Improving Temporal Reasoning of Language Models via Recounted Narratives

Anonymous ACL submission

Abstract

Reasoning about time and temporal relations is an integral aspect of human cognition, essential for perceiving the world and navigating our experiences. Though language models (LMs) have demonstrated impressive performance in many reasoning tasks, temporal reasoning remains challenging due to its intrinsic complexity. In this work, we first study an essential task of temporal reasoning—temporal graph generation, to unveil LMs' inherent, global reasoning capabilities. We show that this task presents great challenges even for the most powerful large language models (LLMs), such as GPT-3.5/4. We also notice a significant performance gap by small LMs ($< 10B$) that lag behind LLMs by 50%. Next, we study how to close this gap with a budget constraint, e.g., not using model finetuning. We propose a new prompting technique tailored for temporal reasoning, GenSort, that first converts the events set to a Python class, then prompts an LM to generate a temporally grounded narrative, guiding the final generation of a temporal graph. Extensive experiments showcase the efficacy of GenSort in improving various metrics. Notably, GenSort attains the highest F1 on the Schema-11 evaluation set, while securing an overall F1 on par with GPT-3.5. GenSort also achieves the best structural similarity across the board, even compared with GPT-3.5/4.

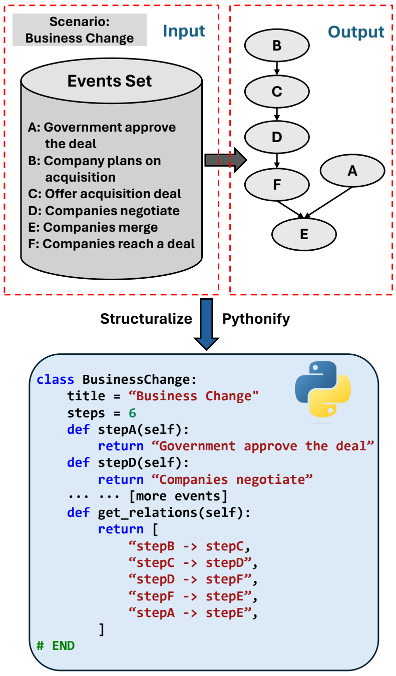

Figure 1: Task overview of **temporal graph generation (TGG)**, where the input is a goal and a set of unordered events. In this work, to better unleash the pre-training power of LMs trained with a mixture of text and code, we cast TGG as a code completion task.

1 Introduction

Temporal reasoning is essential for humans to perceive the world, understand daily communications, and interpret the temporal aspects of experiences (Allen, 1983; Nebel and Bürckert, 1995). The recent advent of language models (LMs) has garnered substantial attention to their impressive performance in various reasoning tasks, such as arithmetic reasoning (Cobbe et al., 2021; Zhong et al., 2024) and commonsense reasoning (Talmor et al., 2019; Anil et al., 2023). Nonetheless, few LMs exist to handle temporal reasoning well (Wang and Zhao, 2023; Chu et al., 2023; Chan et al., 2024), due to the task's inherent complexity, mingled with implicit logical inference and the necessity for profound world knowledge.

To gain deeper insights, the research community mainly focuses on two extremes along the spectrum: either a simple relation extraction task that orders a pair of events (UzZaman et al., 2013; Yuan et al., 2023), or a perplexing commonsense understanding task demanding multi-axis reasoning skills beyond the mere temporal aspect (Wenzel and Jatowt, 2023; Tan et al., 2023; Xiong et al., 2024). Worse still, the former is limited to a *local* scope spanning two adjacent sentences only and fails to account for the significance of *global* temporal relations, leading to overly optimistic results (Yuan and Liu, 2022; Wang and Zhao, 2023). Therefore, neither setup provides a clear understanding of LMs' true temporal reasoning abilities.

In this work, we aim to unveil the **inherent, global temporal reasoning capabilities of LMs**, evaluating them in isolation *free from confounding factors*, and addressing the limitations of previous studies which only focused on local contexts. We first introduce a task of **temporal graph generation (TGG; fig. 1)**: Given a high-level goal $\mathcal{T}$[1] (e.g., business change) and a set of events $\mathcal{V}$, the objective is to produce a temporal graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where a directed edge in $\mathcal{E}$ reveals the temporal order between events. Though this specific notion of TGG is new, many of its applications are not. With TGG, we put forth the first research question. **RQ1: What is the temporal reasoning capability of popular LMs?** Prior work (Wang and Zhao, 2023; Chu et al., 2023) shows a huge gap between AI systems and human performance on various temporal understanding tasks. Additionally, there is a notable performance disparity between proprietary LMs (e.g., GPT-4) and open-source LMs, particularly those with fewer than 10 billion parameters (henceforth, small LMs). Our study on temporal reasoning reveals a similar trend and identifies the existence of both gaps, as demonstrated in Table 1. This further highlights the importance of an in-depth investigation of TGG, since the performance of downstream tasks (e.g., temporal commonsense understanding) is positively correlated with the inherent, global temporal reasoning capability. Observing the model deficiencies, we are motivated to *fill the gap between open-source, small LMs and proprietary large models*. This is due to the fact that open-source LMs are generally more accessible, reproducible, and cost-effective to use (Chen et al., 2023; Zhou et al., 2023). In pursuit of this goal, we present the second research question. **RQ2: With a budget constraint (e.g., not allowing further training), how can small LMs catch up with large models like GPT-3.5/4?** Given the constraint that no training will be used, we propose GenSort, a special prompting technique tailored for temporal reasoning. This method capitalizes on the recent success of the Chain-of-Thought (CoT) technique (Wei et al., 2022b; Kojima et al., 2022), found effective in solving complex reasoning tasks. To approach TGG, GenSort produces a final temporal graph via first <u>Gen</u>erating a *temporally grounded narrative*[2] then <u>Sort</u>ing the input events topologically in reference to the recounted narrative. Inspired by Madaan et al. (2022); Chen et al. (2022); Gao et al. (2023), GenSort also features structural representations by converting the input-output mapping to a Python class, and instructing the generation in code space. We further improve GenSort by introducing high-quality reference narratives as part of few-shot demonstrations.

Extensive experiments across three evaluation benchmarks of diverse genres reveal six interesting findings: 1) small LMs critically **struggle** with temporal reasoning even with few-shot examples; 2) CoT is also **ineffective** at temporal reasoning, in line with existing finding (Chu et al., 2023); 3) GPT-4 sometimes falls off the throne due to **alignment**, when answering sensitive queries; 4) GenSort is a powerful tool to assist small LMs to catch up with or even **surpass GPT-3.5**, and presents strong **compatibility** with various base LMs; 5) the **temporally grounded narratives** are significant in improving LMs' temporal reasoning process; 6) AI systems are far from mastering temporal reasoning, **trailing** the human baseline by 30 F1 points.

We also analyze the impact of shot numbers and perform a holistic evaluation of reference narratives in few-shot examples. 5-shot is found to be the sweet spot for temporal reasoning, after which the performance plateaus, likely due to long-context challenge. We identify three key characteristics of reference narratives for them to avail small LMs most: conciseness, simplicity, and factuality.

2 Related Work

2.1 Temporal Reasoning

This work is deeply rooted in a long-standing yet still challenging NLP domain—temporal reasoning (Allen, 1983; Nebel and Bürckert, 1995), which involves extraction, representation and reasoning with time and events (Sanampudi and Kumari, 2010). Depending on the cognitive complexity, temporal reasoning in NLP is studied at three levels: temporal expression detection, temporal relation extraction, and temporal graph generation. The simplest **temporal expression detection** task is to identify phrases in the text that convey temporal information (Setzer, 2001; Mani et al., 2001; Pustejovsky et al., 2003), commonly known as TimeX.

[1] We use *goal* and *scenario* interchangeably.

[2] In our context, "temporally grounded" refers to events being organized and presented in a way that accurately reflects their temporal sequence or timeline.

Further, under-specified TimeX is typically converted to explicit expressions (e.g., Summer 2024) through a process called *time expression normalization* (Verhagen et al., 2010).

Explicit TimeX is often absent in text, and events usually carry implicit temporal information. To bridge the gap, TempEval (Verhagen et al., 2009; UzZaman et al., 2013) is curated to support the study of **temporal relation extraction**, which aims to detect the temporal relation between two *events* in a document. The most common benchmarks, TB-dense (Chambers et al., 2014) and MATRES (Ning et al., 2018), have witnessed the technique evolution from LSTM (Dligach et al., 2017) and GNN-augmented BERT (Mathur et al., 2021; Wang et al., 2022), to LMs prompting (Yuan et al., 2023). Yet, these benchmarks are limited by their *locality assumption*, where only pairs of events within a two-sentence window are annotated. Even in this simplified scenario of temporal relation extraction, ChatGPT perform poorly, trailing supervised systems by over 30% (Chan et al., 2024).

The most challenging task, **contextualized temporal graph extraction**, is defined as, given a document, generating a corresponding event-level temporal graph (UzZaman et al., 2013; Madaan and Yang, 2021). This task addresses the limitation of locality by priming models to comprehend the entire article and infer relationships even between distant events. Yet, this area is largely under-investigated, partly due to the scarcity of available datasets. A similar task is **script learning** (Regneri et al., 2010; Modi et al., 2016; Sakaguchi et al., 2021), which targets inducing a stereotypical progression of *complex* events (Schank and Abelson, 1975), represented as a temporal graph of more *atomic* events. This task is usually approached by first extracting information snippets from a given document to build an instance graph, and then expanding the graph to generate a schematic graph using GNN (Li et al., 2021; Jin et al., 2022) or LLM prompting (Dror et al., 2023). Given the remarkable similarities between these two tasks, we instead study a temporal reasoning task formulation that is *fundamental* to both, i.e., **temporal graph generation**. It differs from prior work in at least two dimensions: (1) a limited-context setting, where only abstract event descriptions are available, and (2) only a few training samples at hand, rendering fine-tuning techniques inapplicable. This motivates a *training-free assessment* of LMs' *inherent, global* temporal reasoning capability.

2.2 Chain-of-Thought and its Variants

Despite the strong problem-solving capability in the general domain (Wei et al., 2022a), LMs struggle to address more complex reasoning tasks, such as commonsense understanding and arithmatic reasoning (Patel et al., 2021; Talmor et al., 2021a; Huang and Chang, 2023). Wei et al. (2022b) first introduce the concept *Chain-of-Thought (CoT)* by decomposing multi-step problems into intermediate steps. Kojima et al. (2022) further adds a phrase *"Let's think step by step"* to perform zero-shot CoT. These studies underpin the CoT technique in enhancing LMs' capability for complex reasoning.

Down the line, sophisticated prompting schemes are devised through *structuralization*. One approach is to extend the linear chain structure to Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024), enabling expanded exploration space. The huge search space, however, results in a computational resource dilemma. On top of that, leveraging the deterministic execution to narrow the discrepancy between reasoning and final answer, PoT (Chen et al., 2022), PAL (Gao et al., 2023) and Faithful CoT (Lyu et al., 2023) introduce programming languages to describe the reasoning process structurally. These methods are designed exclusively for solving mathematical reasoning and symbolic reasoning, where the reasoning process and computation can be decoupled. In contrast, for temporal reasoning, the reasoning process and the temporal sorting step are intrinsically interleaved. In fact, Chu et al. (2023) has attempted to apply CoT but proved unsuccessful.

Moreover, existing methods are mostly applied to generate intermediate rationales for *simple, atomic outputs*, usually in the format of multi-choice options (Mihaylov et al., 2018; Talmor et al., 2019; Liu et al., 2020), a number (Cobbe et al., 2021; Hendrycks et al., 2021), or yes/no options (Talmor et al., 2021b; Wei et al., 2022a). Our work draws a clear distinction where our focus is on **structural output generation**, augmented with producing a rationale in the form of a compelling and pertinent narrative.

3 Method: GENSORT

Figure 2 provides an overview of the proposed GENSORT method, and draws a comparison against common prompting techniques. Overall, given a scenario and a set of events, GENSORT first converts the input into a Python class, then

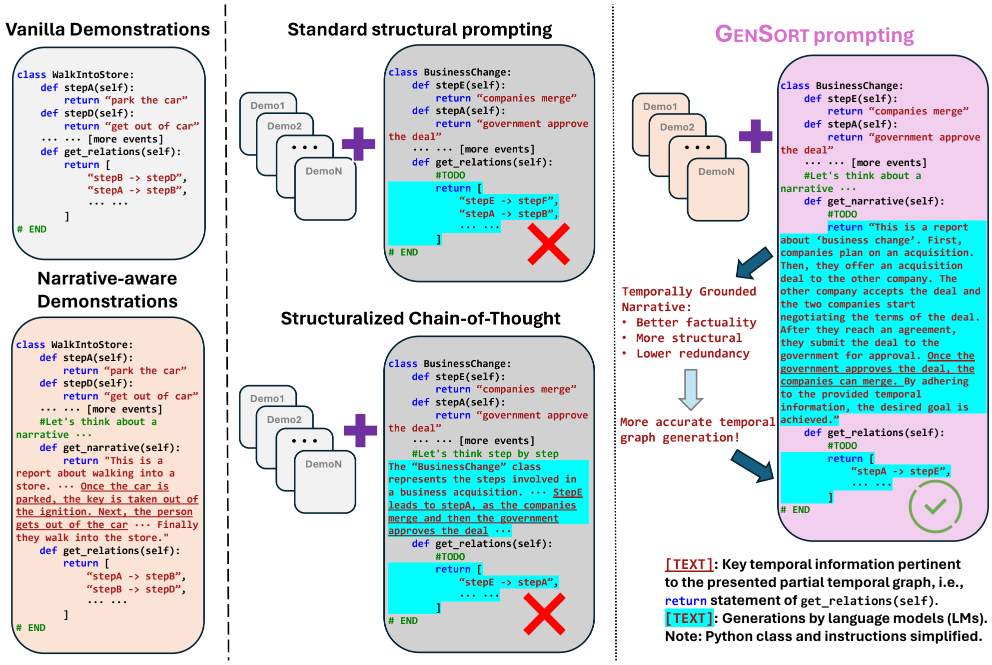

Figure 2: Overview of GenSort, a prompting technique tailored for temporal reasoning. GenSort improves the temporal graph by *recounting* a temporally grounded narrative. Also shown are comparisons with existing methods. The same test example from fig. 1 is displayed. Full example is in fig. A4 with GenSort output in fig. A7.

guides LMs to produce a temporally grounded narrative by arranging events in the correct temporal order, leveraging LMs' intrinsic temporal knowledge. Based on the *recounted* temporal relations articulated in the narrative, LMs are instructed to sort events into a temporal graph. This section will discuss major components in detail: (1) structural representation, (2) GenSort prompting template, and (3) narrative-aware demonstrations.

Structural Representation. Following prior work (Madaan et al., 2022; Chen et al., 2022; Gao et al., 2023), we cast temporal reasoning as a code completion task. This design decision is motivated by the unordered nature of both event sets and temporal relation sets, making a structural representation the optimal choice. Wang et al. (2023a) also shows that combining structural event representations with LMs trained with a mixture of text and code can unleash the full pretraining power. We extend this framing to handle cross-event structures. Specifically, a temporal graph is commonly presented in DOT format (Madaan and Yang, 2021; Sakaguchi et al., 2021), the appearance of which lends itself naturally to the usage of coding format. Furthermore, code execution follows a clear, step-by-step logical flow, mirroring the process of reasoning. Bringing these aspects together results in an alignment between temporal graphs and code structure, facilitating temporal reasoning process.

Concretely, each scenario is represented as a Python class. Each class encapsulates events as functions, where the function name is in the form of "step[A-Z]" such as "stepX", and the function body indicates the event description. The temporal graph is represented as a collection of pairwise temporal relations, enclosed within the return statement of "get_relation()" function, marked by "TODO" for LMs to implement.

GenSort. At inference time, GenSort first prompts LMs to produce a temporally grounded narrative using *Narrative Prompt*. Drawing on the generated narrative, LMs proceed and complete generation in response to *Temporal Graph Prompt*. The entire generation process is in an end-to-end manner, ensuring that LMs explicitly leverage the temporal relations articulated in the narrative to assist the generation of the final temporal graph. We provide a complete example in Appendix C.

Narrative Prompt

```
# Let's think of a narrative to link aforementioned events in the correct temporal order.
def get_narrative(self):
# TODO
```

Temporal Graph Prompt

```
def get_relations(self):
# TODO
# END
```

Overall, GenSort narrows the gap between pre-training and inference by allowing the LM to unfold the narrative knowledge seen during pre-training.

Concretely, our approach leverages LMs' inherent strengths in *generating* and *comprehending* text for narrative and temporal graph generation, respectively. In contrast, directly mapping abstract events to a temporal graph is less effective, as such examples are rarely encountered during pre-training. Practically, generated narratives create imagined experiences for LMs to navigate, which are crucial for tasks requiring temporal reasoning. By reading the *recounted* narrative, it becomes easier for the LMs to construct an implicit timeline to guide event sorting, significantly reducing the reasoning complexity compared to generating temporal graphs from scratch (i.e., using abstract events alone).

Our GENSORT draws a clear distinction from the CoT prompting and its variants in three aspects. First, for CoT, a final answer cannot be easily extracted unless a post-hoc script is designed (Kojima et al., 2022; Wang et al., 2023b), while the output of GENSORT is easy to obtain by parsing the `get_relations()` function. Second, GENSORT produces final outputs in the structural space, while existing methods solely produce *simple, atomic outputs* as discussed in §2.2. Third, the generated rationales by CoTs are not necessarily grounded in real-world experience. In contrast, generated narratives by GENSORT are steered to be more **temporally grounded**, creating an imagined experience for LMs to navigate, which is proved effective.

Narrative-aware Demonstrations. Existing studies (Brown et al., 2020; Wei et al., 2022a) have demonstrated that in-context demonstrations play a critical role in guiding LMs to produce meaningful outputs. GENSORT is no exception, as Table 1 reveals that even GPT-3.5 struggles with temporal reasoning in a zero-shot setting. Thus, few-shot examples are provided by default. For GENSORT to succeed, high-quality and relevant rehearsed narratives, termed *reference narratives*, need to be created and embedded in these demonstrations.

Capitalizing on the recent success of using LMs to generate demonstrations (Yu et al., 2023; Li et al., 2023), we prompt GPT-3.5/4 to produce reference narratives. Concretely, for each demonstration, abstracted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we feed both $\mathcal{V}$ and $\mathcal{E}$ into GPT-3.5/4, using our designed reference narrative generation templates, dubbed *meta prompts*. In total, we create 4 types of meta prompts covering diverse genres like news and children's stories. Additionally, when feeding $\mathcal{G}(\mathcal{V}, \mathcal{E})$ into GPT-3.5/4, we use two *input formats* to define a Python class (*alphabetical* like "step`X`" in fig. A8 vs. descriptive like "pushPedal" in fig. A9). We later evaluate the usefulness of each meta prompt in §5.2. Details of meta prompts are documented in Appendix D.

4 Experiment

In this work, we focus on **Temporal Graph Generation (TGG)**, an essential task of temporal reasoning. Here, we discuss datasets, experimental setup, baselines, and evaluation metrics. We provide additional implementation details in Appendix A.

4.1 Dataset

In line with the literature, we use **ProScript** (Sakaguchi et al., 2021) as the major benchmark, where a temporal script is represented as a directed acyclic graph, which were collected from a diverse range of sources including ROCStories (Mostafazadeh et al., 2016), Descript (Wanzare et al., 2016), and Virtual home (Puig et al., 2018). We also adopt two other datasets to enrich the evaluated genres and domains, and make necessary changes for the TGG task: 1) **Schema-11** evaluation set (Dror et al., 2023), which contains human-curated event schemas for 11 newsworthy topics, such as *armed robbery* and *business change*; and 2) **WikiHow Script** corpus (Lyu et al., 2021), a collection of multilingual how-to articles depicting necessary steps performed in sequence to achieve a high-level goal, covering a wide range of daily activities. Dataset statistics are included in Table A2, and we provide detailed dataset processing scripts in Appendix B.

4.2 Setup

As our goal is to study the capability and generalizability of existing LMs, and our GENSORT without any fine-tuning, we assume no access to large-scale training sets except for few-shot demonstrations. Therefore, all experiments are conducted in a 5-shot setting. We provide analysis on the impact of the shots numbers in §5.2. We consider three base models to spotlight the compatibility and versatility of GENSORT. We include very recent, strong LMs, showing promising results on various reasoning tasks and code completion tasks, MISTRAL-7B (Jiang et al., 2023), GEMMA-7B (Mesnard et al., 2024), and LLAMA3-8B (AI@Meta, 2024). For all base models, we use their instruction-fine-tuned versions for experiments.

Shown in fig. 2, we represent the event set as a suite of Python methods, by serializing the unordered event set. For each scenario, we randomly

Method	Proscript				Schema-11				WikiHow Script				Avg.	
	F1↑	GED↓	$k(\mathcal{G})$	Cons.↑	F1↑	GED↓	$k(\mathcal{G})$	Cons.↑	F1↑	GED↓	$k(\mathcal{G})$	Cons.↑	F1↑	GED↓
Baselines														
Random	14.0	1.47	1.00	7.8	19.4	3.91	1.00	7.8	14.2	0.06	1.00	8.8	15.9	1.81
GPT-3.5 (0-shot)*	18.4	2.25	1.06	38.6	30.1	4.48	1.27	30.2	17.2	2.80	1.11	40.8	21.9	3.18
GPT-3.5	43.4	1.71	1.07	38.8	62.8	3.30	1.36	50.2	31.0	1.58	1.10	35.4	45.7	2.20
GPT-4	63.9	1.64	1.02	61.4	44.1	7.97	0.64	46.3	43.0	1.71	1.04	48.5	50.3	3.77
GEMMA-7B (Mesnard et al., 2024)														
Standard Prompting	19.7	**2.35**	1.02	**20.4**	27.8	5.03	**1.03**	18.3	17.5	**2.88**	**0.96**	**17.3**	21.7	**3.42**
Chain-of-Thought	20.0	**2.35**	1.01	20.0	26.4	5.03	**1.03**	14.9	13.6	5.91	0.73	11.5	20.0	4.43
GENSORT (no reference)	20.0	2.47	**1.00**	17.3	27.9	**4.78**	1.09	18.1	15.2	5.03	0.81	13.9	21.0	4.09
GENSORT (alphabetical meta)	**21.8**	2.48	**1.00**	18.3	**36.0**	4.84	1.06	19.7	**17.9**	2.95	**0.96**	16.9	**25.2**	**3.42**
GENSORT (descriptive meta)	21.3	2.60	0.99	17.8	34.8	5.00	1.06	**20.8**	**17.9**	**2.88**	0.95	16.8	24.7	3.49
MISTRAL-7B (Jiang et al., 2023)														
Standard Prompting	30.7	2.16	1.05	22.3	35.3	4.55	1.12	29.1	**22.5**	**2.09**	1.11	**18.9**	29.5	2.93
Chain-of-Thought	29.8	2.66	**1.02**	22.1	35.2	5.33	0.94	30.5	20.5	2.59	1.10	17.4	28.5	3.53
GENSORT (no reference)	32.5	3.04	0.95	19.4	42.3	5.27	**1.00**	27.6	21.8	3.33	0.98	15.4	32.2	3.88
GENSORT (alphabetical meta)	35.2	**2.11**	**1.02**	22.4	50.9	4.30	1.03	**36.1**	21.7	2.49	**1.04**	14.8	35.9	2.97
GENSORT (descriptive meta)	**35.4**	2.14	**1.02**	**23.0**	**52.7**	**3.90**	1.06	32.5	22.1	2.53	**1.04**	15.1	**36.7**	**2.86**
LLAMA3-8B (AI@Meta, 2024)														
Standard Prompting	25.1	2.39	1.18	19.9	28.3	4.42	1.24	19.9	20.6	1.17	1.07	21.2	24.7	2.66
Chain-of-Thought	30.1	2.06	**1.00**	23.3	37.3	5.79	0.85	23.5	22.6	**0.99**	**1.02**	**24.3**	30.0	2.95
GENSORT (no reference)	35.5	1.88	**1.00**	25.3	52.6	**3.18**	1.12	35.0	25.4	**0.99**	**1.02**	20.9	37.8	**2.02**
GENSORT (alphabetical meta)	**39.5**	1.87	1.01	**28.8**	59.0	3.72	1.12	39.1	26.3	1.01	1.03	22.5	41.6	2.20
GENSORT (descriptive meta)	38.7	**1.86**	1.01	28.4	**61.5**	3.57	**1.09**	**45.6**	**26.5**	1.04	1.03	22.3	**42.2**	2.16

Table 1: Main results of base LMs and strong baselines on TGG evaluation benchmarks (average of 3 runs). For each base model, best results are **bold**, and GENSORT's variants better than both Standard Prompting and CoT are highlighted. GENSORT results that outperform 5-shot GPT-3.5 and GPT-4 are in blue. Results that meet both criteria are in purple. On average, GENSORT boosts F1 metric over its base model by 16% to 71%, and sometimes improves the GED metric. GENSORT-augmented LLAMA3-8B achieves best overall F1 (63.5 F1 by 3-shot variant; fig 3) and GED results on Schema-11. Also, it only trails GPT-3.5 and GPT-4 by 8% and 14% on average, while yielding a lower average GED. By default, 5-shot examples are provided. Full results in Table A1.

shuffle the input Python methods three times, apply models to each shuffle with greedy decoding at inference. For GENSORT, we use *Simple Report*-style narratives by GPT-4 (style details in table A3).

4.3 Baselines

To showcase the effectiveness of GENSORT, for each base model we compare with standard structural prompting and structuralized chain-of-thought prompting (fig. 2). We also remove reference narratives in demonstrations to highlight the importance of narrative-aware few-shot demonstrations, and conduct a holistic evaluation of reference narratives in §5.2. We include a random baseline, where events are naively connected to form a *linear* temporal chain based on the order they appear in the input. We also experiment with two strong proprietary models, GPT-3.5[3] and GPT-4 (OpenAI, 2023)[4] to help gauge the gap between AI systems and human-level performance.

4.4 Evaluation Metrics

We denote the ground-truth and generated temporal graphs as $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and $\hat{\mathcal{G}}(\mathcal{V}, \hat{\mathcal{E}})$, respectively. we compare both semantic and structural similarities between $\mathcal{G}$ and $\hat{\mathcal{G}}$, following prior work (Sakaguchi et al., 2021; Madaan et al., 2022). To evaluate semantic similarity, we report *precision (P)* and *recall (R)*, defined as below, as well as *F1*.

$$\text{Precision} = \frac{|\mathcal{E} \cap \hat{\mathcal{E}}|}{|\hat{\mathcal{E}}|} \quad \text{Recall} = \frac{|\mathcal{E} \cap \hat{\mathcal{E}}|}{|\mathcal{E}|}$$

To assess structural similarities, we consider:

- *Graph Edit Distance* (*GED*; Abu-Aisheh et al., 2015) calculates the minimum number of edits (node/edge removal/additions) to transform $\hat{\mathcal{G}}$ to a graph isomorphic to $\mathcal{G}$.
- *Graph Statistics*: fraction of the number of edges between $\hat{\mathcal{G}}$ and $\mathcal{G}$ ($\frac{|\hat{\mathcal{E}}|}{|\mathcal{E}|}$); the number of connected components in $\hat{\mathcal{G}}$, denoted as $k(\mathcal{G})$. The goal is to bring both statistics closer to 1, additionally ensuring $k(\mathcal{G})$ is at least 1.

We further calculate *Pair-wise Consistency* between $\hat{\mathcal{G}}_i$ and $\hat{\mathcal{G}}_j$, where we compare generated graphs, based on two randomly shuffled inputs, and compute the proportion of common temporal links produced in both graphs, i.e., $\frac{|\hat{\mathcal{E}}_i \cap \hat{\mathcal{E}}_j|}{|\hat{\mathcal{E}}_i \cup \hat{\mathcal{E}}_j|}$.

5 Results and Analyses

5.1 Main Results

Major results are included in Table 1, and the full results (across all 7 metrics) can be found in Ta-

[3] https://chat.openai.com/; gpt-35-turbo-16k-0613, training data up to Sept. 2021.

[4] gpt-4-turbo-0125-preview, data up to Dec. 2023.

ble A1. Below are our major findings.

1) *With the few-shot setup, small LMs are dramatically underperforming, reaching barely 50% of GPT-4's capabilities.* The three base models, whether using standard prompting or CoT, consistently under-perform GPT-4 and attain 40% to 60% of its average F1 scores. Among them, MISTRAL-7B achieves the highest F1 scores, while LLAMA3-8B produces temporal graphs most similar to the ground truth, as measured by GED.

2) *Unlike many other reasoning tasks, CoT does not always work for temporal reasoning and sometimes degrades performance.* Unlike mathematical or logical reasoning (Wei et al., 2022b), CoT prompting does not necessarily enhance model performance on temporal reasoning tasks. Across all three base models, there is a notable degradation in F1 and GED scores with CoT, except for LLAMA3's F1 scores. This is not TGG-specific, but rather a common pattern across various temporal understanding tasks (Chu et al., 2023), highlighting the need for specialized approaches to temporal reasoning. Outputs by CoT are included in fig. A6.

3) *GPT-4 is not always the champion, owing to the added safety layer.* GPT-4 implements safety measures through human-preference alignment (OpenAI, 2023), which enhances model safety by prompting more cautious responses, potentially leading to performance drop (Bai et al., 2022; Bekbayev et al., 2023). Especially on **Schema-11**, GPT-4 refrains from providing answers to sensitive scenarios like "bombing attacks",[5] and thus fails to produce a valid temporal graph.

4) *With* GENSORT*, small LMs can perform comparably to GPT-3.5, or even take the lead.* When equipped with GENSORT, the overall semantic correctness (F1) and structural similarity (GED) of the generated temporal graphs are significantly enhanced, regardless of which base LM is used. The average improvement of F1 over naively prompting the base model is between 16% to 71%. As the power of the base LM grows, GENSORT demonstrates greater consistency in its outputs. Notably, with LLAMA3-8B, the strongest base LM, GENSORT achieves an F1 score that is comparable to GPT-3.5 (42.2 vs. 45.7), and even outperforms GPT-3.5/4 on GED. These results demonstrate the potential of applying GENSORT in a wide range of temporal understanding tasks in future research.

5) *Recounting temporally grounded narrative is a prerequisite for LMs to generate temporal graphs accurately.* Without high-quality reference narratives, LMs struggle to generate temporally grounded narratives, leading to a detrimental impact on GENSORT-augmented GEMMA-7B (e.g., a 0.7 F1 drop and a 0.67 GED increase).

6) *LMs, including the powerful GPT-4, lag far behind human-level performance in temporal reasoning.* The SOTA F1 score (by GPT-4) on ProScript is 63.9, whereas the human baseline F1 is 89.3 (Sakaguchi et al., 2021). While GENSORT has notably narrowed the gap between small and large LMs, AI models have not mastered temporal reasoning yet, and further research efforts are needed for LMs to match human performance.

[5] In our experiments, we already disabled content filtering.

5.2 Further Studies on GENSORT

We conduct ablation studies using LLAMA3-8B, to explore the effect of the few-shot demonstrations and the recounted reference narratives.

Does the number of shots matter? Fig. 3 illustrates how F1 scores change with the number of shots in demonstrations. As can be seen, GPT-3.5 and GENSORT show resilience to changes in shot numbers after an initial sharp increase. The performance nearly stabilizes in the range of 5-10 shots, though a slight drop is observed later, presumably due to insufficient capability of long-context comprehension (Liu et al., 2023; Li et al., 2024). Of particular interest is the performance of GENSORT with 3 shots on Schema-11, outperforming the best variant of GPT-3.5 (F1 of 63.5 vs. 62.8). This further illustrates GENSORT's potential of boosting small LMs in the long run. It is also noticeable that F1 scores of the standard prompting technique have a V-shape between 1-shot and 5-shot, highlighting its sensitiveness to in-context demonstrations.

We also display the GED scores in relation to number of shots in fig. A1. We observe similar instability in the standard prompting technique, along with the performance plateau after 5 shots.

What characteristics define effective reference narratives? Given that reference narratives in GENSORT are machine-generated, we aim to explore what qualities matter most for the TGG task. Here, the three variables influencing reference narratives are: (1) narrative generation model (GPT-3.5 vs. GPT-4), (2) input format (alphabetical vs. descriptive), and (3) 4 meta prompt types (varying degrees of factuality and readability). We show detailed meta prompts in Appendix D.

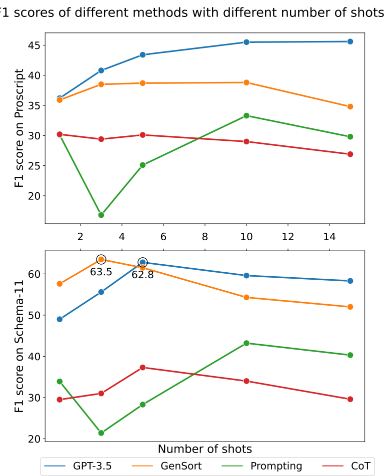

Figure 3: F1 scores on ProScript and Schema-11 in relation to the number of shots in demonstrations. We identify the **instability** in the standard prompting, and the **performance plateau** after 5 shots.

Fig. 4 and fig. A2 show results of F1 and GED with varying meta prompts. Surprisingly, the choice of the generator does not significantly impact the graph quality, with average F1 scores of 36.4 for GPT-3.5 and 37.0 for GPT-4, and GED scores of 1.90 vs. 1.94. Similarly, there is no significant difference between alphabetical and descriptive input formats. The most *impactful* factor is the meta prompt type. Grouping performance bars by prompt type reveals a clear variance in model performance. Among the first three groups, *Simple English* narratives, i.e., good for 10-year-olds, stand out. This suggests that narratives should be simple and concise, as verbose ones are less effective. We find that *News Report* narratives prioritize procedural and factual content, minimizing distractions like descriptive settings or figurative language that can often be found in both fiction or non-fiction stories. We thus combine *Simple English* and *News Report* to leverage their strengths, dubbed *Simple Report*. In summary, we identify three key characteristics for reference narratives: *conciseness*, *simplicity* and *factuality*.

How faithful is the temporal graph to intermediate narratives? Here, we look into whether GENSORT-augmented LMs are **self-faithful**, i.e., whether the narrative and the temporal graph **align** in terms of the temporal order of events. Higher self-faithfulness is crucial and desired, as misalignment would diminish the effort of generating a temporally grounded narrative.[6]

Motivated by the recent success of using LMs as judges (Zheng et al., 2023; Zhang et al., 2024), we employ GPT-4 to assess the self-faithfulness of 600 randomly sampled outputs by GENSORT-augmented LLAMA3-8B. We prompt GPT-4 to perform a 5-way assessment and provide judgment rationales. Additionally, GPT-4 is instructed to count the temporal links in the temporal graphs and identifies aligned temporal links for a sanity check. This helps humans capture the failure modes and make necessary interventions. Based on automated responses and on-demand human inspections, we find a medium-to-high alignment of 72.8%. Details of templates and the inspection process are included in Appendix E.

[6]Faithfulness $\neq$ correctness. A faithful temporal graph may still contain logical errors from the generated narratives.

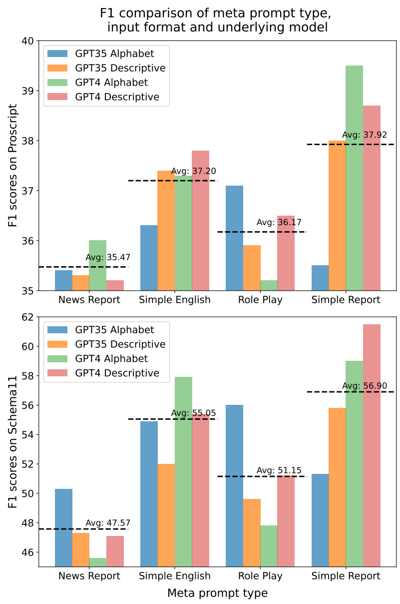

Figure 4: F1 scores on ProScript and Schema-11 with different meta prompts. Average performance grouped by prompt type is also shown. Notably, a *Simple Report*-style, GPT-4 generated narrative leads to the best score due to its **conciseness**, **simplicity** and **factuality**, essential qualities for a *high-quality* reference narrative.

6 Conclusion

In this paper, we assess the inherent, global temporal reasoning capabilities of LMs, by studying the core challenge of temporal reasoning—temporal graph generation (TGG). To this end, we propose GENSORT, a novel prompting technique tailored for temporal reasoning. Concretely, with few-show narrative-aware demonstrations as references, GENSORT prompts LMs to first generate a temporally grounded narrative and then sort the input events topologically into a temporal graph, by manipulating the generation in code space. Extensive experiments showcase GENSORT's effectiveness, demonstrated by its superior performance over GPT-3.5 on multiple metrics, as well as the compatibility of GENSORT with various LMs.

7 Limitations

Evaluation benchmarks. In this work, we have included three evaluation benchmarks, aiming to cover a diverse array of genres and domains. Yet, these three benchmarks cannot comprehensively represent the entire spectrum. For example, healthcare and biomedical (Alfattni et al., 2020) domains offer great opportunities to study temporal graph generation as well. In future research, we plan to extend GENSORT to more applications, and examine its true generalizability in the wild.

Human Baseline Comparison. The last finding we deliver in §5.1 might not hold for all benchmarks, as the human baseline comparison was conducted solely on the ProScript dataset. We will continue the endeavor of seeking participants to perform human evaluations on the other two datasets to enhance the credibility of our claim.

GPU resources. The base LMs used in this work are of 7 to 8 billions parameters. It is thus more time-consuming than traditionally small models like BERT (Devlin et al., 2019) at inference time, which in turn results in a higher carbon footprint. Specifically, we run each base LM on 1 single NVIDIA A40 or NVIDIA L40 with significant CPU and memory resources. The combined inference time for each LM on the three benchmarks ranges from 10 to 20 hours, depending on the configurations.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In *ICPRAM 2015 - Proceedings of the International Conference on Pattern Recognition Applications and Methods, Volume 1, Lisbon, Portugal, 10-12 January, 2015*, pages 271–278. SciTePress.

AI@Meta. 2024. Llama 3 model card.

Ghada Alfattni, Niels Peek, and Goran Nenadic. 2020. Extraction of temporal relations from clinical free text: A systematic review of current approaches. *J. Biomed. Informatics*, 108:103488.

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. 2023. Palm 2 technical report. *CoRR*, abs/2305.10403.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Aibek Bekbayev, Sungbae Chun, Yerzat Dulat, and James Yamazaki. 2023. The poison of alignment. *CoRR*, abs/2308.13449.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Chunkit Chan, Cheng Jiayang, Weiqi Wang, Yuxin Jiang, Tianqing Fang, Xin Liu, and Yangqiu Song. 2024. Exploring the potential of ChatGPT on sentence level relations: A focus on temporal, causal, and discourse relations. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 684–721, St. Julian's, Malta. Association for Computational Linguistics.

Hailin Chen, Fangkai Jiao, Xingxuan Li, Chengwei Qin, Mathieu Ravaut, Ruochen Zhao, Caiming Xiong, and Shafiq Joty. 2023. Chatgpt's one-year anniversary: Are open-source large language models catching up? *CoRR*, abs/2311.16989.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *CoRR*, abs/2211.12588.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. 2023. Timebench: A comprehensive evaluation of temporal reasoning abilities in large language models. *CoRR*, abs/2311.17667.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 746–751, Valencia, Spain. Association for Computational Linguistics.

Rotem Dror, Haoyu Wang, and Dan Roth. 2023. Zero-shot on-the-fly event schema induction. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 705–725, Dubrovnik, Croatia. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Xiaomeng Jin, Manling Li, and Heng Ji. 2022. Event schema induction with double graph autoencoders. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2025, Seattle, United States. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Manling Li, Sha Li, Zhenhailong Wang, Lifu Huang, Kyunghyun Cho, Heng Ji, Jiawei Han, and Clare Voss. 2021. The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5203–5215, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rui Li, Guoyin Wang, and Jiwei Li. 2023. Are human-generated demonstrations necessary for in-context learning? *CoRR*, abs/2309.14681.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *CoRR*, abs/2404.02060.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *CoRR*, abs/2307.03172.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 184–200, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Aman Madaan and Yiming Yang. 2021. Neural language modeling for contextualized temporal graph generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 864–881, Online. Association for Computational Linguistics.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Inderjeet Mani, George Wilson, Lisa Ferro, and Beth Sundheim. 2001. Guidelines for annotating temporal information. In *Proceedings of the First International Conference on Human Language Technology Research*.

Puneet Mathur, Rajiv Jain, Franck Dernoncourt, Vlad Morariu, Quan Hung Tran, and Dinesh Manocha. 2021. TIMERS: Document-level temporal relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 524–533, Online. Association for Computational Linguistics.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2016. InScript: Narrative texts annotated with script information. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3485–3493, Portorož, Slovenia. European Language Resources Association (ELRA).

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Bernhard Nebel and Hans-Jürgen Bürckert. 1995. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *J. ACM*, 42(1):43–66.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba.

2018. Virtualhome: Simulating household activities via programs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8494–8502. Computer Vision Foundation / IEEE Computer Society.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 28–34. AAAI Press.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden. Association for Computational Linguistics.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Suresh Kumar Sanampudi and G. Vijaya Kumari. 2010. Temporal reasoning in natural language processing: A survey. *International Journal of Computer Applications*, 1:68–72.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans and knowledge. In *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, September 3-8, 1975*, pages 151–157.

Andrea Setzer. 2001. *Temporal information in newswire articles : an annotation scheme and corpus study*. Ph.D. thesis, University of Sheffield, UK.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. 2021a. Commonsenseqa 2.0: Exposing the limits of AI through gamification. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. 2021b. Commonsenseqa 2.0: Exposing the limits of ai through gamification. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. 2023. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14820–14835, Toronto, Canada. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.

Marc Verhagen, Robert J. Gaizauskas, Frank Schilder, Mark Hepple, Jessica L. Moszkowicz, and James Pustejovsky. 2009. The tempeval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43:161–179.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden. Association for Computational Linguistics.

Liang Wang, Peifeng Li, and Sheng Xu. 2022. DCT-centered temporal relation extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2087–2097, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xingyao Wang, Sha Li, and Heng Ji. 2023a. Code4Struct: Code generation for few-shot event structure prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yuqing Wang and Yun Zhao. 2023. TRAM: benchmarking temporal reasoning for large language models. *CoRR*, abs/2310.00835.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3494–3501, Portorož, Slovenia. European Language Resources Association (ELRA).

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Georg Wenzel and Adam Jatowt. 2023. An overview of temporal commonsense reasoning and acquisition. *CoRR*, abs/2308.00002.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. *CoRR*, abs/2401.06853.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Chenhan Yuan, Qianqian Xie, and Sophia Ananiadou. 2023. Zero-shot temporal relation extraction with ChatGPT. In *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 92–102, Toronto, Canada. Association for Computational Linguistics.

Weizhe Yuan and Pengfei Liu. 2022. restructured pretraining. *CoRR*, abs/2206.11147.

Xinliang Frederick Zhang, Carter Blum, Temma Choji, Shalin Shah, and Alakananda Vempala. 2024. ULTRA: Unleash LLMs' potential for event argument extraction through hierarchical modeling and pairwise refinement. *ArXiv*, abs/2401.13218.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, Bo Du, and Dacheng Tao. 2024. Achieving >97 *arXiv preprint arXiv:2404.14963*.

Jianlong Zhou, Heimo Müller, Andreas Holzinger, and Fang Chen. 2023. Ethical chatgpt: Concerns, challenges, and commandments. *CoRR*, abs/2305.10646.

Method	Proscript							Schema-11							WikiHow Script							Avg			
	P	R	F1	GED	$\frac{\lvert\hat{\mathcal{E}}\rvert}{\lvert\mathcal{E}\rvert}$	$k(\mathcal{G})$	Cons.	P	R	F1	GED	$\frac{\lvert\hat{\mathcal{E}}\rvert}{\lvert\mathcal{E}\rvert}$	k(G)	Cons.	P	R	F1	GED	$\frac{\lvert\hat{\mathcal{E}}\rvert}{\lvert\mathcal{E}\rvert}$	$k(\mathcal{G})$	Cons.	F1	GED	$k(\mathcal{G})$	Cons.
Baselines																									
Random	14.6	13.6	14.0	1.47	0.93	1.00	7.8	20.2	18.8	19.4	3.91	0.96	1.00	7.8	14.2	14.2	14.2	0.06	1.00	1.00	8.8	15.9	1.81	1.00	8.1
GPT-3.5 (0-shot)	18.8	18.1	18.4	2.25	0.95	1.06	38.6	30.8	30.1	30.1	4.48	1.02	1.27	30.2	17.0	17.8	17.2	2.80	1.04	1.11	40.8	21.9	3.18	1.15	36.5
GPT-3.5	44.9	42.3	43.4	1.71	0.92	1.07	38.8	65.8	60.5	62.8	3.30	0.92	1.36	50.2	31.0	31.1	31.0	1.58	1.01	1.10	35.4	45.7	2.20	1.18	41.5
GPT-4	65.7	62.6	63.9	1.64	0.94	1.02	61.4	44.9	43.5	44.1	7.97	0.57	0.64	46.3	43.0	43.1	43.0	1.71	0.98	1.04	48.5	50.3	3.77	0.90	52.1
GEMMA-7B (Mesnard et al., 2024)																									
Standard Prompting	20.2	19.4	19.7	**2.35**	0.96	1.02	**20.4**	28.5	27.6	27.8	5.03	0.97	**1.03**	18.3	16.9	18.5	17.5	**2.88**	**0.99**	**0.96**	**17.3**	21.7	**3.42**	**1.00**	**18.7**
Chain-of-Thought	20.3	19.9	20.0	**2.35**	**0.98**	1.01	20.0	26.4	26.4	26.4	5.03	**1.01**	**1.03**	14.9	13.0	14.5	13.6	5.91	0.77	0.73	11.5	20.0	4.43	0.92	15.5
GENSORT (no reference)	20.5	19.7	20.0	2.47	0.95	**1.00**	17.3	28.6	27.4	27.9	**4.78**	0.96	1.09	18.1	14.6	16.1	15.2	5.03	0.80	0.81	13.9	21.0	4.09	0.97	16.4
GENSORT (alphabetical meta)	**22.4**	**21.4**	**21.8**	2.48	0.95	**1.00**	18.3	**36.9**	**35.5**	**36.0**	4.84	0.95	1.06	19.7	17.1	**18.9**	**17.9**	2.95	0.96	**0.96**	16.9	**25.2**	**3.42**	1.01	18.3
GENSORT (descriptive meta)	21.9	21.0	21.3	2.60	0.94	0.99	17.8	35.0	34.8	34.8	5.00	0.98	1.06	**20.8**	**17.2**	**18.9**	**17.9**	**2.88**	0.96	0.95	16.8	24.7	3.49	**1.00**	18.5
MISTRAL-7B (Jiang et al., 2023)																									
Standard Prompting	31.7	30.0	30.7	2.16	0.94	1.05	22.3	37.6	33.7	35.3	4.55	0.93	1.12	29.1	**22.3**	**23.0**	**22.5**	**2.09**	1.02	1.11	**18.9**	29.5	2.93	1.09	23.4
Chain-of-Thought	30.7	29.3	29.8	2.66	0.92	**1.02**	22.1	36.0	34.6	35.2	5.33	0.95	0.94	30.5	20.9	20.5	20.5	2.59	**0.99**	1.10	17.4	28.5	3.53	**1.02**	23.3
GENSORT (no reference)	33.4	31.9	32.5	3.04	0.89	0.95	19.4	44.0	41.3	42.3	5.27	0.86	**1.00**	27.6	21.7	22.3	21.8	3.33	0.91	**0.98**	15.4	32.2	3.88	0.98	20.8
GENSORT (alphabetical meta)	35.9	34.8	35.2	**2.11**	**0.96**	**1.02**	22.4	52.8	49.5	50.9	4.30	0.95	1.03	**36.1**	21.4	22.2	21.7	2.49	0.96	1.04	14.8	35.9	2.97	1.03	**24.4**
GENSORT (descriptive meta)	**36.2**	**34.9**	**35.4**	2.14	**0.96**	**1.02**	**23.0**	**54.5**	**51.4**	**52.7**	**3.90**	**0.96**	1.06	32.5	21.8	22.5	22.1	2.53	0.95	1.04	15.1	**36.7**	**2.86**	1.04	23.5
LLAMA-8B (AI@Meta, 2024)																									
Standard Prompting	27.3	23.4	25.1	2.39	0.85	1.18	19.9	30.8	26.6	28.3	4.42	0.91	1.24	19.9	21.5	20.1	20.6	1.17	0.97	1.07	21.2	24.7	2.66	1.16	20.3
Chain-of-Thought	30.1	30.4	30.1	2.06	**1.00**	**1.00**	23.3	38.0	36.9	37.3	5.79	0.83	0.85	23.5	21.9	23.5	22.6	**0.99**	1.05	**1.02**	**24.3**	30.0	2.95	0.96	23.7
GENSORT (no reference)	36.7	34.7	35.5	1.88	0.93	**1.00**	25.3	54.0	51.6	52.6	**3.18**	**0.96**	1.12	35.0	25.4	25.6	25.4	**0.99**	0.99	**1.02**	20.9	37.8	**2.02**	1.05	27.1
GENSORT (alphabetical meta)	**40.4**	**38.8**	**39.5**	1.87	0.95	1.01	**28.8**	61.9	56.8	59.0	3.72	0.93	1.12	39.1	25.9	**26.9**	26.3	1.01	1.01	1.03	22.5	41.6	2.20	1.05	30.1
GENSORT (descriptive meta)	39.8	38.0	38.7	**1.86**	0.94	1.01	28.4	**64.1**	**59.6**	**61.5**	3.57	**0.96**	**1.09**	**45.6**	**26.2**	**26.9**	**26.5**	1.04	**1.00**	1.03	22.3	**42.2**	2.16	**1.04**	**32.1**

Table A1: Full results of three base LMs and select strong baselines on our compiled suite of TGG evaluation benchmarks. For each base model, best results are in **bold**. For precision (P), recall (R), F1 and Consistency (Cons.). a higher number indicates a better performance. For GED, a lower number indicates a better results. For $\frac{|\hat{\mathcal{E}}|}{|\mathcal{E}|}$, the optimal value is 1. For $k(\mathcal{G})$, the best value is 1 and numbers smaller than 1 are not favored which indicates LMs fail to generate a valid graph for some input scenarios. *Unless otherwise noted, 5-shot demonstrations are provided.

A Additional Implementation Details

Few-shot Demonstration Selection. To construct the demonstration bank, we select 15 examples from the training set of ProScript, following Madaan et al. (2023). We do so because we expect to include non-linear temporal graph examples in our demonstrations, for which only ProScript can fulfill the requirement. Then, we use the same demonstrations as few-shot examples for experiments, regardless of the evaluation benchmark.

Model Cards. In this work, we have experimented with 3 base LMs. Below lists the exact Huggingface model cards used in this work.

- Gemma-7B: `google/gemma-7b-it`
- Mistral-7B: `mistralai/Mistral-7B-Instruct-v0.2`
- Llama3-8B: `meta-llama/Meta-Llama-3-8B-Instruct`

B Dataset Processing

This section documents the processing steps performed on Schema-11 and WikiHow Script to cater for the temporal reasoning task of our interest. We do not use any Python packages for dataset processing. Meanwhile, based on our inspection, we do not spot any offensive content in these three datasets.

Schema-11. In their original annotations, an event node is marked in arg_0-trigger-arg_1 format, and we manually convert it to a natural sentence. We specifically adopt annotations under *schemas_dan_d* directory.

WikiHow Script corpus. The original dataset features multilingualism, while we only take their English portion for this study. Then, We only keep ordered how-to articles where steps are presented in chronological order. Lastly, we cap the maximum number of steps at 20, which reduces the corpus size from $3,3035$ to $2,077$.

C Complete Examples

Using the same example as in fig. 1 and fig. 2, we show the complete examples (including generations by one base LM, Llama3-8B) of Standard Prompting, CoT and GenSort. We first show the input part of Standard Prompting and CoT in fig. A3, and the input of GenSort in fig. A4. Outputs by Standard Prompting, CoT and GenSort are displayed in fig. A5, fig. A6 and fig. A7, respectively. As we can easily see, the output of Standard Prompting is completely wrong and fails to capture any correct temporal relation. Worse still, it even forms a loop. For the output of CoT, at least, it gets one temporal relation correct. However, the generated rationales are verbose, not to-the-point, and the mixture of natural language and programming language in the output might confuse the generation process as well. In contrast, the generated temporal graph by GenSort captures most of the right temporal relations, yielding a high F1 score of 80 points, and a very low GED, which is just 1.

D Meta Prompt

This section discusses the major components of a meta prompt, used to generate reference narratives. As shown in fig. A8 and fig. A9, a meta prompt consists of two parts: input (in Python programming language) and instruction (above and below the input). The input contains both $\mathcal{V}$ (event set) and $\mathcal{E}$ (temporal relation set), and the goal is to prompt LMs to generate a high-quality *reference narrative*. The input has two formats: **alphabetical** (fig. A8) format where the function header is represented in the same fashion as in fig. 2, and **descriptive** (fig. A9) where the function header is the camel-cased version of the complete event description. The instruction part specifies how LMs are supposed to carry out the narrative generation, reflecting different types and genres. Specifically, we designed four different instructions, listed in Table A3. They are *News Report*, *Simple English*, *Role Play* and *Simple Report*, which is essentially a seamless combination of *News Report* and *Simple English*.

E Faithfulness Checking Details

Table A4 shows the template being used to prompt GPT-4 to produce a judgment. GPT-4 performs a 5-way assessment: yes, largely yes, ambivalent, largely no, and no, where yes means exact alignment while no means no alignment at all. With the counting puzzle as a sanity check, we find that GPT-4 does not count the number of temporal links wrong at all. We thus rely on the returned value of *correct temporal links* as a means to determine the failure mode. Before human inspection, the distribution among yes/largely yes/largely no/no is 243/190/32/135, where GPT-4 does not output "ambivalent".

	#scenarios	#events	Max #events	#temporal links	Event length	%Non-linear	Domain	License
ProScrpt (Sakaguchi et al.)	2,077	7.46	9	6.95	4.64	39%	Daily	N/A
Schema-11 (Dror et al.)	11	7.91	11	7.18	3.48	27%	News	N/A
WikiHow Script (Lyu et al.)	291	8.37	20	7.37	9.63	0%	Daily	MIT

Table A2: Basic statistics of evaluation benchmarks. Max #events indicate the maximum number of events for a scenario. Event length is defined as the number of words in the event description. %Non-linear tells the proportion of temporal graphs that contain at least one branch. Two domains are considered, *Daily* activity and *News* journalism. "N/A" in the License column indicates that the datasets are released without a license attached.

Instruction Type	Detailed Instruction
News Report	You are provided with a set of unordered event descriptions. You are also provided with a set of event relations which instructs you how to temporally link a pair of events. They are displayed as functions defined within a python class. \n Your goal is to write a *news report* based on the provided event descriptions and event relations set. The generated *news report* should adhere to the non-fiction genre. Meanwhile, the generated *news report* should honor the provided temporal information. \n
Simple English	You are provided with a set of unordered event descriptions. You are also provided with a set of event relations which instructs you how to temporally link a pair of events. They are displayed as functions defined within a python class. \n Your goal is to write a *simple and concise story* based on the provided event descriptions and event relations set. The generated *story* should be simple such that it can be understood by a 10-year-old child, and it should be concise such that it can be written within a short paragraph. Meanwhile, the generated *story* should honor the provided temporal information. \n
Role Play	You are provided with a set of unordered event descriptions. You are also provided with a set of event relations which instructs you how to temporally link a pair of events. They are displayed as functions defined within a python class. \n Your goal is to write a *simple and concise story* based on the provided event descriptions and event relations set. The generated *story* should honor the provided temporal information. \n Now, imagine you are a character in the *story*. Let's write a *story* that clearly depicts how you, as a character, experience the events, and how you react to them.
Simple Report	You are provided with a set of unordered event descriptions. You are also provided with a set of event relations which instructs you how to temporally link a pair of events. They are displayed as functions defined within a python class. \n Your goal is to write a *simple and concise report* based on the provided event descriptions and event relations set. The generated *report* should be simple such that it can be understood by a 10-year-old child, and it should be concise such that it can be written within a short paragraph. Meanwhile, the generated *report* should honor the provided temporal information. \n

Table A3: Detailed instruction for different meta prompt type, a.k.a., instruction type.

Faithfulness Checking Manual Inspection We notice that there are 39 cases where the value of correct temporal links is 0, and 5 cases where GPT-4 refuses to produce a value. Thus, we manually look into these 44 cases. Among these 44 cases, we correct 4 of them. In one case, GPT-4's rationale is "Additionally, all other links, despite being in the correct order, are rendered incorrect due to the initial incorrect link." and GPT-4 marks 0 correct temporal links. However, as GPT-4 has discovered, all except for one link are actually correct, so we change the label from "no" to "yes". There are three cases where GPT-4 is not judging the faithfulness but instead the *correctness*. As we have noted in the main content, faithfulness is not the same as correctness. For example, one rationale is "Given the fundamental logical error in the sequence of dialing and answering, all links are considered incorrect in the context of real-world logic, despite matching the narrative's order" where the narrative mistakenly says "dialing the phone" happens after "answer the phone", so GPT-4 marks "no". Yet, as GPT-4 has also discovered that the temporal graph actually perfectly matches the generated narrative, we thus correct the label from "no" to yes. The aforementioned two cases are the ones where GPT-4 got stuck in this assessment task.

After human inspection, the final adjudicated distribution is 247/190/32/131. This leads to an alignment level of 72.8% where we consider both "yes" and "largely yes" as entailing *alignment*.

The temporal graph is represented as a list of tuples, where each tuple contains two events. The first event happens before the second event, connected with ’->’.\n Your task is to determine whether the narrative is faithful to the temporal graph. The faithfulness is solely determined by whether the temporal relations in the temporal graph *honor* the chronological order among events in the narrative.\n How to make an assessment: If the temporal graph is completely faithful to the narrative, type ’yes’. If largely faithful with minor mistakes, type ’largely yes’. If largely not faithful with only a few temporal relations captured, type ’largely no’. If completely not faithful, type ’no’. For other cases, type ’ambivalent’.\n Your response should be in the following format:\n\n ’’’ Answer: yes/largely yes/ambivalent/largely no/no Rationale: <your rationale> Temporal links: <count the number of temporal links in the graph> Correct temporal links: <determine the number of *correct* temporal links> ’’’ Let’s start! Scenario: [SCENARIO] Events: [EVENTS] Narrative: [NARRATIVE] Temporal Graph: [TEMPORAL GRAPH]

Table A4: Template used to prompt GPT-4 for self-faithfulness checking. [·] are placeholders that will be replaced with real contents to be examined when prompted. <·> are also placeholders but are used to instruct GPT-4 what the output format should look like.

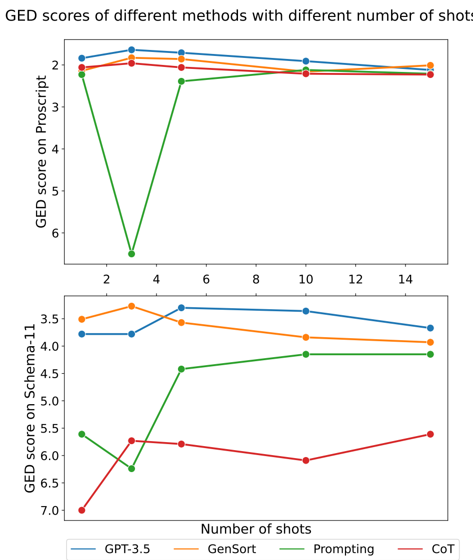

Figure A1: GED scores on ProScript (top) and Schema-11 (bottom) in relation to the number of shots in demonstrations. We identify the **instability** in the standard prompting, and the **performance plateau** after 5 shots, along with a slight decline with even more shots.

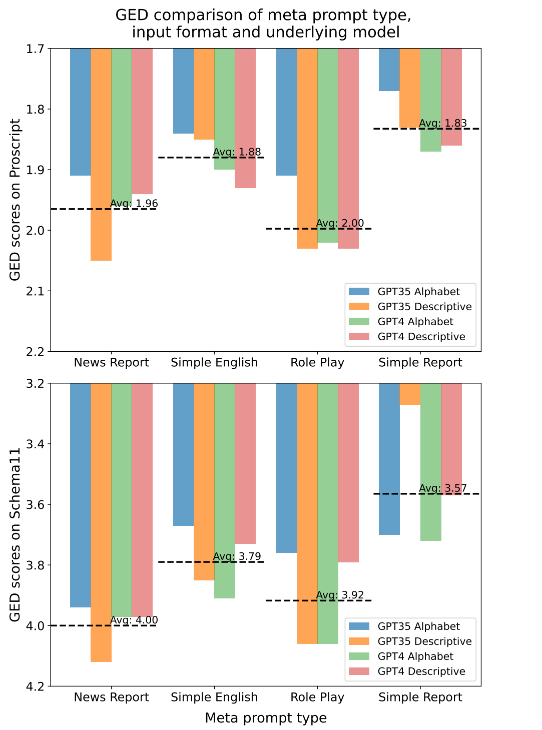

Figure A2: GED scores on ProScript (top) and Schema-11 (bottom) with different meta prompts. Notably, a *Simple Report*-style, GPT-4 generated narrative leads to the best performance due to its **conciseness**, **simplicity** and **factuality**, which are essential qualities of a *high-quality* reference narrative.

```
# *** Complete the class "BusinessChange" by
implementing "get_relations()" function
marked by #TODO. You should *ONLY* implement
the function "get_relations()" and not
generate anything else. Don't generate the
entire class "BusinessChange". Don't
generate comments. Your response must end in
"# END".
# *** You are first given a set of
demonstrations of how to implement the
"get_relations()" function for different
classes.
class WalkIntoStore:
    title = "walk into store"
    steps = 9
    def stepE(self):
        return "stop for red lights and stop
signs"
    def stepC(self):
        return "shut car door and press lock
button"
    def stepH(self):
        return "get in car and go to store"
    def stepG(self):
        return "pull into store driveway"
    def stepA(self):
        return "park the car"
    def stepB(self):
        return "take the key out of the
ignition"
    def stepD(self):
        return "get out of the car"
    def stepI(self):
        return "walk into store"
    def stepF(self):
        return "push gas pedal to move
vehicle"
    def get_relations(self):
        return [
            "stepF -> stepE",
            "stepE -> stepG",
            "stepG -> stepA",
            "stepB -> stepD",
            "stepA -> stepB",
            "stepD -> stepC",
            "stepC -> stepI",
            "stepH -> stepF",
        ]
# END
# *** Complete the class "BusinessChange" by
implementing "get_relations()" function
marked by #TODO. You should *ONLY* implement
the function "get_relations()" and not
generate anything else. Don't generate the
entire class "BusinessChange". Don't
generate comments. Your response must end in
"# END".

class BusinessChange:
    title = "business change"
    steps = 6
    def stepC(self):
        return "offer acquisition deal"
    def stepF(self):
        return "companies reach a deal"
    def stepE(self):
        return "companies merge"
    def stepD(self):
        return "companies negotiate"
    def stepA(self):
        return "government approve the deal"
    def stepB(self):
        return "company plans on
acquisition"
    def get_relations(self):
        #TODO
# END
```

Figure A3: Input for Standard Prompting with 1-shot demonstration. The input for CoT is almost identical to this one, except for an additional comment "Let's think step by step" added right above `get_relations(self)`

```
# *** Complete the class "BusinessChange" by
implementing "get_relations()" function
marked by #TODO. You should *ONLY* implement
the function "get_relations()" and not
generate anything else. Don't generate the
entire class "BusinessChange". Don't
generate comments. Your response must end in
"# END".
# *** You are first given a set of
demonstrations of how to implement the
"get_relations()" function for different
classes.
class WalkIntoStore:
    title = "walk into store"
    steps = 9
    def stepE(self):
        return "stop for red lights and stop
signs"
    def stepC(self):
        return "shut car door and press lock
button"
    def stepH(self):
        return "get in car and go to store"
    def stepG(self):
        return "pull into store driveway"
    def stepA(self):
        return "park the car"
    def stepB(self):
        return "take the key out of the
ignition"
    def stepD(self):
        return "get out of the car"
    def stepI(self):
        return "walk into store"
    def stepF(self):
        return "push gas pedal to move
vehicle"
    #Let's think about a narrative to link
aforementioned events in the correct
temporal order.
    def get_narrative(self):
        return "This is a report about
walking into a store. First, someone gets in
the car and starts to go to the store. While
driving, they push the gas pedal to move the
vehicle but stop for red lights and stop
signs along the way. After safely navigating
the roads, they pull into the store's
driveway and then park the car. Once the car
is parked, the key is taken out of the
ignition. Next, the person gets out of the
car and shuts the car door, pressing the
lock button to ensure the car is locked.
Finally, they walk into the store. By
adhering to the provided temporal
information, the desired goal is achieved."
    def get_relations(self):
        return [
            "stepF -> stepE",
            "stepE -> stepG",
            "stepG -> stepA",
            "stepB -> stepD",
            "stepA -> stepB",
            "stepD -> stepC",
            "stepC -> stepI",
            "stepH -> stepF",
        ]
# END

# *** Complete the class "BusinessChange" by
implementing "get_narrative()" and
"get_relations()" functions marked by #TODO.
"get_narrative()" serves as an auxiliary
function facilitating the temporal cohesion
of events. Essentially, it helps ensure the
temporal accuracy of the predicted temporal
graph produced in "get_relations()", by
explicitly constructing a coherent,
temporally correct story involving all
provided events.
# You should *ONLY* implement the function
"get_narrative()" and "get_relations()", but
not generate anything else. Don't generate
the entire class "BusinessChange". Don't
generate comments. Your response must end in
"# END".
class BusinessChange:
    title = "business change"
    steps = 6
    def stepC(self):
        return "offer acquisition deal"
    def stepF(self):
        return "companies reach a deal"
    def stepE(self):
        return "companies merge"
    def stepD(self):
        return "companies negotiate"
    def stepA(self):
        return "government approve the deal"
    def stepB(self):
        return "company plans on
acquisition"
    #Let's think of a narrative to link
aforementioned events in the correct
temporal order.
    def get_narrative(self):
        #TODO

    def get_relations(self):
        #TODO
# END
```

Figure A4: Input for GENSORT with 1-shot demonstration including a high-quality reference narrative.

```
[
    "stepA -> stepB",
    "stepB -> stepD",
    "stepD -> stepE",
    "stepE -> stepF",
    "stepF -> stepC",
    "stepC -> stepA",
]# END
```

Figure A5: Output by Standard Prompting.

```
Here is the implementation of the
"get_relations()" function for the
"BusinessChange" class:

    def get_relations(self):
        return [
            "stepA -> stepB",
            "stepB -> stepD",
            "stepD -> stepC",
            "stepC -> stepF",
            "stepF -> stepE",
            "stepE -> stepA",
        ]

Rationale:

The "BusinessChange" class represents the
steps involved in a business acquisition.
The steps are:
1. Company plans on acquisition (stepA)
2. Company plans on acquisition (stepB)
3. Companies negotiate (stepD)
4. Offer acquisition deal (stepC)
5. Companies reach a deal (stepF)
6. Companies merge (stepE)

The relations between these steps are as
follows:
- StepA leads to stepB, as the company plans
on acquisition and then decides on the
acquisition.
- StepB leads to stepD, as the company plans
on acquisition and then starts negotiating
with the target company.
- StepD leads to stepC, as the companies
negotiate and then offer an acquisition
deal.
- StepC leads to stepF, as the companies
offer an acquisition deal and then reach a
deal.
- StepF leads to stepE, as the companies
reach a deal and then merge.
- StepE leads to stepA, as the companies
merge and then the government approves the
deal.

These relations represent the logical flow
of events in a business acquisition process.
```

Figure A6: Output by CoT.

```
def get_narrative(self):
    return "This is a report about 'business
change'. First, companies plan on an
acquisition. Then, they offer an acquisition
deal to the other company. The other company
accepts the deal and the two companies start
negotiating the terms of the deal. After
they reach an agreement, they submit the
deal to the government for approval. Once
the government approves the deal, the
companies can merge. By adhering to the
provided temporal information, the desired
goal is achieved."

def get_relations(self):
    return [
        "stepB -> stepC",
        "stepC -> stepD",
        "stepD -> stepF",
        "stepF -> stepA",
        "stepA -> stepE",
    ]
# END
```

Figure A7: Output by GENSORT.

```
You are provided with a set of unordered
event descriptions.
You are also provided with a set of event
relations which instructs you how to
temporally link a pair of events.
They are displayed as functions defined
within a python class.

Your goal is to write a *simple and concise
report* based on the provided event
descriptions and event relations set.
The generated *report* should be simple such
that it can be understood by a 10-year-old
child, and it should be concise such that it
can be written within a short paragraph.
Meanwhile, the generated *report* should
honor the provided temporal information.

'''
class WalkIntoStore:

    title = "walk into store"
    steps = 9

    def stepE(self):
        return "stop for red lights and stop
signs"

    def stepC(self):
        return "shut car door and press lock
button"

    def stepH(self):
        return "get in car and go to store"

    def stepG(self):
        return "pull into store driveway"

    def stepA(self):
        return "park the car"

    def stepB(self):
        return "take the key out of the
ignition"

    def stepD(self):
        return "get out of the car"

    def stepI(self):
        return "walk into store"

    def stepF(self):
        return "push gas pedal to move
vehicle"

    def get_relations(self):
        return [
            "stepF -> stepE",
            "stepE -> stepG",
            "stepG -> stepA",
            "stepB -> stepD",
            "stepA -> stepB",
            "stepD -> stepC",
            "stepC -> stepI",
            "stepH -> stepF",
        ]
'''
Start your generation with "This is a report
about walk into store".
End your generation with this sentence: By
adhering to the provided temporal
information, the desired goal is achieved.
```

Figure A8: Meta prompt used to generate reference narrative, where the input format **alphabetical** and the meta prompt type is *Simple Report*.

```
You are provided with a set of unordered
event descriptions.
You are also provided with a set of event
relations which instructs you how to
temporally link a pair of events. Note, for
example, "turnOffLight -> leaveClassroom"
indicates that turnOffLight *must* happen
before leaveClassroom. Observing the
provided temporal relations is imporant!
They are displayed as functions defined
within a python class.

Your goal is to write a *news report* based
on the provided event descriptions and event
relations set.
The generated *news report* should adhere to
the non-fiction genre.
Meanwhile, the generated *news report*
should honor the provided temporal
information.
'''
class WalkIntoStore:

    title = "walk into store"
    steps = 9

    def stopForRedLightsAndStopSigns(self):
        return "stop for red lights and stop
signs"

    def shutCarDoorAndPressLockButton(self):
        return "shut car door and press lock
button"

    def getInCarAndGoToStore(self):
        return "get in car and go to store"

    def pullIntoStoreDriveway(self):
        return "pull into store driveway"

    def parkTheCar(self):
        return "park the car"

    def takeTheKeyOutOfTheIgnition(self):
        return "take the key out of the
ignition"

    def getOutOfTheCar(self):
        return "get out of the car"

    def walkIntoStore(self):
        return "walk into store"

    def pushGasPedalToMoveVehicle(self):
        return "push gas pedal to move
vehicle"

    def get_relations(self):
        return [
            "pushGasPedalToMoveVehicle ->
stopForRedLightsAndStopSigns",
            "stopForRedLightsAndStopSigns ->
pullIntoStoreDriveway",
            "pullIntoStoreDriveway ->
parkTheCar",
            "takeTheKeyOutOfTheIgnition ->
getOutOfTheCar",
            "parkTheCar ->
takeTheKeyOutOfTheIgnition",
            "getOutOfTheCar ->
shutCarDoorAndPressLockButton",
            "shutCarDoorAndPressLockButton -
> walkIntoStore",
            "getInCarAndGoToStore ->
pushGasPedalToMoveVehicle",
        ]
'''
Start your generation with "This is a report
about walk into store".
End your generation with this sentence: By
adhering to the provided temporal
information, the desired goal is achieved.
```

Figure A9: Meta prompt used to generate reference narrative, where the input format **descriptive** and the meta prompt type is *News Report*.