ThreadSumm: Summarization of Nested Discourse Threads Using Tree of Thoughts

Olubusayo Olabisi, Ekata Mitra, Ameeta Agrawal
Department of Computer Science, Portland State University, USA
{oolabisi, ekata, ameeta}@pdx.edu

Abstract

Summarizing deeply nested discussion threads requires handling interleaved replies, quotes, and overlapping topics, which standard LLM summarizers struggle to capture reliably. We introduce *ThreadSumm*, a multi-stage LLM framework that treats thread summarization as a hierarchical reasoning problem over explicit aspect and content unit representations. Our method first performs content planning via LLM-based extraction of discourse aspects and Atomic Content Units, then applies sentence ordering to construct thread-aware sequences that surface multiple viewpoints rather than a single linear strand. On top of these interpretable units, *ThreadSumm* employs a Tree of Thoughts search that generates and scores multiple paragraph candidates, jointly optimizing coherence and coverage within a unified search space. With this multi-proposal and iterative refinement design, we show improved performance in generating logically structured summaries compared to existing baselines, while achieving higher aspect retention and opinion coverage in nested discussions.

1 Introduction

Discussion forums are known to have nested structures, as users respond to one another, quoting or replying to topics of interest, leading to a long thread of content associated with a single post. These multi-turn conversations on discussion forums have additional layers of complexity required to navigate the discussion threads (Liu et al., 2025). There is a tendency for threads to attract off-topic replies that get interleaved with the author's own continuation posts, burying the main content. The nested discourse threads can be represented as graphs connected by reply, retweet, and quote edges, making it inherently harder to follow linearly compared to traditional structured documents. The fragmented and tree-like structure is further branched out by mixed reply types such as reposts, quoted posts and replies as shown in Figure 1, and navigating these branches to follow one coherent strand is crucial for summarization.

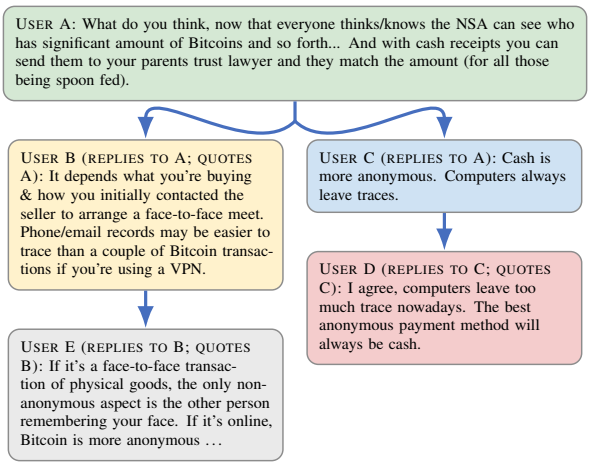

Figure 1: Nested thread structure illustrating reply and quote relations (A→B→E and A→C→D).

Traditional approaches to abstractive text summarization largely focus on optimizing the selection and reduction of document content. While these methods have demonstrated the ability to condense information, they frequently overlook deeper document-level structures, resulting in summaries that may lack coherence, logical flow, and a clear representation of the original text's hierarchy as seen in most discussion forums. Large Language Model (LLM)-based multi-agent frameworks have been used in summarization for long-form documents through hierarchical processing (Chang et al., 2023; Guo et al., 2024; Zhang et al., 2024; Kim and Kim, 2025). However these methods do not balance dynamic viewpoints, especially in noisy, multi-speaker settings (Olabisi et al., 2022).

Early work on structured textual threads and conversations explored abstractive summarization across a spectrum of multi-turn settings. Zhang et al. (2021) introduced a hierarchical model for summarizing multi-turn email threads, capturing

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32225–32240

structure at both the email and token levels. Fabbri et al. (2021) then extended this line to multi-party conversations, integrating argument mining to identify argumentative roles and relationships. Fabbri et al. (2022) carried these ideas into Q&A forums, proposing a pipeline that combines answer selection, content unit extraction, and abstraction to summarize multiple answers to the same question. Building on this, Wu et al. (2025) framed content selection as an iterative self-questioning process that guides summary generation. More recent work, such as (Overbay et al., 2023), further broadens the paradigm by incorporating multimodal signals from images into thread summarization. However, these methods do not employ explicit multi-step reasoning over structure or content units to ensure content coverage and coherence.

In this work, we are concerned with two main research questions. **(RQ1): How do we address discourse coverage, especially when multiple interleaved topics are present in the threaded documents?** In addition, in multi-party speech, turns often overlap and speakers interrupt each other, making it difficult to rely on a simple linear adjacency model to infer what response was targeted to a comment, leading to our second research question **(RQ2): How do we ensure that the overlap and interruptions present in complex threaded discussions are considered in order to generate a coherent summary, even without predefined thread order?**

To answer these questions, we make the following contributions:

- We introduce *ThreadSumm*, a multi-stage LLM pipeline framework for threaded multi-document summarization.
- We employ a thread-aware content-planning layer (Aspects + Atomic Content Units) feeding into a multi-proposal Tree of Thoughts search to iteratively refine the summary for optimized coherence and coverage.
- We conduct extensive experiments to show the effectiveness of our method using three LLMs and three datasets across various domains.

While our framework integrates established ideas such as Aspect extraction, Atomic Content Units, and LLM-based evaluation, our contribution lies in operationalizing these components using Tree of Thoughts as a single controllable pipeline specifically tailored to nested threaded discourse.

2 Related Work

Early neural models incorporated discourse structure into end-to-end architectures: Ishigaki et al. (2019) proposed a discourse-aware hierarchical attention network encoding RST dependencies, and Kwon et al. (2021) introduced NeRoBERTa, which injects nested syntactic and discourse trees into a Transformer encoder. In the area of multi-party speech, Zhang et al. (2021) demonstrated that modeling email thread hierarchy improves abstractive summaries, while Fabbri et al. (2021) and Fabbri et al. (2022) proposed pipelines that identify argumentative roles and cluster responses to preserve multiple perspectives in community Q&A discussions. Query-based summarization of discussion threads was explored by Verberne et al. (2020), who showed that conditioning summaries on user information needs helps surface relevant branches of a discussion. Recent LLM-based frameworks adopt multi-agent or multi-stage processing for long inputs (Guo and Vosoughi, 2024; Zhang et al., 2024; Kim and Kim, 2025). Nevertheless, these approaches either rely on relatively structured interactions or flattened thread structure and optimize for a single-pass summary.

In contrast, *ThreadSumm* explicitly targets unstructured, deeply nested discourse threads by framing summarization as a hierarchical reasoning problem using a thread-aware content-planning layer feeding into a multi-proposal Tree of Thoughts search over paragraph realizations.

3 ThreadSumm

We introduce *ThreadSumm*, a multi-stage LLM pipeline framework that summarizes threaded documents using Tree of Thoughts. Figure 2 presents an overview of the process. First, we decompose content into aspects and atomic units, use Tree of Thoughts reasoning to iteratively reorder and select candidate paragraphs, jointly optimizing coverage and coherence without using predefined thread order. This contrastive design directly addresses the gap between hierarchical summarization of linear documents and the requirements of real-world nested discourse.

Let $D = \{D_1, \ldots, D_m\}$ denote a collection of input documents.

- **Aspects** $A = \{a_1, \ldots, a_n\}$ denote a collection of n aspects representing the *who, what, where* information.

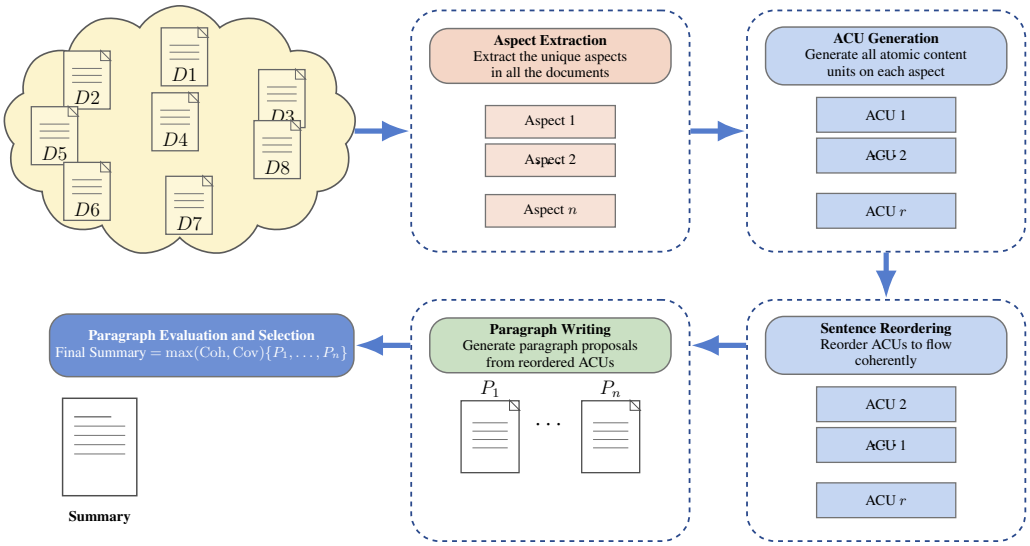

Figure 2: Our proposed framework for Summarization of Nested Discourse Threads

- **Atomic Content Units (ACUs)** $C = \{c_1, \ldots, c_r\}$, denote a collection of r ACUs, where each unit c is a self-contained semantic statement and $r \geq n$.

Aspect Extraction An important step in summarization is deciding the *who, what, where* elements that are present in the input document. It encourages coverage across multiple facets of the source document, instead of over-focusing on the most frequent topic. This ultimately reduces information loss on less frequent but important aspects, which is especially valuable in multi-document settings. We prompt the LLM to extract the aspects in the source document using a few-shot method, the prompt is made available in Appendix A.1.

Atomic Content Unit Generation After extracting the aspects, we generate Atomic Content Units on each of the aspects. Atomic Content Units are self-contained semantic units that represent a single piece of information within a text that cannot be further broken down without losing its meaning (Liu et al., 2022). This step enables pruning, and it is important because it gives a fine-grained, interpretable basis for judging the key informational elements in the source document. It also ensures broader coverage across aspects ensuring that the salience in the source documents is preserved. Similar to the preceding step, we apply a few-shot approach to prompt the LLM to generate the ACUs including a detailed list of the ACU generation process. The prompt is made available in Appendix A.2.

Sentence Ordering To maintain a natural discourse structure, especially given the nested nature of the input documents, we reorder the generated ACUs so that they form a coherent and logical passage - each sentence logically following its predecessor (Logeswaran et al., 2016; Golestani et al., 2021). This results in a connected narrative rather than a list of disjointed statements, maintaining the logical progression and contextual flow needed for the next step. We apply a zero-shot method here, prompting the LLM to reorder the ACU list to follow a logical and coherent flow. The prompt is made available in Appendix A.3. Sentence ordering was done using LLM because of its potential to solve a harder, more global discourse problem than simple heuristics or logic rules can reliably handle. For example, positional/section-based heuristics (e.g., lead sentences first) are not particularly well-suited to threaded discourse representative in our work, where important content appears anywhere in reply chains, not just at the top. Similarly, temporal/chronological rules (e.g., sort by timestamp) could work in principle but would require metadata that is often unavailable in real world settings. Our framework is deliberately metadata agnostic to mimic these constraints, making LLM-based ordering the more robust choice.

Paragraph Writing This step takes the reordered sentences from the previous step as input, and generates multiple coherent paragraph proposals using a zero-shot prompt-based method. The model is prompted to start the paragraph with the first sentence in the reordered sentence list, and

Dataset	Domain	#Instances	Avg Input Length	Avg Summary Length	#Docs/Example	Ref Summary
Reddit	Discussion Forum	250	641	127.1	7.88	Y
Stack	Community Q&A	117	1207	149.5	9.72	Y
Bitcoin *	Bitcoin Forum	1	1250	190.7	26	N

* *included as a case study for thread visualization.*

Table 1: Dataset Statistics

to end the paragraph with the last sentence in the reordered list. The prompt instructs the model to summarize the key points from the set of ordered sentences into a coherent paragraph. This addresses coherence by turning the set of ordered sentences into a fluent well-structured paragraph that reads as a single cohesive unit rather than a list of sentences. The prompt also explicitly instructs the model to not omit any of the sentences when summarizing the key points into a paragraph. By conditioning on the full set of ordered sentences, the model should integrate and balance all perspectives or aspects mentioned, rather than selectively omitting minority views. The prompt is shown in Appendix A.4.

We employ the Tree of Thoughts (ToT) approach introduced in Yao et al. (2023) to iteratively generate paragraph proposals, selecting the best based on coherence and coverage scores. Coverage measures how well the summary includes the important information in the source document, this is optimized to address **RQ1**. To address **RQ2**, we optimize coherence which measures the idea connectedness and logical flow in the summary. ToT takes a given number of steps s as a parameter, and for each step, it takes multiple proposal generation of sentence reordering and paragraph candidates, and evaluates each candidate against the original source document for its coherence and coverage using an LLM prompt shown in Appendix A.5. The reordering associated with the highest-scoring paragraph gets carried forward to the next iteration, allowing the model to explore a more diverse set of "thoughts" and potentially converge on a higher quality output. This iterative refinement and branching significantly enhances the generation of coherent and comprehensive paragraphs by systematically exploring and evaluating multiple compositional paths for a more robust search for the best summary.

Our method employs multi-step reasoning by decomposing thread summarization into a sequence of structured interdependent reasoning modules tailored specifically to threaded discourse, each step performs a distinct reasoning task with intermediate outputs that are passed on to the next step to generate the final summary

4 Experimental Setup

This section describes the implementation details, dataset, baseline methods and evaluation metrics used in our experiments.

4.1 Implementation Details

We experiment with three LLMs ordered from proprietary to open-source, both larger and smaller models, including GPT-4[1] (*gpt-4-turbo*), Claude[2] (*claude-3-sonnet*), and LLaMA 3 (*llama3-70b*)[3]. All detailed prompts and model parameters are shown in Appendix A.

4.2 Dataset

Table 1 presents the overall characteristics of the datasets. In a threaded document setup, one set of documents represents a single thread consisting of one root document or post, plus all its nested replies and sub-replies. We make use of conversational datasets like StackOverflow (Stack) (Hoogeveen et al., 2015), a community Q&A forum, and Reddit (Zhang et al., 2017), from CoarseDiscourse. For both datasets, we make use of the preprocessed test set from Fabbri et al. (2021). We selected the Reddit and StackOverflow datasets from ConvoSumm because they most closely match the discourse characteristics our method is designed to address, namely complex, multi-party threaded discussions with heterogeneous and branching structures. Both Reddit and StackOverflow contain open-ended, community-driven threads in which many participants contribute diverse viewpoints across nested replies, reflecting the core challenges of real-world discussion forums. In contrast, the NYT dataset is centered on responses to a single

[1] https://developers.openai.com/api/docs/models/gpt-4-turbo

[2] https://platform.claude.com/docs/en/about-claude/models/overview

[3] https://www.llama.com/models/llama-3/

		Reddit			Stack			Bitcoin		
		QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1
GPT-4	Vanilla	36.46	30.16	30.54	33.46	**35.34**	32.24	71.43	48.17	5.44
	arg-graph	38.46	30.18	27.11	34.55	34.44	29.22	**85.71**	57.86	4.96
	CHRONOS	41.49	30.31	28.59	33.82	35.23	31.62	71.43	46.26	6.00
	mRedditSumm	38.23	**30.76**	29.55	34.55	35.10	31.84	28.57	**61.08**	5.92
	ThreadSumm (ours)	**50.34**	30.40	**33.30**	**49.94**	33.46	**35.89**	57.14	40.07	**15.28**
CLAUDE	Vanilla	38.34	29.51	30.88	40.05	36.79	31.86	28.57	45.54	4.56
	arg-graph	40.91	32.01	30.19	42.61	37.07	30.48	28.57	57.44	6.72
	CHRONOS	45.43	32.11	26.35	41.64	33.56	27.76	71.43	**67.73**	4.96
	mRedditSumm	45.66	30.64	26.82	39.93	34.87	29.22	57.14	62.28	5.20
	ThreadSumm (ours)	**55.66**	**42.42**	**34.37**	**57.75**	**50.29**	**39.29**	**85.71**	63.72	**22.16**
LLAMA	Vanilla	33.20	28.66	**30.53**	32.84	32.46	32.69	57.14	41.33	6.08
	arg-graph	35.49	30.93	24.21	36.75	38.97	27.35	28.57	49.80	6.16
	CHRONOS	43.09	28.82	27.10	31.50	36.41	28.82	42.86	31.67	5.20
	mRedditSumm	40.34	30.34	30.37	37.36	35.00	31.96	28.57	45.52	6.24
	ThreadSumm (ours)	**50.00**	**33.98**	25.25	**51.28**	**42.44**	**38.39**	**71.43**	**59.36**	**20.08**

Table 2: Results comparing *ThreadSumm* and other baselines across three models, three datasets, and three evaluation metrics.

news article and exhibits a more constrained, less-branching structure, while the Email dataset involves a small number of participants and predominantly linear reply chains. These settings are less representative of the discourse complexity our approach targets.

We also make use of a small amount of data from a bitcoin forum, Bitcoin[4], which consists of conversation threads, with an average discussion tree depth and breadth of 4 by 6. We include this dataset as a case study as it allows us to perform a more granular analysis given the visual hierarchical structure not included in the other datasets.

4.3 Baselines

We conduct experiments using our proposed method, and compare against the following methods which are adapated to use the same underlying LLMs as our method. **Vanilla** - this is a simple prompt-based method using LLMs with the prompt *"You are an expert text summarizer tasked with providing summaries of documents. Please provide a summary of the text. Do not itemize the summary"*. **arg-graph** (Fabbri et al., 2021) uses argument mining through graph construction to model the issues and viewpoints, and use a sequence-to-sequence model to generate summaries. **CHRONOS** (Wu et al., 2025) uses a retrieval based approach to timeline summarization by iteratively posing questions about the topic and the retrieved documents to generate chronological summaries. **mRedditSumm** (Overbay et al., 2023) uses an approach of comment clustering, cluster summarization, and cluster summary summarization. Although this method was proposed for text and images, we make use of the text-only method.

[4] https://bitcointalk.org/index.php

4.4 Evaluation Metrics

We employ the following metrics for evaluation. **QAGS** (Wang et al., 2020) uses a question-generating model and a question-answering model to evaluate the quality of generated summaries by detecting factual inconsistencies. **SummaC** (Laban et al., 2022) is an NLI-based method used to assess factual consistency by evaluating whether the generated summary is entailed by the original documents, we report the SummaC-Conv variant. **ROUGE** (Lin, 2004) calculates the lexical overlap between the machine-generated output and the human-written reference summaries. To align with coverage evaluation, we report the recall scores of ROUGE-1 (overlapping unigrams) for our experiments. Since the Bitcoin dataset has no reference summary, we directly use the source document for the overlap as introduced in Bao et al. (2023).

5 Results and Discussion

5.1 Meta Evaluation

Table 2 presents the detailed results of our experiments. We observe that *ThreadSumm* consistently outperforms all baselines across all three models

(GPT-4, Claude, LLaMA) and all three datasets (Reddit, Stack, Bitcoin) on almost every metric. The performance gains from *ThreadSumm* are substantial, often exceeding baselines by large absolute margins (e.g., +10 to +20 points on QAGS or ROUGE-1), and is the only method that generalizes robustly across domains. Although the performance boost provided by *ThreadSumm* is evident regardless of the underlying LLM used (GPT-4, Claude, or Llama), *ThreadSumm* paired with Claude achieves the best overall performance in the entire table. Even as a smaller and open-source model compared to the others, Llama with *ThreadSumm* remains highly competitive. CHRONOS and mRedditSumm remain the most competitive baselines. Interestingly, we observe that although the ROUGE scores on the Bitcoin dataset are significantly lower compared to the other datasets, and this can be attributed to the use of the actual source document as the reference document in the calculation, it is worthy of note that *ThreadSumm* still significantly outperforms all other methods, achieving double digit scores, while all the other methods have single digit scores. To better align evaluation with our objectives, we complement these standard metrics with targeted analyses of positional bias, aspect retention, and opinion-cluster coverage, and with human ratings focused specifically on coherence and coverage.

5.2 How does positional bias manifest in nested discourse threads?

Positional bias is a known challenge in text summarization, this form of bias happens when the model relies on where information appears in the input rather than the content salience, making generated summaries overly sensitive to sentence order (Olabisi and Agrawal, 2024; Mayilvaghanan et al., 2025; Ma, 2025). It is essential to control positional bias when summarizing nested discourse threads because posts are often a mix of replies and quotes to either the initial post or replies to the initial post. If a summarizer favors early-position documents, whichever group or perspective that appears first will be overrepresented in the summary thereby creating a distorted view. To evaluate position bias in our method, we compute the proportion of source sentences covered by the summary as a function of their position in the source. We employ the method in (Sun et al., 2025) to identify sentences underrepresented or omitted in the generated summary using semantic similarity, softmax normalization, quantile transformation, and a dynamic threshold. A sentence is considered represented if it is greater than or equal to the dynamic threshold. To better visualize if the summary content is skewed to a certain part of the source document, we plot the position coverage indicating which source sentences are represented in the summary. Figure 3 shows the position of represented vs. non-represented sentences in the source document. We see that only 6 of the 62 sentences are not represented, and also that our method does not favor a specific location, with the sentence position not represented spread out. We further analyze the omitted sentences in Appendix C and find that they are sentences with no major significance to the key points.

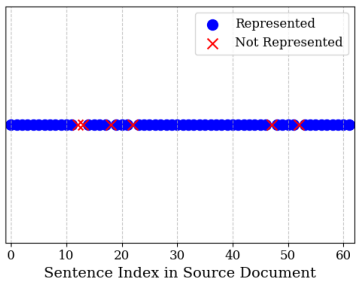

Figure 3: Represented vs. Not represented sentences using *ThreadSumm* on Bitcoin dataset. The x-axis represents the index of the sentence in the source document.

5.3 Aspect retention in summary

Summarization models should be able to preserve the key elements in the source document without information loss. These key elements (aspects) contain information about the *who, what, where* present in the document. A good summary must cover those aspects rather than just high-frequency surface content. This is closely related to balanced coverage which ensures that summaries include all important aspects instead of over-focusing on just those that occur frequently, as is common in nested discourse threads. To measure this, we compute the aspect overlap between the source document and summaries by extracting the aspects using the prompt *"Extract the unique aspects (places, names, products, features, objects) mentioned in the following document"*. After extracting the aspects from both source document and summary, we count the occurrence of common aspects in both lists and present the final score as a fraction of the common aspects relative to the full aspects in the source document. The results are plotted in Figure 4, and we observe that our method yields the highest as-

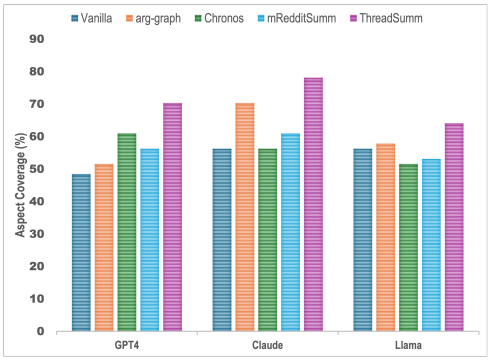

Figure 4: Aspect Coverage of the summary relative to the source document on Bitcoin dataset.

		Reddit	Stack	Bitcoin
GPT-4	Vanilla	0.35	0.49	0.40
	arg-graph	0.37	0.48	0.40
	CHRONOS	0.34	0.51	0.60
	mRedditSumm	0.34	0.50	0.40
	ThreadSumm (ours)	**0.54**	**0.58**	**0.60**
CLAUDE	Vanilla	0.40	0.51	0.40
	arg-graph	0.39	0.51	0.60
	CHRONOS	0.34	0.45	0.60
	mRedditSumm	0.34	0.43	0.40
	ThreadSumm (ours)	**0.64**	**0.75**	**0.80**
LLAMA	Vanilla	0.33	0.45	0.40
	arg-graph	**0.34**	0.48	0.60
	CHRONOS	0.30	0.42	0.20
	mRedditSumm	0.32	0.48	0.60
	ThreadSumm (ours)	0.32	**0.62**	**0.80**

Table 3: Opinion analysis comparing ThreadSumm and other baselines across the three models and datasets.

pect overlap as compared to the baselines across all models. This is as expected as we included aspect extraction as the starting point in our framework to optimize aspect overlap, and this ultimately yields more informative and less biased summaries. In addition, we observe that Claude consistently has the best performance across all methods.

5.4 Opinion Diversity Analysis

Coverage across distinct viewpoints is essential in nested threaded documents to ensure that not only the most popular or repeated viewpoints are included in the summary. Ideally, a good summarizer should be able to capture the opinion of the original poster and the diverging opinion of the responders. To evaluate the performance of our method on opinion diversity coverage, we conduct an analysis to compare the opinion diversity in the generated summary to the source documents. We achieve this by clustering source sentences into opinion clusters using sentence embeddings and K-means using *all-MiniLM-L6-v2* from the Sentence Transformers family. A cluster is considered covered if at least one summary sentence is semantically aligned with a source sentence in that cluster, beyond a similarity threshold. We set the clusters $k = 5$ to account for a variety of stances/viewpoints, and similarity threshold $t = 0.6$. In this sense, coverage is used as a measure of opinion diversity preservation, ensuring that the summary is touching all the different positions that exist, not just the prominent opinions. The final coverage is computed as the fraction of source clusters represented in the summary. We compare our method against the baselines, and the results in Table 3 show that *ThreadSumm* mostly achieves better opinion coverage compared to other methods. This indicates that our method performs better at preserving opinion balance rather than factual redundancy.

	Coverage	Coherence
Vanilla	3.4	3.5
arg-graph	3.5	3.3
CHRONOS	3.7	3.7
mRedditSumm	3.0	3.1
ThreadSumm (ours)	**4.9**	**4.6**

Table 4: Human Annotation results comparing our method to other baselines.

5.5 Human Evaluation

We also collect human evaluation on the three datasets to compare the summaries generated from *ThreadSumm* versus the baselines. Since our research goal is primarily focused on improving discourse coverage and coherence, we use the same two quality dimensions for human evaluation. We ask the annotators to rate the summaries with respect to the source document on a scale of 1-5 (1 being the worst, and 5, the best). We provide clear guidelines to the annotators, as shown in Appendix B. We collect evaluation from two annotators for each summary, and compute the inter-annotator agreement on coherence and coverage provided by the annotators using Krippendorff's alpha. The result in Table 4 shows that our method performed better on both coherence and coverage with medium to high inter-annotator agreement of 0.67 and 0.82.

	Uncontrolled Length (number of words)				Controlled Length (number of words)			
	Length	QAGS	SummaC	ROUGE-1	Length	QAGS	SummaC	ROUGE-1
Vanilla	107	37.29	30.71	31.53	76	28.12	25.19	24.01
arg-graph	104	39.47	31.28	33.62	71	31.29	27.03	23.72
CHRONOS	103	43.22	30.85	34.91	70	37.85	27.61	25.29
mRedditSumm	117	44.93	34.12	36.44	78	39.37	**29.55**	26.38
ThreadSumm (ours)	169	**52.74**	**34.79**	**40.18**	73	**45.12**	26.41	**27.93**

Table 5: Summary Length Analysis

5.6 Summary Length Sensitivity

We run a controlled experiment with summaries set to length = 5 sentences and compute new scores in Table 5. We observe that shortening the summaries lowers performance on all metrics, with ROUGE being the most sensitive to length reduction, followed by QAGS, while SummaC changed only marginally. However, under both settings, controlled length as well as uncontrolled length, *ThreadSumm* mostly outperforms other methods, except on SummaC where mRedditSumm has the highest score. In addition, we conducted a brief analysis comparing our method to mRedditSumm on the Bitcoin dataset under reduced depth and breadth of the input document set, and observed comparable performance between both methods only when both depth and breadth of the document set were limited to 2. This further shows the advantage of our method on more complex, deeply nested hierarchical discourse structures.

5.7 Qualitative Analysis

We manually analyze the summaries qualitatively in addition to the automated evaluation. Table 6 shows a sample source document, the corresponding summary generated using the vanilla prompt, mReddit method, and the summary generated using our method. We observe that the vanilla method gives a very high-level summary, omitting key points like the current listing price, car mileage, and country of production. This is also observed in the mReddit summary, where there is no mention of the platform where the car was listed, the car mileage or country of production. Our method, on the other hand, covers a lot more key points in the source document (highlighted in green font), yielding longer summaries, while still maintaining a coherent flow of information from the first sentence to the last sentence. Also notable is the fact that slangs such as "grand" (thousand), and acronyms such as "AUS" (Australia) are not omitted, but properly represented in the summary.

5.8 Ablation

Effect of Aspect Extraction, Sentence Reordering, and Tree of Thoughts To assess the contribution of **Aspect Extraction**, we remove this component from our pipeline and report results on the Bitcoin dataset with the CLAUDE model (Table 7). While the ROUGE-1 drop is under 1 point, QAGS and SummaC degrade sharply, indicating that Aspect Extraction is critical for factual consistency and overall summary quality. To measure the effect of **Sentence Reordering**, we compare our framework without this step. We see that removing Sentence Reordering leads to an even larger decline in SummaC and ROUGE-1 than omitting Aspect Extraction, confirming that reordering sentences before generation is important for producing a coherent, connected narrative. Finally, we ablate the **Tree of Thoughts** step. Although QAGS remains comparable to the settings without Aspect Extraction and Sentence Reordering, both SummaC and ROUGE-1 drop substantially when Tree of Thoughts is removed. This suggests that Tree of Thoughts is the single most impactful component in our pipeline, yielding the largest overall gains.

Effect of Number of Proposals To assess the effect of the number of proposals in the ToT step, we analyze how our method performs with varying number of proposal parameters. Table 8 provides an initial sensitivity analysis over the number of reordering and paragraph proposals on the Bitcoin case study. While we observe that different configurations can shift QAGS, SummaC, and ROUGE in non-uniform ways, the gains from significantly larger search are not consistently monotonic across metrics. These mixed patterns suggest that more aggressive ToT search does not automatically translate into better summaries, and that different evaluation metrics may favor different regions of the ToT hyperparameter space.

Input : Current car is surely on its way out. Too much rust and too much shit going wrong with it to warrant all the repairs that it needs. I found a 1988 Mazda 323 on Craigslist with sub-30k miles on it in pretty much new condition. What's the most you'd pay for something like this? </s>I might buy it first... Late 80's were epic for Japan. Keep on the look out for an AZ-1 as well ;) </s>I wouldn't pay $4900, that's for sure. </s>Definitely not ;) </s>People ask for outrageous sums for these low mileage cars that nobody cares about. I'm guessing the same is the case here?..... **(truncated 20 lines)**
Vanilla Summary : The user is considering buying a 1988 Mazda 323 with low mileage, found on Craigslist, but thinks the current price is too high. The car has been barely used, with a little old lady driving it only once a week, but this lack of use may have led to neglect of regular maintenance, such as replacing the timing belt. The user and others discuss what a fair price for the car would be, with suggestions ranging from $2,000 to $3,500, and some expressing a fondness for 80s cars, including the Mazda 323, for their reliability and low maintenance costs.
mReddit Summary : A person is considering purchasing a 1988 Mazda 323 with low mileage, but is hesitant due to the high asking price and potential maintenance costs. They would actually prefer a higher-mileage vehicle with a proven track record of maintenance. The initial asking price was $7,000, which was later reduced to $5,000, but most people agree that this is still too high. A more reasonable price for the vehicle would be between $2,000 to $3,500. The buyer is waiting to see if the price drops further and is seeking more information before making a decision.
ThreadSumm Summary : Japan produced many epic cars in the late 80s. Mazda 323 is a car model from the late 80s. The AZ-1 is another car model from Japan. Interestingly, a notable example of a well-preserved car from this era is the 1988 Mazda 323, which was found on Craigslist. The 1988 Mazda 323 has sub-30k miles on it and is in pretty much new condition. This remarkable condition can be attributed to the fact that the car has been maintained over the years by a little old lady, and the car has been driven only once a week for years. Despite its good condition, the car will need a complete service and timing belt replacement. The Mazda 323 was initially listed for nearly $7k, which is around the same price as an RX7. However, the listing price of the Mazda 323 has been lowered to $5k. In comparison, in Australia, a Mazda 323 in good condition would cost around $2 to $3 thousand. It's worth noting that some people prefer a well-maintained car with 200,000 miles over a low-mileage car.

Table 6: Example showing an input document from the Reddit dataset, the summaries generated using the vanilla prompt, mReddit method, and our method. The words highlighted in green are the key points included using our method, but omitted in either the vanilla or mReddit method.

	QAGS	SummaC	ROUGE-1
ThreadSumm	85.71	63.72	22.16
w/o Aspect Extraction	57.14	50.19	21.36
w/o Sentence Reordering	57.14	45.06	21.04
w/o Tree Of Thought	57.14	44.44	6.08

Table 7: Ablation study of the necessity of Aspect Extraction, Sentence Reordering and Tree of Thoughts

		QAGS(Δ)	SummaC(Δ)	R-1(Δ)
GPT4	r = 2, p =2	2.38	14.29	-4.56
	r = 1, p =1	9.79	-14.28	-3.12
	r = 2, p =1	1.62	42.86	-3.36
CLAUDE	r = 2, p =2	14.16	-14.28	2.72
	r = 1, p =1	13.71	14.29	2.4
	r = 2, p =1	12.49	-57.14	-2.64
LLAMA	r = 2, p =2	-11.54	0	-2.16
	r = 1, p =1	-10.73	14.28	-2.88
	r = 2, p =1	3.02	14.28	-2.72

Table 8: Delta observed in varying reordering proposals (r) and paragraph proposals (p) compared to r = 1, and p = 2 used in our experiments.

6 Conclusion

We introduce *ThreadSumm*, a multi-stage LLM pipeline framework consisting of Aspect extraction, ACU generation, Sentence ordering, Paragraph writing, Paragraph evaluation and selection using Tree of Thoughts for nested discourse summarization. Our method optimizes summary coherence and coverage of the source document without needing predefined thread order. We conduct extensive experiments across models and forum datasets, and our results show that our proposed method offers significant improvement compared to the baseline methods, while achieving higher aspect retention and opinion coverage in nested discussions. Since we focused on optimizing only coherence and coverage in this work, for future work, we will extend the Tree of Thoughts approach to optimize other dimensions of summary quality considering the quality-cost tradeoff. We will also explore how this method can be used in other areas of text summarization aside from discussion forums.

Limitations

We acknowledge certain limitations that may affect the performance and generalization of our method. Our method does not consider multimodal datasets, as we focused on datasets with textual data specifically English threads only. Also, our Tree of Thoughts search relies on LLM-based coherence and coverage scores, which could introduce potential biases and instability in the evaluation signal. We treat these scores as black box outputs and do not systematically calibrate or validate them against human judgments. Because the same LLMs are used both to generate summaries and to score them, our evaluation loop may inherit model-specific preferences. Furthermore, our discussions related to the performance of prompting of large language models in generating summaries of nested discourse threads are based solely on the summaries that were developed during this study and the summarization models that were adopted in our experiments.

Ethical Considerations

All the datasets and models we used in this paper are publicly available and accessible. The datasets used in this study are sourced from social forums which are known to contain offensive language; therefore, it is possible for the models to also generate summaries with offensive words.

Acknowledgments

We thank the anonymous reviewers as well as the members of PortNLP lab for their valuable feedback. This research was partially supported by the National Science Foundation grant CRII:RI 2246174.

References

Forrest Bao, Ruixuan Tu, Ge Luo, Yinfei Yang, Hebi Li, Minghui Qiu, Youbiao He, and Cen Chen. 2023. Docasref: An empirical study on repurposing reference-based summary quality metrics as reference-free metrics. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1226–1235.

Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2023. Booookscore: A systematic exploration of book-length summarization in the era of llms. *arXiv preprint arXiv:2310.00785*.

Alexander Richard Fabbri, Faiaz Rahman, Imad Rizvi, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir Radev. 2021. Convosumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6866–6880.

Alexander Richard Fabbri, Xiaojian Wu, Srini Iyer, Haoran Li, and Mona Diab. 2022. Answersumm: A manually-curated dataset and pipeline for answer summarization. In *Proceedings of the 2022 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 2508–2520.

Melika Golestani, Seyedeh Zahra Razavi, Zeinab Borhanifard, Farnaz Tahmasebian, and Hesham Faili. 2021. Using bert encoding and sentence-level language model for sentence ordering. In *International Conference on Text, Speech, and Dialogue*, pages 318–330. Springer.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Xiaobo Guo and Soroush Vosoughi. 2024. Modabs: Multi-objective learning for dynamic aspect-based summarization. *arXiv preprint arXiv:2406.03479*.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian document computing symposium*, pages 1–8.

Tatsuya Ishigaki, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. 2019. Discourse-aware hierarchical attention network for extractive single-document summarization. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 497–506, Varna, Bulgaria. INCOMA Ltd.

Hyuntak Kim and Byung-Hak Kim. 2025. NexusSum: Hierarchical LLM agents for long-form narrative summarization. *arXiv preprint arXiv:2505.24575*. Accepted to the main track of ACL 2025.

Jingun Kwon, Naoki Kobayashi, Hiroya Takamura, and Manabu Okumura. 2021. Considering nested tree structure in sentence extractive summarization with pre-trained transformer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4018–4030, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. Summac: Re-visiting nli-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yijun Liu, Frederick Choi, and Eshwar Chandrasekharan. 2025. Needling through the threads: A visualization tool for navigating threaded online discussions. *arXiv preprint arXiv:2506.11276*.

Yixin Liu, Alexander R Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shafiq Joty, Chien-Sheng Wu, Caiming Xiong, et al. 2022. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation. *arXiv preprint arXiv:2212.07981*.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2016. Sentence ordering using recurrent neural networks.

Jing Ma. 2025. Input order shapes llm semantic alignment in multi-document summarization. *arXiv preprint arXiv:2512.02665*.

Kawin Mayilvaghanan, Siddhant Gupta, and Ayush Kumar. 2025. Spot the blindspots: Systematic identification and quantification of fine-grained llm biases in contact center call summarization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1299–1340.

Olubusayo Olabisi and Ameeta Agrawal. 2024. Understanding position bias effects on fairness in social multi-document summarization. *arXiv preprint arXiv:2405.01790*.

Olubusayo Olabisi, Aaron Hudson, Antonie Jetter, and Ameeta Agrawal. 2022. Analyzing the dialect diversity in multi-document summaries. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6208–6221.

Keighley Overbay, Jaewoo Ahn, Fatemeh Pesaran Zadeh, Joonsuk Park, and Gunhee Kim. 2023. mredditsum: A multimodal abstractive summarization dataset of reddit threads with images. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4117–4132.

Xu Sun, Lionel Delphin-Poulat, Christèle Tarnec, and Anastasia Shimorina. 2025. Posum-bench: Benchmarking position bias in llm-based conversational summarization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7996–8020.

Suzan Verberne, Emiel Krahmer, Sander Wubben, and Antal van den Bosch. 2020. Query-based summarization of discussion threads. *Natural Language Engineering*, 26(1):3–29.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. *arXiv preprint arXiv:2004.04228*.

Weiqi Wu, Shen Huang, Yong Jiang, Pengjun Xie, Fei Huang, and Hai Zhao. 2025. Unfolding the headline: Iterative self-questioning for news retrieval and timeline summarization. *arXiv preprint arXiv:2501.00888*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. In *proceedings of the international AAAI conference on web and social media*, volume 11, pages 357–366.

Shiyue Zhang, Asli Celikyilmaz, Jianfeng Gao, and Mohit Bansal. 2021. Emailsum: Abstractive email thread summarization. *arXiv preprint arXiv:2107.14691*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

A Experimental Details

In this section, we provide more experimental details and LLM prompts used. The full algorithm for our method is in Algorithm 1. We set the temperature at 0 for all LLMs used in this study to ensure that the models generate a deterministic summary. All experiments were run using Nvidia A100 GPU. ThreadSumm introduces additional LLM calls for aspect extraction, ACU generation, sentence ordering, and Tree-of-Thoughts evaluation, which increases computational cost relative to single-pass prompting. On average, summarizing a single thread in our setting requires 5-8 LLM calls depending on the number of steps and proposal, resulting in higher cost of up to 1.5X vs Vanilla method, but yielding ROUGE improvements of up to 65%, particularly for complex nested discourse structures. For ToT hyperparameter selection, we selected 25 samples from the Reddit dataset and 12 samples from the Stack dataset for the preliminary pilot run, and we picked the configuration with the best mean evaluation performance for our experimental setup: number of steps = 3, number of reordering proposals = 1, and number of paragraph proposal = 2, chosen to balance summary quality and computational cost. We do not claim that our chosen depth and branching factors are globally optimal for all datasets or models.

A.1 Aspect Extraction

We use the prompt in Figure 5 across all models to extract all the key aspects in the Source Document.

Aspect Extraction Prompt

You are an expert at understanding documents. Extract the unique aspects (places, names, products, features, objects) mentioned in the following document : {Input}

Figure 5: Aspect Prompt

A.2 ACU Generation Prompt

We use the prompt in Figure 6 across all models to generate ACUs on all aspects extracted in the previous step. Human evaluation was done on a subset of the data to verify the faithfulness of Aspect Extraction and ACU generation, across different prompts.

ACU Generation Prompt

SystemPrompt : You are an expert at analyzing documents and breaking them down into atomic content units (ACUs). An ACU is the smallest self-contained statement of fact, claim, or idea that cannot be further broken down without losing its meaning.
UserPrompt : Follow this process step by step:
1. Read the document carefully.
2. Read the Aspect list carefully.
3. For each aspect, identify all distinct factual claims, propositions, or ideas.
4. Each ACU should express only one idea.
5. Each ACU should be independent of surrounding text.
6. Each ACU should be written in clear and concise language.
7. Rewrite each idea as a minimal, standalone statement.
8. Do not include reasoning steps or explanations, only the extracted statements.
9. Output the ACUs in a list, where each item is one ACU string.
10. Return only the file as output.

Figure 6: ACU Generation Prompt

A.3 Sentence Ordering

We use the prompt in Figure 7 to reorder the ACUs generated in the previous sentence to form a coherent flow.

Sentence Ordering Prompt

You are an expert at reordering documents for them to follow a logical and coherent flow. Ensure every sentence appears exactly once (none omitted or duplicated). Do not add new content. The reordered output should be a single continuous text, not an itemized list.

Figure 7: Sentence Ordering Prompt

A.4 Paragraph Writing

We use the prompt in Figure 8 across all models to summarize all the key points in the reordered sentences into a succinct paragraph.

Paragraph Writing Prompt

SystemPrompt: You are an expert at reading documents and summarizing the key points into one coherent paragraph.
UserPrompt: Given the following sentences, write a single coherent paragraph (not in bullet points or itemized. The first sentence of the paragraph must be the first sentence of the input document, and the last sentence of the paragraph must be the last sentence of the input document. Do not omit or duplicate any sentences. Use natural transitions and make it flow well.

Figure 8: Paragraph Writing Prompt

A.5 Iterative Evaluation

We use the prompt in Figure 9 to evaluate the paragraph candidates against the original text for its coherence and coverage.

Evaluation Prompt

SystemPrompt: You are an expert at evaluating text for coherence and coverage.
UserPrompt: "Given the original text (representing a set of core ideas) and a rewritten paragraph, score the paragraph for:
1. Coherence (logical flow, readability, and overall structure, ranging from 0.0 to 1.0)
2. Coverage (how completely it includes all key ideas/sentences from the original text, ranging from 0.0 to 1.0) Return the two scores as two numbers separated by a space (e.g., '0.9 1.0'). If the paragraph contains significantly fewer or more sentences than the original text, or if it changes the core meaning, score coverage lower. Do not provide any other text besides the two scores.

Figure 9: Evaluation Prompt

Algorithm 1 ThreadSumm

Input: Documents D
Output: Summary S
$p \leftarrow$ ConstructAspectPrompt(D)
raw $\leftarrow$ LLM$_{\text{FS}}(p, E)$
$A \leftarrow$ NormalizeAspects(raw)
$C \leftarrow \emptyset$
for each $a \in A$ **do**
  $p \leftarrow$ ConstructACUPrompt(D, a)
  raw $\leftarrow$ LLM$_{\text{FS}}(p, E)$
  $C \leftarrow C \cup$ ExtractACUs(raw)
end for
$C \leftarrow$ Deduplicate(C)
scores $\leftarrow$ ComputeCoherenceMatrix(C)
start $\leftarrow$ SelectMostCoherent$(C, scores)$
ord $\leftarrow$ start
while unplaced ACUs remain **do**
  next $\leftarrow$ SelectBestFollower(ord.last, C, $scores$)
  Append(ord, next)
end while
$P \leftarrow \emptyset$
for i = 1 to K **do**
  $p \leftarrow GenerateParagraph($
  start = ord[1],
  end = ord[last])
  $P \leftarrow P \cup p$
end for
for each paragraph p in P **do**
  $coh \leftarrow$ ComputeCoherence(p)
  $cov \leftarrow$ ComputeCoverage(p)
  score$[p] \leftarrow$ CombineScores(coh, cov)
end for
$S \leftarrow ArgMax$(score)
return S

B Annotation Guidelines

We provide the two English-speaking student annotators, familiar with community forums with clear guidelines on how to conduct evaluation of the samples, shown in Figure 10.

Annotation Guidelines

Given the following summaries, and the document, read the content and the summaries properly, and rate them on the following dimensions.

Coherence: On a scale of 1-5, is the summary well-structured, grammatical and easy to follow?
Coverage: On a scale of 1-5, how well does the summary cover all the points in the content provided?

Figure 10: Annotation Guidelines

C LLM-as-a-judge evaluation

We carried out additional analyses using LLM-as-a-judge with a different model "gpt-oss-20b". Our results in Table 9 show that even under this new metric, *ThreadSumm* outperforms the other methods on Coherence and Coverage, and mRedditSumm remains the most competitive baseline.

D Enhanced Vanilla Prompt

The vanilla baseline we used in the main experiment was kept simple to get a fair, reproducible baseline, and approximate how an off-the-shelf user would call the model. However, we run additional experiments with an enhanced vanilla prompt - *"You are an expert text summarizer tasked with providing summaries of nested multi-turn conversations, with multiple replies and quotes. Read through the documents, to understand the logical flow of the conversation. Please provide a summary of the text, ensuring coherence and coverage of all salient points. Do not itemize the summary. Return nothing else."* We report the results in Table 10, and note that while Enhanced Vanilla improves over the simple Vanilla method, our method still achieves superior performance relative to the enhanced baseline.

		Reddit			Stack			Bitcoin		
		QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1
Coherence	Vanilla	8.7	8.2	8.6	8.6	8.9	8.6	8	8	8
	arg-graph	8.2	7.8	8.4	7.6	8.7	6.7	6	8	7
	CHRONOS	8.6	8.4	8.2	8.5	7.2	8.6	8	9	8
	mRedditSumm	8.8	8	8.6	8.2	8.7	8.3	9	8	8
	ThreadSumm (ours)	**9.1**	**9.3**	**8.9**	**9.1**	**9.2**	**8.9**	**10**	**10**	**9**
Coverage	Vanilla	7.8	7.8	7.4	7.6	8.5	7.8	7	8	6
	arg-graph	7.6	7.8	7.2	8.4	7.9	7.4	7	7	7
	CHRONOS	8	7.6	7.8	8	8.8	8	7	8	7
	mRedditSumm	8.2	8	8.2	7.9	8.9	8.5	8	8	8
	ThreadSumm (ours)	**8.8**	**8.4**	**8.2**	**8.6**	**9**	**8.5**	**9**	**9**	**8**

Table 9: LLM-as-a-judge results comparing *ThreadSumm* and other baselines across three models, three datasets, and three evaluation metrics.

		Reddit			Stack			Bitcoin		
		QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1	QAGS	SummaC	ROUGE-1
GPT-4	EnhancedVanilla	38.54	30.26	31.09	37.92	**38.34**	33.91	71.43	49.20	6.29
	Vanilla	36.46	30.16	30.54	33.46	35.34	32.24	71.43	48.17	5.44
	arg-graph	38.46	30.18	27.11	34.55	34.44	29.22	**85.71**	57.86	4.96
	CHRONOS	41.49	30.31	28.59	33.82	35.23	31.62	71.43	46.26	6.00
	mRedditSumm	38.23	**30.76**	29.55	34.55	35.10	31.84	28.57	**61.08**	5.92
	ThreadSumm (ours)	**50.34**	30.40	**33.30**	**49.94**	33.46	**35.89**	57.14	40.07	**15.28**
CLAUDE	EnhancedVanilla	40.44	31.76	32.93	44.65	39.13	34.42	57.14	48.39	5.61
	Vanilla	38.34	29.51	30.88	40.05	36.79	31.86	28.57	45.54	4.56
	arg-graph	40.91	32.01	30.19	42.61	37.07	30.48	28.57	57.44	6.72
	CHRONOS	45.43	32.11	26.35	41.64	33.56	27.76	71.43	**67.73**	4.96
	mRedditSumm	45.66	30.64	26.82	39.93	34.87	29.22	57.14	62.28	5.20
	ThreadSumm (ours)	**55.66**	**42.42**	**34.37**	**57.75**	**50.29**	**39.29**	**85.71**	63.72	**22.16**
LLAMA	EnhancedVanilla	35.87	30.59	**32.35**	34.77	36.42	33.53	57.14	44.28	6.26
	Vanilla	33.20	28.66	30.53	32.84	32.46	32.69	57.14	41.33	6.08
	arg-graph	35.49	30.93	24.21	36.75	38.97	27.35	28.57	49.80	6.16
	CHRONOS	43.09	28.82	27.10	31.50	36.41	28.82	42.86	31.67	5.20
	mRedditSumm	40.34	30.34	30.37	37.36	35.00	31.96	28.57	45.52	6.24
	ThreadSumm (ours)	**50.00**	**33.98**	25.25	**51.28**	**42.44**	**38.39**	**71.43**	**59.36**	**20.08**

Table 10: Results comparing *ThreadSumm* and other baselines across three models, three datasets, and three evaluation metrics, including enhanced vanilla prompt.

E Analysis of sentences not represented in summary

In this section, we analyze the sentences in the Bitcoin source document that are not represented in the summary. A sentence is considered represented if it is greater than or equal to the dynamic threshold which is calculated based on the statistical properties of each conversation-summary pair. Table 11 shows the *ThreadSumm* summary, as well as the index and corresponding text not represented in the summary. It is obvious that the underrepresented text does not improve or diminish the summary content in any way, indicating that our method does not show position bias.

F Aspect Coverage on Reddit and Stack Datasets

We plot the Aspect Coverage on Reddit and Stack datasets in Figure 11, and Figure 12, and we see that our method mostly outperforms the baseline methods. Claude also has the best performance on both datasets as observed in the Bitcoin datset.

G Ablation on Reddit and Stack

We provide further results in Table 12 and Table 13, which match the same findings observed on the Bitcoin dataset that Tree of Thoughts is the single

ThreadSumm Summary
The discussion of anonymous payment methods begins with the consideration of cash, which is generally considered the most anonymous way to pay. Cash is considered the best anonymous payment method, but it has its own set of limitations. However, paying with cash can be more anonymous than paying with Bitcoin, but in some cases, Bitcoin can be more anonymous than cash. Large volumes of cash cannot be stored or sent easily, and banks may ask questions or refuse to release large amounts of cash. Using cash can leave fingerprints unless gloves are worn, and the CIA may investigate cash transactions for forensic analysis, although serial numbers on cash do not serve the purpose of tracing ownership. In contrast to cash, Bitcoin transactions have their own set of characteristics. The blockchain is a permanent record of all Bitcoin transactions. Bitcoin transactions can be traced, but ownership cannot be proven. However, Bitcoins can be mixed and re-mixed to prevent the NSA from identifying a Bitcoin holder, and the NSA can see who has a significant amount of Bitcoins. Paying with mixed Bitcoins can be more anonymous than paying with cash. Using a VPN can make Bitcoin transactions more anonymous. Bitcoin transactions are more anonymous online than in person. The use of Bitcoin for various transactions is also noteworthy. Bitcoins can be used for all transactions, regardless of the amount. Using Bitcoin for small transactions contributes to the freedom of the currency. The blockchain can be used for small transactions without 'polluting' it. Face to face transactions of physical goods can only be traced by the other person seeing and remembering the buyer's face. Phone and email records can be traced, but Trac phones and certain emails cannot be traced. Computers always leave digital traces. Ultimately, the most anonymous transaction is the one that utilizes complete online anonymity measures. The only totally anonymous transaction is for digital goods bought online using Bitcoin with complete online anonymity measures. Buying Bitcoin anonymously is possible but often requires more effort than buying cash. Some exchanges, like btc-e, do not require identification to buy Bitcoin. Cash is generally considered the most anonymous way to pay, but Bitcoin can offer a high level of anonymity when used correctly.
Posts and Index from Source Document not represented in the summary
#12: ;D. Of course it is. #13: But unless it's lying around you have to pull it out of your account and if it's a large amount that's pretty tricky ... #18: It just depends!. #22: Whoever you give it to is likely to spend it fast, so most of the time you don't have to worry. #47: Absolutely it is. #52: I'm always surprised when I read that some people want to use BTC to buy a soda.

Table 11: Analysis of sentences not represented

most impactful component in our pipeline, yielding the largest overall gains, and in Table 14 and Table 15 showing that different evaluation metrics may favor different regions of the ToT hyperparameter space.

H Disclosure of use of AI tools

Portions of this manuscript were prepared with the assistance of AI tools for editing, and research on related work (OpenAI ChatGPT, Perplexity). These tools were used to improve sentence clarity, while all ideas, analysis, and conclusions are the authors' own.

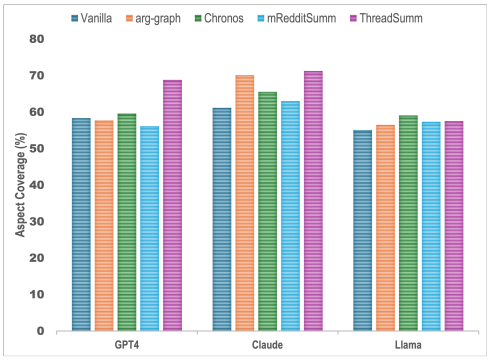

Figure 11: Aspect Coverage of the summary relative to the source document on Reddit dataset.

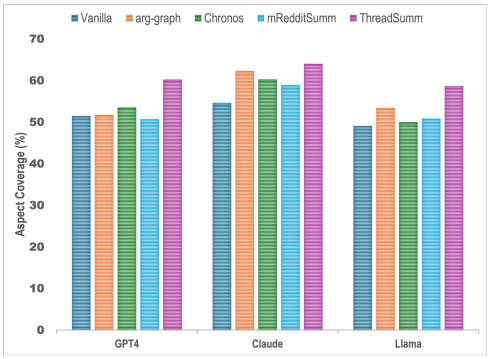

Figure 12: Aspect Coverage of the summary relative to the source document on Stack dataset.

	QAGS	SummaC	ROUGE-1
ThreadSumm	55.66	42.42	34.37
w/o Aspect Extraction	48.19	31.06	30.12
w/o Sentence Reordering	48.02	30.24	27.46
w/o Tree Of Thought	39.53	28.36	25.15

Table 12: Ablation study of the necessity of Aspect Extraction, Sentence Reordering and Tree of Thoughts on Reddit dataset

	QAGS	SummaC	ROUGE-1
ThreadSumm	57.75	50.29	39.29
w/o Aspect Extraction	41.19	40.81	35.46
w/o Sentence Reordering	40.63	37.22	33.75
w/o Tree Of Thought	38.08	33.14	30.21

Table 13: Ablation study of the necessity of Aspect Extraction, Sentence Reordering and Tree of Thoughts on Stack dataset

		QAGS(Δ)	SummaC(Δ)	R-1(Δ)
GPT4	r = 2, p =2	12.67	-8.42	-3.87
	r = 1, p =1	14.10	-10.65	-5.18
	r = 2, p =1	10.22	11.29	-3.77
CLAUDE	r = 2, p =2	11.61	-8.49	2.91
	r = 1, p =1	12.39	9.34	-2.53
	r = 2, p =1	10.06	-10.40	-2.74
LLAMA	r = 2, p =2	13.72	-11.26	3.98
	r = 1, p =1	10.12	8.31	-2.05
	r = 2, p =1	10.88	9.42	2.28

Table 14: Delta observed in varying reordering proposals (r) and paragraph proposals (p) compared to r = 1, and p = 2 used in our experiments on Reddit dataset.

		QAGS(Δ)	SummaC(Δ)	R-1(Δ)
GPT4	r = 2, p =2	11.42	9.26	-2.91
	r = 1, p =1	8.94	7.12	-4.06
	r = 2, p =1	-10.32	-10.74	-4.17
CLAUDE	r = 2, p =2	8.63	10.29	-3.96
	r = 1, p =1	8.42	-8.16	2.02
	r = 2, p =1	-9.37	10.67	-2.81
LLAMA	r = 2, p =2	9.54	7.42	-2.21
	r = 1, p =1	-5.62	10.19	-2.64
	r = 2, p =1	-8.71	9.23	-2.11

Table 15: Delta observed in varying reordering proposals (r) and paragraph proposals (p) compared to r = 1, and p = 2 used in our experiments on Stack dataset.