DEA-TLS: A Divide-Extract-and-Aggregate Framework for Costless and Accurate Timeline Summarization

Ming Qiao[1,2,3,4,5], Yinlong Xiao[4,5], Minghao Hu[4,5(✉)], Jianyong Duan[1,2,3], Xin Li[1,2,3], Zhunchen Luo[4,5], Shuai Lei[4,5], Yunbo Cao[4,5] and Guotong Geng[4,5]

[1]School of Artificial Intelligence and Computer Science, North China University of Technology, Beijing 100144, China
[2]Beijing Key Laboratory of Key Technologies for AI+ Domain Applications, Beijing 100144, China
[3]CNONIX National Standard Application and Promotion Lab, Beijing 100144,China
[4] Center of Information Research, AMS, Beijing 100142, China
[5]Discipline and Technology Research Center for Large Model Intelligence Application, Beijing 100142, China
humh573@163.com

Abstract. Timeline summarization aims to identify key events from multi-source texts and organize them into a coherent timeline of event evolution in chronological order. Existing methods generally adopt a two-stage document-level framework, which first extracts timeline information from individual documents and then aggregates the extracted results into a final timeline summary.However, such methods often suffer from high computational cost in practical applications. On the one hand, closed-source models incur substantial inference expenses; on the other hand, locally deployed open-source models still require considerable computational resources when processing long documents. To address this issue, we propose DEA-TLS, a Divide-Extract-and-Aggregate framework for Timeline Summarization,which transforms conventional document-level processing into finer-grained paragraph-level processing, thereby reducing the overall computational burden. Nevertheless, this framework also brings new challenges, including the introduction of irrelevant information, inaccurate event extraction, and redundancy during aggregation. To tackle these issues, we further design three modules: Progressive Divider, Reflective Extractor, and Hierarchical Aggregator, which are responsible for selecting high-value paragraphs, improving extraction accuracy, and reducing semantic redundancy, respectively.Experiments on the Open-TLS dataset demonstrate that our method outperforms existing baselines across multiple key metrics, providing an effective solution for achieving low-cost and high-accuracy timeline summarization.

Keywords: Timeline Summarization, Large Language Models,Retrieval-Augmented Generation,Self-Reflection.

1 Introduction

Timeline summarization (TLS) aims to identify key events related to a given topic from multi-source unstructured texts and organize them into a coherent process of event evolution in chronological order [18]. Current TLS research is generally divided into open-domain and closed-domain settings[16]. Open-domain TLS focuses on generating timelines from news documents directly retrieved from the Internet. Early studies mainly adopted extractive methods,constructing timelines by selecting representative content from candidate texts based on topic relevance, temporal expression identification, and sentence importance scoring [1,5,19].Later,some studies introduced neural models to improve timeline coherence and information coverage through neural generation or structured event modeling[3]. More recently, with the rapid development of large language models, research has gradually shifted toward retrieval-augmented generation frameworks, which enable models to generate more coherent and narrative timeline summaries based on retrieved multi-source documents [6,14].

However, existing large language model-based timeline summarization methods still generally follow a two-stage document-level framework, in which timeline information is first extracted from individual documents and then aggregated into a final timeline summary.In practice,this framework often incurs substantial computational cost. On the one hand, approaches based on closed-source models usually involve high inference expenses. On the other hand, when locally deployed open-source models are used, document-level methods often require considerable GPU memory and computational resources to process long texts across multiple documents, which limits their applicability in resource-constrained settings. To address this issue, we propose **DEA-TLS**,a Divide-Extract-and-Aggregate framework for Timeline Summarization. This framework transforms conventional document-level processing into finer-grained paragraph-level processing, reducing the input size of each inference step and thereby lowering computational resource consumption, thus providing a foundation for low-cost timeline summarization with locally deployed open-source models.

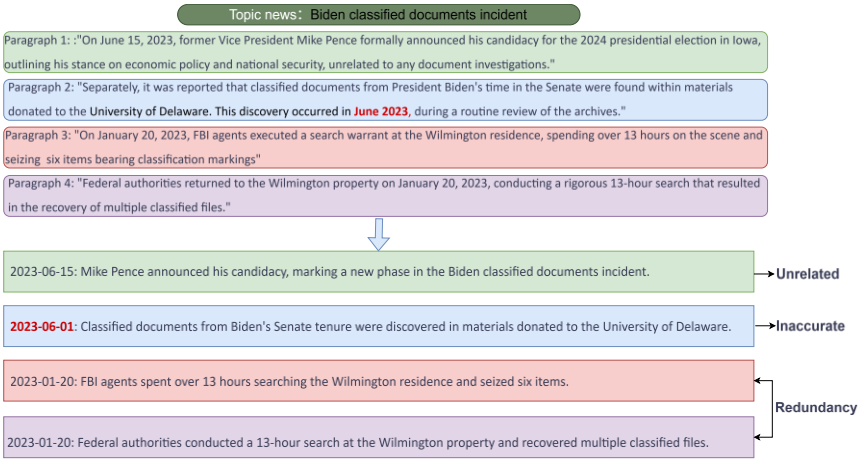

Fig. 1. Three major challenges in paragraph-level timeline summarization: the inclusion of topic-irrelevant paragraphs, inaccurate event extraction, and semantically redundant events.

Nevertheless, as illustrated in Fig. 1,this framework still faces several challenges.First, the model may retain paragraphs that contribute little to timeline generation for the target topic, leading to topic-irrelevant content (refer to the green part). Second,during the event extraction stage, model hallucination may result in inaccurate event extraction (refer to the blue part). Third, during the aggregation stage, because isolated paragraphs lack global context, the framework may produce semantically overlapping candidate events (refer to the red and purple parts), which introduces redundancy into the final summary. Therefore, how to preserve the low-cost advantage of the paragraphlevel framework while alleviating its limitations in relevance, accuracy, and redundancy control remains an important research problem in open-domain timeline summarization.

To address these challenges, we further design three collaborative modules within DEA-TLS. First, **Progressive Divider** adopts a two-stage selection strategy that combines semantic coarse selection with large language model-based fine-grained refinement to select paragraphs that are highly contributive to the target topic and reduce the input context. Second, **Reflective Extractor** extracts structured events from candidate paragraphs and introduces a timeline cue-aware self-reflection mechanism, which corrects extraction errors and supplements omitted events, thereby improving the accuracy of the extracted event set. Finally, **Hierarchical Aggregator** performs aggregation at both the event level and the temporal level, deduplicating and compressing semantically redundant events, reducing the interference of redundant candidates in downstream timeline construction, and improving temporal coherence.

Our contributions are summarized as follows:

--We propose **DEA-TLS**, a Divide-Extract-and-Aggregate framework for Timeline Summarization, which transforms document-level processing into finer-grained paragraph-level processing, thereby supporting low-cost timeline summarization with locally deployed open-source models.

--We design three collaborative modules to address the key limitations of the paragraph-level framework in timeline summarization: **Progressive Divider** selects high-value paragraphs via a two-stage selection strategy, **Reflective Extractor** improves event extraction accuracy through a self-reflection mechanism, and **Hierarchical Aggregator** reduces semantic redundancy and improves temporal organization through event-level and time-level aggregation.

--Experimental results on the Open-TLS dataset demonstrate that DEA-TLS effectively improves the overall performance of timeline summarization, providing an effective solution for achieving low-cost and high-accuracy timeline summarization.

2 Related Work

2.1 Timeline Summarization

Timeline summarization aims to identify topic-relevant key events from multi-source

texts and organize them into a coherent chronological account of event evolution [18]. Early studies mainly adopted extractive methods, combining topic relevance and temporal features to select representative content from candidate sentences or passages for timeline construction [1,19]. With the development of large language models, research has gradually shifted toward generative frameworks, which enable more flexible cross-document information fusion and narrative organization[6,12].For example,CHRONOS decomposes retrieval and generation through iterative self-questioning,thereby improving coverage and timeline construction quality in open-domain settings[16]. Subsequent work has further advanced the field from the perspectives of granularity control and structured modeling. DTELS studies dynamic granularity strategies for timeline construction [20], while TH-VAE improves timeline summarization in specific domains through structured modeling [13]. However, existing open-domain timeline summarization methods still generally rely on feeding one or multiple full documents into the model at once. In long-document and multi-document scenarios,this design introduces substantial computational and memory overhead, and overly long contexts may also weaken the model's focus on salient temporal cues, thereby increasing the risk of temporal factual errors and missing key information.

2.2 Retrieval-Augmented Generation and Self-Reflection

Retrieval-augmented generation improves large language models by introducing external knowledge during inference, which helps mitigate hallucination and outdated knowledge issues [7,8]. It has therefore been widely applied to many tasks, including timeline summarization. However, existing systems often follow a one-way retrieve-then-generate pipeline and are highly sensitive to the quality of the initial retrieval results. Once the retrieved evidence is noisy or lacks key support, the downstream generation stage can hardly correct errors dynamically. Recent studies have shown that reflection mechanisms such as self-evaluation and self-verification can improve the reliability of large language models on complex reasoning tasks [15,17]. This insight is directly relevant to timeline summarization, because the key risks of this task are often not whether the generated text is fluent, but whether events are supported by evidence, whether temporal anchors are accurate, and whether important events are omitted. Therefore, forming a closed loop of generation, verification, and revision and integrating it into the extraction process is a promising direction for improving the accuracy of timeline summaries.

3 Method

We present DEA-TLS, a divide-extract-and-aggregate framework for timeline summarization that replaces document-level processing with paragraph-level processing to support low-cost inference. Unlike document-level methods that directly process long texts across multiple documents, DEA-TLS adopts a staged modeling strategy that decomposes timeline construction into three phases: division, extraction, and aggregation,as illustrated in Fig. 2. Specifically, given an input topic, the framework first builds

a candidate document set through initial news retrieval, few-shot-guided self-questioning,and query rewriting to recover missing stages of event evolution as completely as possible [16]. The resulting documents are then processed by three collaborative modules.Progressive Divider segments candidate documents into para-graphs and selects high-value paragraphs, thereby reducing noise interference and compressing the input size.Reflective Extractor performs structured event extraction and reflective correction on the filtered paragraphs, improving the accuracy of can-didate events. Hierarchical Aggregator conducts unified deduplication and compres-sion of candidate events along the event dimension and organizes them along the temporal dimension, ultimately generating a compact and chronologically coherent timeline summary.Algorithm 1 outlines this process.Sections 3.1, 3.2, and 3.3 detail these components, respectively.

Algorithm 1 DEA-TLS

Input: Target topic T, Documentset $\mathcal{D}$
Output: Timeline summary Y
$\mathcal{P} \leftarrow$ Segment Into Paragraphs$(\mathcal{D})$
$\mathcal{P}^* \leftarrow$ Progressive Divider$(\mathrm{T}, \mathcal{P})$ ▷ Eq. $(4) - (6)$
$\mathcal{E}^* \leftarrow \emptyset$
for each p $\in \mathcal{P}^*$ **do**
$\mathcal{E}^{(0)} \leftarrow$ LLM_Extract(p)
$\mathcal{E}_{\mathrm{p}} \leftarrow$ Reflective Extractor$(\mathcal{E}^{(0)}, \mathrm{p})$ ▷ Eq. $(10) - (11)$
$\mathcal{E}^* \leftarrow \mathcal{E}^* \cup \mathcal{E}_{\mathrm{p}}$
end for
$\mathcal{R} \leftarrow$ Event Level Aggregation$(\mathcal{E}^*)$ ▷ Semantic deduplication
for each time unit u in sort$(\mathcal{R})$ **do**
Merge events with similarity $\geq \gamma$
Keep one per cluster
end for
Y $\leftarrow$ Temporal Level Aggregation$(\mathrm{T}, \mathcal{R})$ ▷ Eq. (16)
return Y

3.1 Progressive Divider

Open-domain timeline summarization usually involves multiple long documents related to the same topic. Direct document-level modeling not only incurs high inference cost,but also introduces large amounts of weakly relevant text into the model context, which can degrade downstream event extraction and timeline construction. To address this issue, Progressive Divider adopts a two-stage strategy that first performs semantic coarse selection and then applies large language model-based fine-grained refinement, thereby reducing noise at the input stage and providing higher-quality evidence for subsequent processing.

Before entering Progressive Divider, the framework constructs a candidate document set around the input topic T. Specifically, the framework uses T as the initial retrieval query to collect directly related news reports. It then combines the retrieved content with a few-shot prompting strategy to let the large language model generate information seeking questions that the currently retrieved articles cannot answer but are useful for recovering missing stages of event evolution. Complex questions are further rewritten into simpler retrieval-oriented queries, which are used to expand the pool of candidate documents. Through the above process, the framework obtains a set of candidate documents relevant to the current query. Let the document collection retrieved for the query q be

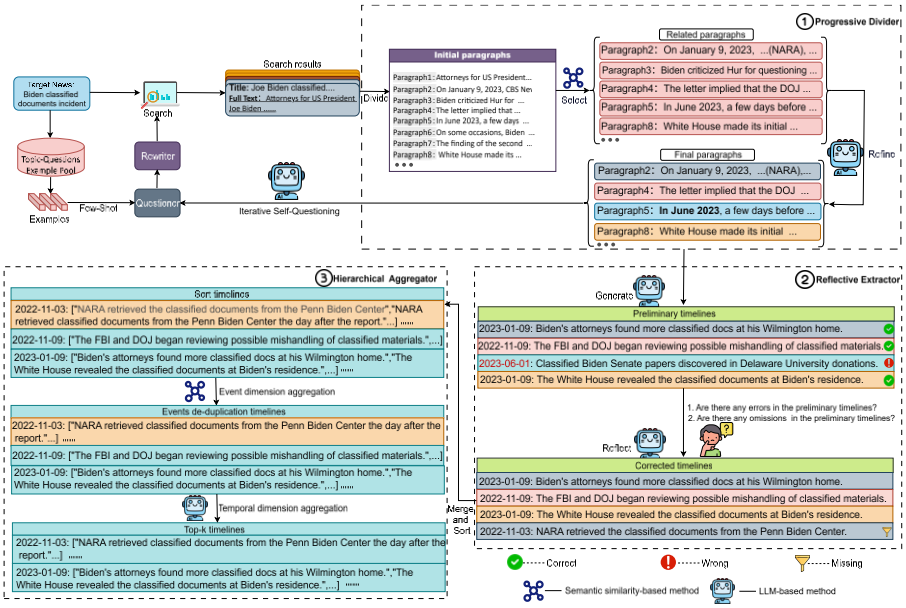

Fig. 2. Overview of the DEA-TLS framework. Given a topic query, the framework first retrieves relevant news articles and applies an iterative self-questioning strategy to retrieve more comprehensive related news. It then applies three dedicated modules: a) Progressive Divider divides the documents into paragraphs, selects topic-relevant and high-value paragraphs via a two-stage selection strategy; b) Reflective Extractor improves event extraction accuracy through reflective revision; and c) Hierarchical Aggregator performs event-level and time-level aggregation to reduce semantic redundancy and produce a compact, chronologically coherent timeline summary.

$$\mathcal{D}(q) = \{d_1, d_2, \ldots, d_N\}, \tag{1}$$

where $q = \mathrm{T}$ in the initial stage.

Each document is then segmented into paragraph units, yielding a set of candidate paragraphs

$$\mathcal{P} = \{p_1, p_2, \ldots, p_M\}. \tag{2}$$

Next, we encode the query q and each paragraph p_i with a lightweight semantic embedding model (bge), obtaining representations e_q and e_{p_i}. We use cosine similarity to estimate the relevance of the paragraph:

$$r_i = sim(q, p_i) = \frac{e_q^\top e_{p_i}}{\|e_q\| \|e_{p_i}\|}. \tag{3}$$

Based on the relevance scores, we first rank all paragraphs and then keep only the paragraphs that are both among the top-ranked candidates and above a relevance threshold τ ,producing the coarse candidate set

$$\mathcal{P}_c = \{p_i \mid p_i \in TopK(\mathcal{P}, r),\ r_i \geq \tau\}. \tag{4}$$

However, semantic similarity alone is insufficient for identifying paragraphs that truly contribute to timeline construction, because highly similar paragraphs may still lack explicit temporal expressions, salient event development, or essential supporting facts. We therefore introduce a second-stage large language model judge. For each candidate paragraph $p \in \mathcal{P}_c$,we define a binary selection function

$$z(p, T) = \begin{cases} 1, if\ p\ makes\ a\ substantive\ contribution \\ \quad to\ constructing\ the\ timeline\ of\ T, \\ 0, otherwise. \end{cases} \tag{5}$$

The final retained paragraph set is

$$\mathcal{P}^* = \{p \mid p \in \mathcal{P}_c, z(p, T) = 1\}. \tag{6}$$

Through this progressive recall-then-judge design, Progressive Divider effectively preserves paragraphs that are topic-relevant and valuable for timeline construction.

3.2 Reflective Extractor

After Progressive Divider, the framework obtains a set of paragraphs that are both topic-relevant and potentially informative for timeline construction. However, owing to the inherent hallucination tendency of large language models, direct event extraction remains prone to two critical issues: the model may generate event candidates that are not fully grounded in the source text, or it may fail to capture all salient events expressed in the selected paragraphs. To mitigate these issues, Reflective Extractor introduces a structured process comprising initial extraction, reflection-based review, and subsequent correction, thereby improving the factual accuracy of the extracted events.

Formally, let the paragraph set returned by Progressive Divider be

$$\mathcal{P}^* = \{p_1, p_2, \dots, p_{|\mathcal{P}^*|}\}. \tag{7}$$

For each paragraph p_i, Reflective Extractor first prompts the large language model to generate an initial candidate event set

$$\mathcal{E}_i^{(0)} = \{e_1, e_2, \dots, e_{n_i}\}. \tag{8}$$

Each candidate event is represented as a structured triplet

$$e = (t, m, c), \tag{9}$$

where t denotes the temporal information of the event,m denotes the event summary, and c denotes the paragraph identifier from which the event is extracted. This representation explicitly preserves both temporal anchors and semantic content, facilitating later verification and aggregation.

Although large language models are strong generators, their outputs are still susceptible to hallucination and may not always remain tightly grounded in source evidence;therefore, the initial event set $\mathcal{E}_i^{(0)}$may contain unsupported or incomplete outputs [4].We therefore introduce a reflection stage that checks candidate events against their source paragraph. For an event e and its source paragraph p_i, we define a verification function

$$v(e, p_i) = \begin{cases} 1, & if\ the\ temporal\ information\ and \\ & event\ summary\ of\ e\ are\ supported\ by\ p_i, \\ 0, & otherwise. \end{cases} \tag{10}$$

Based on this evidence constraint, the final event set for paragraph p_i is obtained through a unified refinement operator:

$$\mathcal{E}_i^* = \mathrm{Refine}(\mathcal{E}_i^{(0)}, p_i). \tag{11}$$

In practice, this refinement step jointly corrects unsupported events, removes candidates that remain ungrounded after review, and supplements important events that were missed during the initial extraction. After refinement over all paragraphs, the global candidate event pool is formed as

$$\mathcal{E}^* = \bigcup_i \mathcal{E}_i^* . \tag{12}$$

Reflective Extractor therefore extends conventional one-pass extraction into an explicit review-and-refinement process that reduces error propagation before global timeline aggregation.

3.3 Hierarchical Aggregator

Although the event pool produced by Reflective Extractor is more reliable, in multi-document news scenarios, the lack of global information for cross-document event consolidation in paragraph-level processing may still cause the same event to be retained as multiple semantically overlapping candidate events, thereby introducing redundancy.To address this issue, Hierarchical Aggregator first performs event-level deduplication and compression over candidate events, and then conducts time-level organization over the reduced candidate space, thereby reducing redundancy and optimizing the timeline structure.

Let the candidate event pool returned by Reflective Extractor be

$$\mathcal{E}^* = \{e_1, e_2, \dots, e_L\}. \tag{13}$$

We first sort all events according to their temporal information and group events with the same date or within the same temporal window into a shared time unit. For any two events e_i and e_j in the same unit, we compute the semantic similarity between their event summaries:

$$s(m_i, m_j) = \mathrm{sim}(m_i, m_j). \tag{14}$$

If the similarity exceeds a threshold γ, the two events are treated as belonging to the same event cluster. Through this process, repeated reports of the same event are compressed into representative semantic clusters, and only the most informative event in each cluster is retained.

During this event-dimension aggregation stage, the paragraph identifier c is retained to preserve traceability to the original paragraph evidence. Once this stage is completed, c is discarded, and each retained event is simplified from $e = (t, m, c)$ to $e' = (t, m)$ for subsequent temporal-dimension aggregation.

Let the compressed candidate set after event-dimension aggregation be denoted as

$$\mathcal{R} = \{e'_1, e'_2, \dots, e'_M\}, \quad M \le L. \tag{15}$$

After event-dimension aggregation, the framework proceeds to temporal-dimension aggregation. The compressed event candidates are then fed into a large language model,which evaluates their global importance under the input topic T and selects the top-k most representative timeline entries, yielding the final timeline summary

$$Y = TopK(G_\theta(T, \mathcal{R}), k). \tag{16}$$

Here, G_θ denotes the large language model, and Y is the final timeline summary constructed from the selected entries. Compared with direct global selection over the raw event pool, this hierarchical strategy substantially reduces input redundancy and enables more efficient and coherent timeline construction.

4 Experiments

4.1 Experimental Setup

Dataset. We conduct experiments on the Open-TLS benchmark [16], a standard evaluation dataset for open-domain timeline summarization. Table 1 reports the main statistics of the dataset, including the numbers of topics and timelines in each domain, the average duration of timelines(days), the average number of dates per timeline (Avg. l), and the average number of event sentences per date (Avg. k).

Evaluation Metrics. We report ROUGE-N and Date F1 following prior work on timeline summarization [10,16]. In particular, ROUGE is computed under three evaluation views, namely Concat-F1, Agree-F1, and Align-F1, which respectively assess overall content overlap, strict date matching,and alignment-conditioned content consistency.We additionally report Date F1 to measure how accurately the generated timelines recover key temporal anchors.

Baselines. To evaluate the effectiveness of DEA-TLS, we compare with the three representative strategies introduced in CHRONOS[16]: DIRECT, which retrieves documents from the original query and directly generates a timeline; REWRITE, which first rewrites the query and then generates a timeline from the expanded retrieval results; and CHRONOS, which iteratively performs self-questioning and retrieval to progressively construct a timeline.

Table 1. Statistics of the Open-TLS dataset.

Statistic	Overall	Politics	Society	Economy	Sport	Technology
#Topics	50	25	12	5	5	3
#Timelines	50	25	12	5	5	3
Avg. duration	4139	4624	1719	1297	8219	7694
Avg. l	23	25	19	22	20	20
Avg. k	1.8	1.8	2.1	1.7	1.8	1.6

4.2 Main Results

We use GPT-4o [11] and Qwen2.5-72B [2] as the core inference engines, representing strong closed-source and open-source large language model settings, respectively.Table 2 presents the main results on Open-TLS.

Table 2. Main results on Open-TLS(iterative search rounds = 5). Best results within each model block are shown in bold.

Model	Method	Concat F1		Agree F1		Align F1		Date
		R-1	R-2	R-1	R-2	R-1	R-2	F1
GPT-4o	DIRECT	0.243	0.063	0.056	0.021	0.071	0.025	0.208
	REWRITE	0.233	0.067	0.054	0.022	0.070	0.026	0.205
	CHRONOS	0.351	0.103	0.105	0.047	0.121	0.051	0.343
	DEA-TLS	**0.389**	**0.133**	**0.148**	**0.074**	**0.153**	**0.075**	**0.402**
Qwen2.5-72B	DIRECT	0.328	0.101	0.087	0.044	0.104	0.049	0.265
	REWRITE	0.337	0.106	0.091	0.046	0.107	0.050	0.291
	CHRONOS	0.368	0.110	0.106	0.049	0.125	0.050	0.324
	DEA-TLS	**0.399**	**0.138**	**0.149**	**0.081**	**0.154**	**0.082**	**0.423**

Overall, DEA-TLS achieves the best performance among all compared methods under both GPT-4o and Qwen2.5-72B settings. Compared with the baseline method CHRONOS, DEA-TLS obtains higher scores on Concat F1, Agree F1, Align F1, and

Date F1 across both settings. In particular, the improvement on Date F1 is especially notable, increasing from 0.343 to 0.402 under GPT-4o and from 0.324 to 0.423 under Qwen2.5-72B. These results suggest that the proposed framework not only helps improve lexical overlap with the reference timelines, but also contributes to better temporal alignment and more accurate date prediction.

These findings further provide some support for the effectiveness of the three-module design in DEA-TLS. Specifically, Progressive Divider reduces weakly relevant and noisy paragraph evidence before extraction; Reflective Extractor improves the accuracy of candidate timeline extraction through review and refinement; and Hierarchical Aggregator mitigates semantic redundancy while organizing events into a more coherent temporal structure. In addition, the framework maintains relatively stable advantages under both closed-source and open-source model settings, suggesting that it exhibits a certain degree of adaptability across different large language model backbones.

4.3 Ablation Study

To validate the contribution of each module, we conduct an ablation study on Open-TLS using Qwen2.5-72B as the backbone model. To keep the computational cost manageable while ensuring internal fairness, all variants in the ablation study run with 3 iterative search rounds, in contrast to the 5 rounds used in the main experiments. Table 3 reports the results after removing the Aggregator, Extractor, and Divider, respectively.

Table 3. Ablation results on Open-TLS using Qwen2.5-72B(iterative search rounds = 3).

Variant	Concat F1		Agree F1		Align F1		Date F1
	R-1	R-2	R-1	R-2	R-1	R-2	
w/o Aggregator	0.263	0.064	0.085	0.028	0.089	0.027	0.327
w/o Extractor	0.332	0.092	0.096	0.029	0.100	0.027	0.349
w/o Divider	0.306	0.089	0.098	0.027	0.102	0.029	0.343
Ours	**0.347**	**0.095**	**0.104**	**0.032**	**0.106**	**0.031**	**0.352**

Removing Removing any module degrades performance, which confirms that the three components contribute complementary benefits. In particular, removing Divider decreases Concat F1 (R-1) from 0.347 to 0.306 and Date F1 from 0.352 to 0.343, showing that paragraph-level progressive filtering is important for suppressing noisy input and improving temporal anchoring. Removing Extractor lowers Concat F1 (R-1) and Date F1 to 0.332 and 0.349, respectively, indicating that reflection-based extraction effectively reduces event errors and omissions. The largest degradation appears when Aggregator is removed: Concat F1 (R-1) drops to 0.263 and Date F1 drops to 0.327, which shows that event-level semantic deduplication is crucial for reducing redundant candidates before time-level selection.

4.4 Analysis

Cost Efficiency. We further analyze DEA-TLS from the perspective of computational efficiency. Specifically, we study both the proportion of very long documents and the input length reduction brought by paragraph-level modeling on five topics: Iraq War,Brexit, Grenfell Tower Fire, The Search for MH370, and Acquisition of Twitter. To align the analysis with model-side input length, we tokenize documents using the same tokenizer as the inference pipeline and compute the number of tokens for each document.

As shown in the left panel of Fig. 3, under document-level processing the proportions of documents longer than 20,000 tokens are 34.6%, 22.6%, 30.6%, 24.0%, and 50.0% across the five topics, and the proportions longer than 50,000 tokens are 26.9%,16.1%, 25.0%, 16.0%, and 33.3%. This shows that open-domain timeline summarization often involves a substantial share of extremely long inputs under document-level modeling, resulting in high GPU memory use and inference cost. The right panel further compares input lengths between document-level processing and DEA-TLS. Traditional document-level TLS directly processes full documents, with an average input length of 10,945.78 tokens and a maximum of 205,716 tokens. By first retaining topic-relevant key paragraphs via Progressive Divider, DEA-TLS reduces the average input length to 434.69 tokens and the maximum to 1,682 tokens, showing clear efficiency advantages in resource-constrained settings.

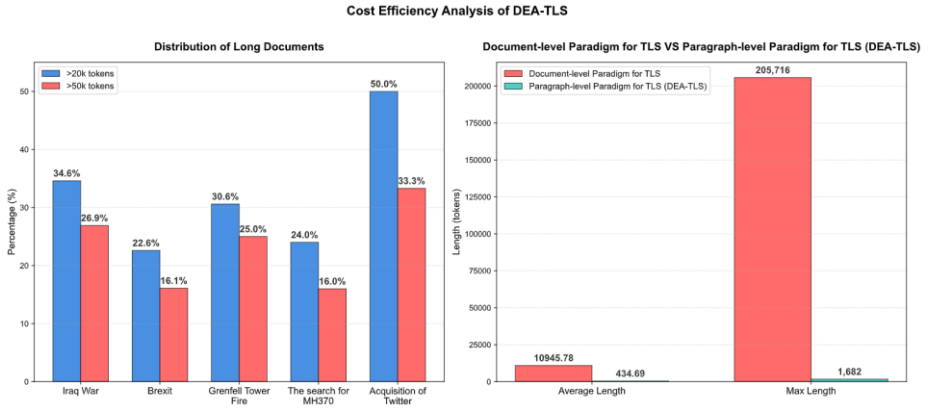

Fig. 3. Cost-efficiency analysis. Left: the proportion of very long documents under document-level processing. Right: the comparison of input lengths between document-level TLS and paragraph-level DEA-TLS.

Factual Accuracy. We also evaluate whether Reflective Extractor improves factual consistency. Following the general idea of large language model-based automatic evaluation [9], we use an LLM judge to assess the factual accuracy of generated timelines

on five representative topics. Accuracy is defined as the proportion of generated timeline entries whose event descriptions are explicitly supported by the source documents.

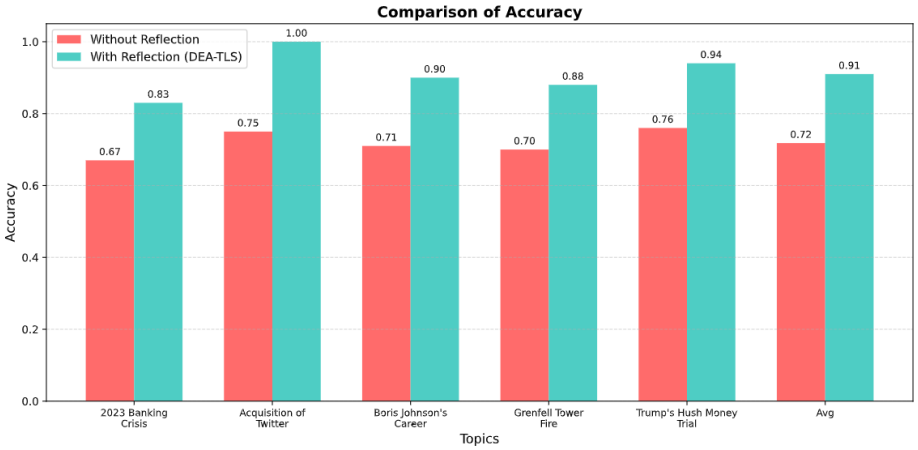

Fig. 4. Accuracy analysis with and without Reflective Extractor.

As shown in Fig. 4, without reflection the model achieves accuracies of 0.67, 0.75, 0.71, 0.70, and 0.76 on 2023 Banking Crisis, Acquisition of Twitter, Boris Johnson's Career, Grenfell Tower Fire, and Trump's Hush Money Trial, respectively. After introducing Reflective Extractor, the corresponding accuracies increase to 0.83, 1.00, 0.90,0.88, and 0.94, and the average accuracy improves from 0.72 to 0.91. This result indicates that reflection-based extraction substantially strengthens factual consistency and makes timeline entries more trustworthy.

Generalization analysis. Furthermore, to assess the generalization ability of our proposed framework beyond open-domain scenarios, we evaluate DEA-TLS on two well-established closed-domain timeline summarization benchmarks: Crisis [22] and T17 [21]. Closed-domain TLS focuses on a fixed document collection, demanding precise temporal and event extraction under more constrained conditions. Following prior work, we report Align F1 (denoted as AR-1 and AR-2) and Date F1 as the primary metrics. We compare DEA-TLS against several baselines including CLUST[5], EGC[23], and CHRONOS. CLUST applies Markov clustering to aggregate events and evaluates cluster importance by the frequency of event-associated dates in the news corpus. EGC introduces an event graph modeling approach that integrates time-aware optimal transport, compressing the full graph into a salient sub-graph for event selection. For our experiments, both DEA-TLS and the CHRONOS baseline are implemented with Qwen2.5-72B as the underlying language model.

As reported in Table 4, DEA-TLS achieves the best performance on both benchmarks across all metrics. On Crisis, DEA-TLS obtains an AR-1 of 0.113, AR-2 of 0.046, and Date F1 of 0.331, consistently surpassing CLUST (0.061/0.013/0.226), EGC

(0.079/0.015/0.291), and CHRONOS (0.108/0.045/0.323). The improvement is particularly notable in Date F1, where DEA-TLS outperforms the previous best CHRONOS by 0.008. On T17, DEA-TLS yields 0.118/0.047/0.536, again exceeding all baselines. Compared to CHRONOS, DEA-TLS improves AR-1 by 0.002, AR-2 by 0.005, and Date F1 by 0.014. These results demonstrate that our framework effectively generalizes to closed-domain TLS scenarios, maintaining strong temporal and event extraction accuracy without any dataset-specific adaptation.

Table 4. Comparison of our method with previous works on closed-domain TLS benchmarks.

	Method	AR-1	AR-2	Date F1
Crisis	CLUST	0.061	0.013	0.226
	EGC	0.079	0.015	0.291
	CHRONOS	0.108	0.045	0.323
	DEA-TLS	**0.113**	**0.046**	**0.331**
T17	CLUST	0.082	0.020	0.407
	EGC	0.103	0.024	0.550
	CHRONOS	0.116	0.042	0.522
	DEA-TLS	**0.118**	**0.047**	**0.536**

5 Conclusion

In this paper, we presented DEA-TLS, a paragraph-level framework for open-domain timeline summarization. By transforming conventional document-level processing into a finer-grained divide-extract-aggregate pipeline, DEA-TLS provides a low-cost solution for timeline summarization with locally deployed open-source models. However,this framework also introduces new challenges at each stage, including noisy paragraph selection, inaccurate event extraction, and semantic redundancy caused by the lack of global information. To address these issues, we further design three collaborative modules: Progressive Divider to select high-value paragraphs, Reflective Extractor to improve extraction accuracy, and Hierarchical Aggregator to reduce redundancy and optimize timeline structure. Experimental results on Open-TLS show that DEATLS consistently outperforms strong baselines across multiple metrics, demonstrating its effectiveness in achieving low-cost and high-accuracy timeline summarization in resource-constrained open-domain settings.

Acknowledgments. This work was supported by the National Natural Science Foundation of China ((Nos. 62476007, 62476283). and in part by the Self-Determined Project of the Disci-pline and Technology Research Center for Large Model Intelligence Applications.We sincerely thank the anonymous reviewers for their valuable comments and suggestions that greatly im-proved the quality of this paper.

References

1. Allan, J., Gupta, R., Khandelwal, V.: Temporal summaries of new topics. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 10–18 (2001)
2. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F.,et al.: Qwen technical report. arXiv preprint arXiv:2309.16609 (2023)
3. Chen, X., Chan, Z., Gao, S., Yu, M.H., Zhao, D., Yan, R.: Learning towards abstractive timeline summarization. In: IJCAI. pp. 4939–4945 (2019)
4. Gao, L., Dai, Z., Pasupat, P., Chen, A., Chaganty, A.T., Fan, Y., Zhao, V., Lao, N., Lee,H., Juan, D.C., et al.: RARR: Researching and revising what language models say, using language models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 16477–16508 (2023)
5. Ghalandari, D.G., Ifrim, G.: Examining the state-of-the-art in news timeline summarization.In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1322–1334 (2020)
6. Hu, Q., Moon, G., Ng, H.T.: From moments to milestones: Incremental timeline summarization leveraging large language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 7232–7246 (2024)
7. Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J.,Joulin, A., Riedel, S., Grave, E.: Atlas: Few-shot learning with retrieval augmented language models. Journal of Machine Learning Research 24(251), 1–43 (2023)
8. Li, H., Su, Y., Cai, D., Wang, Y., Liu, L.: A survey on retrieval-augmented text generation.arXiv preprint arXiv:2202.01110 (2022)
9. Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C.: G-Eval: NLG evaluation using GPT-4 with better human alignment. arXiv preprint arXiv:2303.16634 12, 1 (2023)
10. Martschat, S., Markert, K.: Improving ROUGE for timeline summarization. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. pp. 285–290 (2017)
11. OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F., Almeida,D., et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
12. Qorib, M.R., Hu, Q., Ng, H.T.: Just what you desire: Constrained timeline summarization with self-reflection for enhanced relevance. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 25065–25073 (2025)
13. Song, J., Chim, J., Tsakalidis, A., Ive, J., Atzil-Slonim, D., Liakata, M.: Combining hierachical VAEs with LLMs for clinically meaningful timeline summarisation in social media.In: Findings of the Association for Computational Linguistics: ACL 2024. pp. 14651–14672(2024)
14. Wang, S., Li, Y., Xiao, H., Deng, L., Dong, Y.: Web news timeline generation with extended task prompting. arXiv preprint arXiv:2311.11652 (2023)
15. Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., Zhao, J.: Large language models are better reasoners with self-verification. In: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 2550–2575 (2023)
16. Wu, W., Huang, S., Jiang, Y., Xie, P., Huang, F., Zhao, H.: Unfolding the headline: Iterative self-questioning for news retrieval and timeline summarization. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 4385–4398 (2025)
17. Xie, Y., Kawaguchi, K., Zhao, Y., Zhao, J.X., Kan, M.Y., He, J., Xie, M.: Self-evaluation guided beam search for reasoning. Advances in Neural Information Processing Systems 36,41618–41650 (2023)

18. Yan, R., Kong, L., Huang, C., Wan, X., Li, X., Zhang, Y.: Timeline generation through evolutionary trans-temporal summarization. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. pp. 433–443 (2011)
19. Yu, Y., Jatowt, A., Doucet, A., Sugiyama, K., Yoshikawa, M.: Multi-timeline summarization(MTLS): Improving timeline summarization by generating multiple summaries. 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). pp. 377–387 (2021)
20. Zhang, C., Zhou, T., Cao, P., Jin, Z., Chen, Y., Liu, K., Zhao, J.: DTELS: Towards dynamic granularity of timeline summarization. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 2682–2703 (2025)
21. Tran, G.B., Alrifai, M., Nguyen, D.Q.: Predicting relevant news events for timeline summaries. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 91–92 (2013)
22. Tran, G., Alrifai, M., Herder, E.: Timeline summarization from relevant headlines. In: European Conference on Information Retrieval. pp. 245–256 (2015)
23. Li, M., Ma, T., Yu, M., Wu, L., Gao, T., Ji, H., McKeown, K.: Timeline summarization based on event graph compression via time-aware optimal transport. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 6443–6456 (2021)