all"
    },

...

    {
      "utterance_id": 199,
      "timestamp": "2016-12-19T21:34:00.000000",
      "speaker": "nacc",
      "message": "figure002: can you pastebin the command and output you are
using/get?"
    }
  ]
}

## Target utterance
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

## Subsequent utterances
[
  {
```

```
    "utterance_id": 201,
    "timestamp": "2016-12-19T21:35:00.000000",
    "speaker": "wedgie",
    "message": "froglok: that depends a lot on your site..."
  },

...

  {
    "utterance_id": 220,
    "timestamp": "2016-12-19T21:42:00.000000",
    "speaker": "nacc",
    "message": "ph88^: `nslookup apt.dockerproject.org` or `dig apt.dockerproject.org`"
  }
]

# Confirm
You assign this target utterance.
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Output (JSON ONLY)
Constraints:
- "is_new_dialogue": boolean(true if the target is determined to be the start of a new
dialogue, false if it is a continuation of an existing utterance).
- "utterance_id": If is_new_dialogue is true, set to null. If is_new_dialogue is
false, set the ID of the selected candidate utterance.
- "reason": Detailed explanation based as to why that decision (start of a new
dialogue/continuation of an existing utterance) was made.
Output example:
{
  "is_new_dialogue": false,
  "utterance_id": "17",
  "reason": "..."
}
```

B.4. DLA+SC

The algorithm is shown in Algorithm 1, so we only show the prompt.

```
# Instruction
You are given a multi-user chat with each line labeled with an index number,
timestamp, speaker's name, and text message for an utterance. Your task is to identify
to which previous dialogue the target utterance is responding. Assign the target
utterance to exactly one existing dialogue from candidate dialogue IDs, or determine
it starts a new dialogue. Note that the utterance is more likely to be responding to a
nearby one. Subsequent utterances are provided only as reference information - never
select the id from Subsequent utterances.

# Rule
chat contains system messages that are fundamentally different from user-generated
messages. The following rule should be followed when handling these messages.

## Definition of system message
```

```
- "speaker" is "system_message".
- "message" starts with "===".
Example:
{
  "utterance_id": 1,
  "timestamp": "2023-01-15T10:00:00.000000",
  "speaker": "system_message",
  "message": "=== jack33 [jack33@ca-29palms-cmts2d-189.losaca.adelphia.net] has
entered #channel "
}

## How to assign
Most system messages have no response relationship to other utterances. Generally, you
should set "is_new_dialogue" to true.

# Chat data

## Candidate dialogue IDs
136
169
173
175
184

## Previous dialogues
[
  {
    "dialogue_id": "136",
    "participants": [
      "Elementalist",

...

      "worktoner"
    ],
    "time_difference_from_target_seconds": 480.0,
    "utterances": [
      {
        "utterance_id": 136,
        "timestamp": "2016-12-19T21:18:00.000000",
        "speaker": "MOUD",
        "message": "Hey all"
      },

...

      {
        "utterance_id": 172,
        "timestamp": "2016-12-19T21:27:00.000000",
        "speaker": "worktoner",
        "message": "Ahh I see they've changed around the commands."
      }
    ]
  },

...

  {
```

```
    "dialogue_id": "184",
    "participants": [
      "figure002",
      "froglok",
      "nacc",
      "pavlos"
    ],
    "time_difference_from_target_seconds": 60.0,
    "utterances": [
      {
        "utterance_id": 184,
        "timestamp": "2016-12-19T21:32:00.000000",
        "speaker": "figure002",
        "message": "hello."
      },

...

      {
        "utterance_id": 199,
        "timestamp": "2016-12-19T21:34:00.000000",
        "speaker": "nacc",
        "message": "figure002: can you pastebin the command and output you are
using/get?"
      }
    ]
  }
]

## Target utterance
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

## Subsequent utterances
[
  {
    "utterance_id": 201,
    "timestamp": "2016-12-19T21:35:00.000000",
    "speaker": "wedgie",
    "message": "froglok: that depends a lot on your site..."
  },

...

  {
    "utterance_id": 220,
    "timestamp": "2016-12-19T21:42:00.000000",
    "speaker": "nacc",
    "message": "ph88^: `nslookup apt.dockerproject.org` or `dig apt.dockerproject.org`"
  }
]

# Confirm
You assign this target utterance.
```

```
{
  "utterance_id": 200,
  "timestamp": "2016-12-19T21:35:00.000000",
  "speaker": "froglok",
  "message": "I installed apache2.. what other packages could I be missing?"
}

# Output (JSON ONLY)
Constraints:
- "is_new_dialogue": boolean(true if the target is determined to be the start of a new
dialogue, false if it is a continuation of an existing dialogue).
- "dialogue_id": If is_new_dialogue is true, set to null. If is_new_dialogue is false,
set the ID of the selected candidate dialogue.
- "reason": Detailed explanation based as to why that decision (start of a new
dialogue/continuation of an existing dialogue) was made.
Output example:
{
  "is_new_dialogue": false,
  "dialogue_id": "17",
  "reason": "..."
}
```