Cascading Large Language Models for Salient Event Graph Generation

Anonymous ACL submission

Abstract

Generating event graphs from long documents is challenging due to the inherent complexity of multiple tasks involved such as detecting events, identifying their relationships, and reconciling unstructured input with structured graphs. Recent studies typically consider all events with equal importance, failing to distinguish salient events crucial for understanding narratives. This paper presents CALLMSAE, a CAscading Large Language Model framework for SAlient Event graph generation, which leverages the capabilities of LLMs and eliminates the need for costly human annotations. We first identify salient events by prompting LLMs to generate summaries, from which salient events are identified. Next, we develop an iterative code refinement prompting strategy to generate event relation graphs, removing hallucinated relations and recovering missing edges. Fine-tuning contextualised graph generation models on the LLM-generated graphs outperforms the models trained on CAEVO-generated data. Experimental results on a human-annotated test set show that the proposed method generates salient and more accurate graphs, outperforming competitive baselines. [1]

1 Introduction

Events are fundamental discourse units which form the backbone of human communication. They are interconnected through various event relations such as hierarchical, temporal, or causal relations. Event relation graphs are vital for representing and understanding complex event narratives, with nodes representing events and edges denoting relationships between them. High-quality event relation graphs can enhance numerous downstream tasks, such as question answering (Lu et al., 2022) and reasoning (Melnyk et al., 2022).

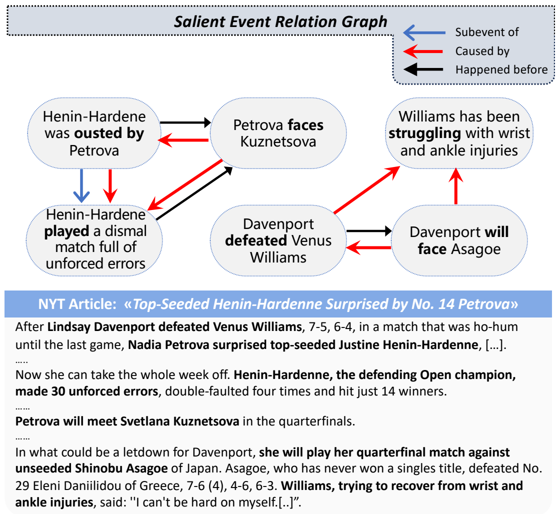


Figure 1: An example of salient event relation graph (top) generated from the NYT article (bottom).

Recent studies on contextualised event graph generation have focused on fine-tuning language models to generate linearised graphs from documents in an end-to-end manner (Madaan and Yang, 2021; Tan et al., 2024a). These methods rely on distant supervision, such as events and event temporal relations detected using an approach called CAEVO (McDowell et al., 2017), due to the data-intensive nature of language models and heavy manual efforts of annotating event graphs. However, CAEVO has limitations. It typically considers predicates (e.g., verbs) in text as events and tends to extract many insignificant events, such as "*say*" and "*think*", which add little value to narrative understanding and have minimal connections to other events, thus introducing noise to the event graphs.

To improve the quality of distant supervision graphs, it is essential to consider the saliency of events. We found that CAEVO-extracted events often have low saliency because CAEVO takes a bottom-up approach to event extraction, classifying each predicate as an event or not. In contrast, identifying salient events requires a top-down approach.

[1] Source code and dataset will be released upon paper acceptance.

Existing studies on identifying salient events or entities use the *summarisation test* to guide human annotation, where an event or entity is considered salient if a human-written summary is likely to include it (Dunietz and Gillick, 2014; Liu et al., 2018). Given that instruction fine-tuned LLMs perform on par with human writers in news summarisation (Zhang et al., 2024), we propose generating salient events by instructing LLMs to first summarise documents before identifying salient events.

Moreover, we extend beyond the CAEVO's temporal-only relations to encompass multiple relation types. We introduce iterative refinement prompting in a code prompt format to generate event relation graphs that include hierarchical, temporal, and causal relations (see Figure 1). The prompting framework is highly efficient because the code prompt format generates each type of relation graph in a single pass, while the naive prompting method needs to query each possible event pair individually. The iterative refinement process further enhances the accuracy of event relation predictions by using a hallucination grader to filter out unfaithful edges and iterative generation to recover missing edges.

Using the LLM-generated dataset, we fine-tune Flan-T5 following the same method as Tan et al. (2024a). However, the abstractive nature of salient events poses challenges for evaluation, as salient events rarely exactly match the gold standards despite having the same semantic meaning. To address this, we propose an evaluation metric based on semantic text embeddings for assessing the event relation graphs. Our experimental results on the New York Times corpus (Sandhaus, 2008) show that CALLMSAE, a novel CAscading Large Language Model framework for SAlient Event graph generation, outperforms the baselines in terms of event saliency and edge quality. The fine-tuned model surpasses previous models trained with CAEVO-generated graphs. Our contributions are summarised as follows:

- We propose CALLMSAE, a CAscading Large Language Model framework for SAlient Event graph generation, serving as a distant signal generator for contextualised graph generation models. We also propose a novel contextualised evaluation metric for comparing salient event graphs.
- We provide a large-scale LLM-generated salient event graph dataset ($10,247$ documents) with three relation types for distant supervision, along with a human-annotated test set (100 documents).
- We present an extensive experimental evaluation on LLM-generated event relation graphs in terms of event saliency and event relation on the NYT corpus, demonstrating how higher quality salient event graphs can improve contextualised graph generation.

2 Related Work

Event Relation Graph Construction The early idea of event relation graph construction comes from UzZaman et al. (2013), which introduces a dataset for evaluating an end-to-end system which takes raw text as input and output TimeML annotations (i.e., temporal relations). CAEVO (McDowell et al., 2017) and Cogcomptime (Ning et al., 2018) both utilise a wide range of manually designed features to train MaxEnt and averaged perception for extracting events and relations. Han et al. (2019b) proposed a joint event and relation extraction model based on BERT (Devlin et al., 2019) and BiLSTM (Panchendrarajan and Amaresan, 2018). Other researchers focus on developing specialised sub-systems to classify extracted event pairs for relations (Ning et al., 2019; Han et al., 2019a; Wang et al., 2020; Tan et al., 2021). ATOMIC (Sap et al., 2019) is a large-scale commonsense knowledge graph containing the causes and effects of events. MAVEN-ERE (Wang et al., 2022) is built with event coreference, temporal, causal and subevent relations. However, ATOMIC and MAVEN-ERE completely rely on crowdsourcing and thus are difficult to extend. MAVEN-ERE is less than half the size of our dataset and does not consider the saliency of events.

Madaan and Yang (2021) fine-tune GPT-2 to generate linearised graphs from documents in an end-to-end manner. Their temporal relation graphs used for training are produced by CAEVO. Following this direction, Tan et al. (2024a) instead view the task as set generation and propose a framework based on set property regularisation and data augmentation. In this paper, we focus on generating multi-relation graphs via in-context learning, prompt interaction, and iterative refinement.

Salient Event Identification Several existing papers investigate the problem of identifying salient events. Choubey et al. (2018) build a rule-based classifier to identify central events by exploiting human-annotated event coreference relations. They find the central events either have large numbers of coreferential event mentions or have large stretch sizes. Jindal et al. (2020) propose a contextual model to identify salient events based on BERT and BiLSTM. They also mention several features, such as event trigger frequency, which are essential features to identify the salient events. Liu et al. (2018) propose a feature-based method using LeToR (Liu et al., 2009) and a neural-based method called Kernel-based Centrality Estimation. To train and evaluate their methods, they build a dataset based on the *summarisation test*: an event is considered salient if a summary written by a human is likely to include it. Zhang et al. (2021) combine the Kernel-based Centrality Estimation with the event and temporal relation extraction model of Han et al. (2019b) to build a salience-aware event chain modelling system. However, they only focus on single-dimensional chains and only model temporal relations.

3 Cascading LLMs to Generate Salient Event Graphs

CALLMSAE combines various prompts in a pipelined manner to generate salient event graphs. In this section, we will first introduce the prompts for generating salient events. Then, we will describe the method for generating relation graphs based on the salient events. Lastly, we define an evaluation metric for comparing event graphs: *Hungarian Graph Similarity*.

3.1 Generate Salient Events

The *summarisation test* (as mentioned in Section 1) is often used to guide the annotation of salient events or entities (Dunietz and Gillick, 2014; Liu et al., 2018). These studies identify events or entities included in human-written summaries as salient. Similarly, we instruct LLMs to generate a summary first and then extract events from it.

3.2 Generate Graphs as Code Completion

While LLMs can extract salient events, they often struggle with identifying event relations (Chan et al., 2023; Tan et al., 2024a). Prompt engineering for extracting event relations is complex due to the need to incorporate various terminologies and graph constraints. Moreover, prompt efficiency is crucial as generating a large-scale dataset with LLMs can still incur significant computational costs, albeit less than crowdsourcing.

In our method, the main prompt for generating the event relation graph is formulated as a Python code completion task. The graph is defined using the NetworkX[2] package in Python, with nodes representing the salient events generated in Section 3.1. LLMs are instructed to complete the code by adding relation edges using NetworkX's APIs.

Recent research suggests that formulating prompts as code can enhance LLMs' reasoning abilities (Wang et al., 2023; Zhang et al., 2023). In our task, the Python code format effectively incorporates all necessary terminologies, enabling LLMs to understand them without confusion. The Python code format also allows for the inclusion of constraints (e.g., ensuring the graph is a directed acyclic graph) and additional instructions (e.g., ask for explanations) as comments. LLMs can generate explanations as comments without disrupting the main content of the graph. Moreover, the code template simplifies parsing the response, as LLMs are directed to use the ".add_edge()" function to add the relations.

Since hierarchical, temporal, and causal relations are asymmetric, each can be represented by a Directed Acyclic Graph (DAG). We formulate three distinct prompts to guide LLMs in generating three DAGs, each representing one of these relation types. This approach avoids the complexity of a multi-label graph, and LLMs can focus on a single relation type and carefully consider the topological structure of the graph. We can also use the ".find_cycle()" function from NetworkX to detect constraint violations reliably. In addition, if relation types are interconnected, the initially generated graphs can help the generation of subsequent graphs (as will be explained in Section 3.4). We provide an example of the code prompt in Appendix (Table 9).

3.3 Iterative Refinement

Hallucination Grader The code prompt efficiently guides LLMs to generate graphs, but it often generates hallucinated relations. Based on our preliminary experiments, these hallucinations stem from the models' overconfidence in their relation predictions. Specifically, LLMs tend to infer event relations without explicit linguistic cues or strong evidence for logical inference. Consequently, LLMs predict far more relations than the gold standards, leading to low precision.

[2] https://networkx.org/documentation/stable/

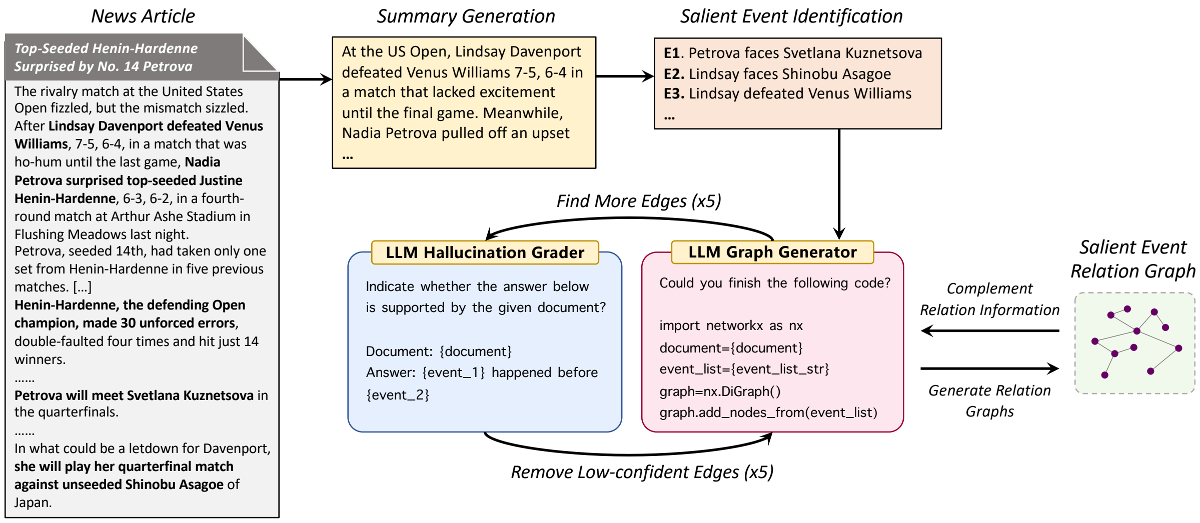


Figure 2: The proposed CALLMSAE framework.

Recent studies show that LLMs can evaluate and correct their own outputs (Madaan et al., 2023; Asai et al., 2024). Thus, we introduce a hallucination grader to address hallucination. For each relation edge generated, we pose a question to the LLMs to determine whether the relation is grounded in the given document. If the LLMs respond with a "yes", the edge is retained; otherwise, it is discarded. An example of the hallucination grader prompt is shown in Appendix (Table 10).

Recover Missing Edges The main benefit of the hallucination grader is that it increases precision by removing low-confident edges. However, this process inevitably reduces recall. To mitigate this side effect, we introduce an iterative refinement process. After discarding hallucinated edges, we reinsert the code block containing the relation edges into the graph generation prompt and ask the LLMs to complete the code again. In this way, the LLMs can reconsider whether there are any missing relations in the document, thereby improving recall.

Once the LLMs generate a new graph, the hallucination grader checks the relation edges again. This process is repeated for a fixed number of times. We set the maximum number of iterations to 5 in our experiments, as the LLMs stop discovering new edges after 2 or 3 iterations in most documents.

3.4 Complement Relation Types

Hierarchical, temporal, and causal relations are not independent of each other. We found that if one type of relation depends on another, providing the graph for the first relation can benefit the generation of the dependent relation's graph. Specifically, we predict the hierarchical relation graph first. Then, we provide this graph to the LLMs and ask them to generate the temporal relation graph. Lastly, with both the hierarchical and temporal relation graphs available, the LLMs predict the causal relation graph.

The hierarchical relation describes two closely related events at different granularity levels. It focuses on the inherent semantics of the events and does not depend on other relation types. For example, "*writing a dissertation*" is a subevent of "*doing a PhD*". Therefore, we choose to predict the hierarchical relations first.

Temporal relations can depend on hierarchical relations. For example, knowing "*doing a PhD*" happened before "*being prompted to Professor*" allows us to deduce that "*writing a thesis*" also happened before "*being prompted to Professor*". Thus, we predict temporal relations after hierarchical relations.

Lastly, causal relations depend on both hierarchical and temporal relation, as the antecedent event in a causal relation must occur before the consequence. Therefore, the causal relation is predicted in the last step. For more details about the entire prompting process, please refer to the descriptions and pseudocode in Appendix C.

3.5 Hungarian Graph Similarity

It is challenging to compare event relation graphs generated by LLMs due to the abstractive nature of generation, making it difficult to align the generated events with the gold standard events (Li et al., 2023). Moreover, salient events are often high-level and abstract rather than fine-grained and concrete, which means some variations in wording is not only acceptable but also expected. Instead of using exact matching (Zhao et al., 2024) or rule-based token matching (Tan et al., 2024b) on events and relations to calculate F_1, adopting semantic-based evaluation metrics is more reasonable and fair. As more tasks adopt text generation frameworks, many researchers are also turning to metrics based on language models rather than traditional token matching metrics like ROUGE and BLUE (Goyal et al., 2022; Pratapa et al., 2023).

In this study, we propose a novel metric for evaluating LLM-generated event graphs, called **Hungarian Graph Similarity (HGS)**. The metric is based on the Hungarian assignment algorithm (Kuhn, 1955), which is widely used in the object detection to match generated objects and target objects (Carion et al., 2020). It can find the optimal assignment given a cost matrix containing the distance between elements in two lists of objects. We adapt this algorithm to match predicted edges with edges in the gold standard graphs as follows:

1. Encode the events using SFR-Embedding-Mistral (Meng et al., 2024), which was ranked 1st on the Massive Text Embedding Benchemark leaderboard (Muennighoff et al., 2022) at the time of our experiments.

2. Given two edges of the same relation type, let $\bar{e}_1^h, \bar{e}_1^t$ be the embeddings of the head event and the tail event in the first edge. Let $\bar{e}_2^h, \bar{e}_2^t$ be the embeddings of the head and tail events in the second edges. We define the distance between the edges as $\max\big(D_{cos}(\bar{e}_1^h, \bar{e}_2^h), D_{cos}(\bar{e}_1^t, \bar{e}_2^t)\big)$, where $D_{cos}(\cdot, \cdot)$ is the cosine distance.

3. Build a cost matrix by computing the distance between every edge pair in the gold and predicted edge sets. Pad the matrix to a square matrix with the maximum cost value of 1.

4. Apply the Hungarian algorithm to the cost matrix to get the minimal cost value. The final score is $1 -$ cost value, making the value more intuitive (higher is better). To compute the HGS over all the documents, we weight the scores by the number of gold edges to obtain an average value.

In step 2, we take the maximum value of the distances between head and tail events because relation edges are considered matched only if both the head and tail events match.

For more detailed analysis, we define precision-oriented HGS and recall-oriented HGS. We match edges without padding the cost matrix in step 3 to obtain the total cost values of all matched edge pairs. Then, the total matched similarity is the number of matched edges minus the total cost. **Precision-oriented HGS** is computed by dividing the total matched similarity by the total number of predicted edges. **Recall-oriented HGS** is computed by dividing the total matched similarity by the total number of edges in the target graph.

4 Dataset

In this section, we describe how we construct the distant supervision dataset and a human-annotated dataset from the New York Times (NYT) corpus.

4.1 Document Selection

We follow the same procedures as in (Tan et al., 2024a) to select documents from the NYT corpus, one of the largest news datasets, with additional filtering based on document length. We select $10,347$ documents based on their descriptors indicating they are related to event narratives instead of opinions and discussions, such as sports and international politics. Among them, 100 documents are randomly sampled as the test set to be annotated by humans. More details about data selection are shown in Appendix A.1.

4.2 Annotation Settings

We recruited annotators from Prolific[3]. There are two subtasks: *salient event identification* and *event relation identification*. In the first subtask, the participants are asked to identify the salient event triplets: *actor*, *trigger*, and *object* (optional). We provide the definition of an event and several examples in the guidelines. They are instructed to do the summarisation test: the salient events should be the events they would include in the summary of the given article. Moreover, we provide some prominent features for helping annotators to identify salient events (Choubey et al., 2018; Jindal et al., 2020):

- Frequency: salient events are frequently mentioned in the articles.
- First appearance: salient events are often mentioned at the beginning of the article.

[3] http://www.prolific.com

- Stretch size: salient events are often mentioned throughout the articles. Stretch size is the distance between the location where the event is first mentioned and last mentioned. A salient event usually has a large stretch size.

In the second stage, we ask participants to identify relation triplets: *a source event*, *a relation type*, and *a target event*. Both the source and target events should be among the salient events identified in the first stage. In the guideline, we define three relation types: *happened_before*, *caused_by*, and *is_subevent_of*. *happened_before* indicates that the source event happened earlier than the target event. *caused_by* means the source event would not have happened if the target event did not happen. *is_subevent_of* signifies that the source event is a subevent of the target event. Annotators were informed that relations would be either explicitly mentioned in the article or inferred based on evidence within the article. Further details about the guidelines and user interface can be found in Appendix A.4.

4.3 Inter Annotator Agreement

Identifying salient events and event relations is complicated and time-consuming. We found it challenging to educate participants about these concepts because, in daily life, the meanings of events and relations differ from their definitions in the field of information extraction. Moreover, the technical definitions are much less intuitive to those outside the academic field. As a result, thorough training of participants is important to obtain high-quality annotations.

In total, we recruited 3 annotators to annotate 100 documents. Due to their varying availability, annotator 1 and 2 each annotated 45 documents, while annotator 3 annotated 20 documents. Among these, 5 documents were annotated by all three annotators. Following prior research in information extraction (Gurulingappa et al., 2012; Zhao et al., 2024), we used F_1 to measure the inter-annotator agreement on these 5 documents. To compute inter-annotator agreement, events or relations identified by one annotator are represented as set S_1. Another annotator's annotation S_2 serves as a pseudo-reference to compute precision $= \frac{|S_1 \cap S_2|}{|S_1|}$, recall $= \frac{|S_1 \cap S_2|}{|S_2|}$, and the F_1 score $= \frac{2|S_1 \cap S_2|}{|S_1|+|S_2|}$.

Table 1 shows the agreement scores for stages 1 and 2. Identifying salient events is subjective, which makes it difficult to reach a complete agreement. Moreover, event relation identification is even more subjective and dependent on the previous stage, leading to less unanimous agreement.

Annotator	Stage 1	Stage 2
1 & 2	0.838	0.676
1 & 3	0.771	0.645
2 & 3	0.847	0.710
Average	0.819	0.677

Table 1: Inter-annotator agreement measured by F_1.

4.4 Dataset Statistics

Table 2 shows the distributions of the relation types after applying the transitive closure to the annotated graphs. *happened_before* emerges as the most frequent relation type, reflecting the predominant focus on temporal sequences in news articles, and they are relatively straightforward to identify. Conversely, *caused_by* is the least frequent as it is the most challenging to identify.

Relation Type	Number
happened_before	310
caused_by	202
is_subevent_of	245
Total	757

Table 2: The distributions of the relation types.

5 Experiments

5.1 Model Settings

We compare our proposed approach with the following baselines:

- CAEVO (McDowell et al., 2017) is a pipeline system based on a Maximum Entropy (MaxEnt) classifier and manually designed features for extracting events and temporal relations.

- Madaan and Yang (2021) trained language models on CAEVO-generated linearised graphs with the language modelling objective. We implemented their method to train a Flan-T5 model.

- Tan et al. (2024a) also trained language models on CAEVO-generated graphs, but applied data augmentations and regularisations to mitigate the set element misalignment issue. We applied their method to train a Flan-T5.

- Han et al. (2019b) proposed a joint event and temporal relation extraction model. We adapted the model to predict hierarchical and causal relations by training it on the MAVEN-ERE dataset (Wang et al., 2022). We also replaced BERT with Longformer (Beltagy et al., 2020) to enable it to process long documents.

- GPT-3.5 is an LLM based on the generative pre-train framework[4]. We used "gpt-3.5-turbo".

- GPT-4 is also an LLM based on the generative pre-train framework (OpenAI et al., 2024). We used "gpt-4-1106-preview".

- MIXTRAL is an LLM based on the Mistral model and the mixture of expert framework. We used the Mixtral 8x7B instruct version (Jiang et al., 2024).

- LLAMA3 is an LLM based on the Llama framework[5]. We used the Llama3-70B-instruct 8-bit version provided by ollama[6]. The 8-bit quantization is shown to be degradation-free (Dettmers et al., 2022).

We fine-tuned a Flan-T5-base (250M) with the relation graphs generated by CALLMSAE, following the same method as in Tan et al. (2024a). The baseline prompt evaluates whether each event pair is supported by the document, akin to the hallucination grader described in Section 3.3. Thus, it serves as an ablation of our method without incorporating the code prompt.

CALLMSAE is designed to be model-agnostic. Due to budget constraints and the preliminary test results, we chose Llama3 as the backbone of all the prompt-based methods detailed in Table 5.

5.2 Event Saliency Evaluation

Table 3 shows the salient features (defined in Section 4.2, computation formulas in Appendix B) extracted from various backbone LLMs using summarisation prompts, alongside comparison with CAEVO and human annotations. The LLM-generated events are much more salient than CAEVO-generated events and exhibit similarity to human annotations.

We also use human annotations to evaluate the saliency. In the salient event identification annotation, we provide the events generated by CAEVO and Mixtral as candidate salient events. Note that only the top CAEVO events ranked in saliency features are shown. Half of the candidates are from CAEVO and the other half are from Mixtral. They are randomly shuffled and then shown to the annotators. We compute the precision, recall, and F_1 based on how the annotators select them. We also compute HGS using human-annotated salient events as references (Table 4). It is clear that although CAEVO extracted more events than Mixtral, many of them are not salient. Mixtral outperforms CAEVO significantly across all evaluation metrics.

[4]https://platform.openai.com/docs/models/gpt-3-5-turbo

[5]https://ai.meta.com/blog/meta-llama-3/

[6]https://ollama.com/library/llama3:70b-instruct-q8_0

	Mean event number	Event frequency ↑	First appearance ↓	Stretch Size ↑
CAEVO	34.71	0.05	0.46	0.07
Human	8.26	0.11	0.31	0.20
GPT-4	6.49	0.09	0.37	0.18
Llama3	5.17	0.09	0.30	0.19
Mixtral	10.60	0.10	0.33	0.20

Table 3: The average number of extracted events and the saliency features (in percentage values).

	P	R	F_1	HGS
CAEVO	3.29	3.72	3.49	18.18
Mixtral	48.97	56.77	52.59	67.15

Table 4: Precision, recall, and F_1 based on the choices of the annotators. Hungarian graph similarity (HGS) is defined in Section 3.5. The values are in percentage.

5.3 Salient Event Relation Graph Evaluation

The salient event relation graph evaluation results are shown in Table 5. Even with the most basic prompting (*Baseline Prompt*), which queries the relation of each event pair, Llama3 outperforms all the baseline methods on all relation types. However, *Baseline Prompt* is slow and costly because the number of prompts it needs for building one graph is $O(n^2)$, where n is the number of events in the document. On the other hand, *Code Prompt* only needs $O(1)$. Moreover, *Code Prompt*'s overall HGS is significantly higher than *Baseline Prompt* on all relation types. *Baseline Prompt* check the event pairs more thoroughly and thus have higher recall but its precision is much lower. The complete CALLMSAE combines the code prompt and hallucination grader for iterative refinement, checking missing relations and verifying them to prevent hallucination. It significantly increases the precision and strikes a balance with recall.

	Hierarchical			Temporal			Causal		
	PHGS	*RHGS*	*HGS*	*PHGS*	*RHGS*	*HGS*	*PHGS*	*RHGS*	*HGS*
Han et al. (2019b)	0.158	0.247	0.098	0.092	0.352	0.148	0.084	0.316	0.116
CAEVO	-	-	-	0.030	0.558	0.092	-	-	-
Madaan and Yang (2021)	-	-	-	0.061	0.439	0.116	-	-	-
Tan et al. (2024a)	-	-	-	0.126	0.335	0.187	-	-	-
Baseline Prompt	0.076	0.651	0.248	0.085	0.627	0.195	0.062	0.657	0.207
Code Prompt	0.174	0.559	0.315	0.153	0.678	0.283	0.121	0.632	0.272
Code Prompt (dependent rels)	0.196	0.544	0.334	0.211	0.601	0.341	0.135	0.599	0.272
CALLMSAE (ours)	0.196	0.544	0.334	0.294	0.509	0.327	0.198	0.529	0.295
Fine-tuned T5 (CALLMSAE)	0.314	0.434	0.339	0.244	0.544	0.362	0.366	0.397	0.343

Table 5: The Hungarian graph similarity (HGS) of the LLM-generated graphs on the human-annotated NYT dataset. PHGS is precision-oriented HGS. RHGS is recall-oriented HGS. *Code Prompt (dependent rels)* means adding hierarchical graphs in the prompts for temporal graphs; and adding hierarchical and temporal for causal graphs. *Fine-tuned T5 (CALLMSAE)* means fine-tuning a flan-T5 using the graphs generated by CALLMSAE.

In the temporal category, the results of *Code Prompt (dependent rels)* are obtained when provided with hierarchical graphs generated by CALLMSAE to LLMs. It has much higher overall HGS and precision than *Code Prompt* without hierarchical information, showing that hierarchical information can mitigate hallucinations during the temporal graph generation. In the casual category, the results of *Code Prompt (dependent rels)* are obtained when given both hierarchical and temporal graphs generated by CALLMSAE. The additional information also increases precision.

Fine-tuned T5 outperform all the methods based on CAEVO (McDowell et al., 2017; Madaan and Yang, 2021; Tan et al., 2024a), showing that the high-quality graphs generated by CALLMSAE can boost the contextualised graph generation. Interestingly, the performance of the *Fine-tuned T5*, fine-tuned on CALLMSAE-generated data, exceeds that of CALLMSAE itself, implying that the fine-tuned model can effectively adapt the reasoning patterns provided by Llama3 and generalise them.

5.4 Format Error and Cycles in the Graphs

A format error occurs when the generated code blocks fail to pass the Python interpreter. We detected these errors by executing the generated code. If the Python interpreter returns an error, it is classified as a format error. We specified the relation graphs as directed acyclic graphs in the prompt. If there is a cycle in the generated graph, it means that the LLM failed to follow the instructions. A cycle also indicates a violation of logic constraints because all the relations in the event relation graphs are asymmetric. We detected the cycles using the find_cycle() from the NetworkX after obtaining the transitive closure of the graphs.

	Format Error	Cycle
GPT-3.5	0	10.67
GPT-4	3.67	1.67
Mixtral	3.33	2.33
Llama3	0	0

Table 6: The average number of CALLMSAE-generated graphs out of 100 with format errors or cycles.

We prompt each LLM three times on the annotated test set. Table 6 shows the average number of documents encountering format errors or cycles. All LLMs have low rates of format errors which shows that state-of-the-art LLMs can understand the instruction well and generate executable Python code. Among them, GPT-3.5 and Llama3 achieve zero errors. The occurrence of cycles can serve as an indicator of the reasoning ability of the LLMs. About 10% of graphs generated by GPT-3.5 have cycles, suggesting that GPT-3.5 may have limited reasoning ability compared to other LLMs. GPT-4 and Mixtral both have low rates of cycle occurrence, but they are beaten by Llama3 which has no cycle in all generations, showing its remarkable understanding of the transitive and asymmetric constraints in the complex event relation graphs.

6 Conclusion

This study explored utilising LLMs to generate salient event relation graphs from news documents without relying on human annotations. We studied how the events generated by LLMs are compared to the traditional methods in terms of event saliency. We further demonstrated that CALLMSAE-generated graphs can serve as distant signals to fine-tune smaller models and outperform those based on CAEVO.

Limitations

Although we have tested many prompting methods and included several of the most effective ones in this paper, we have not explored all possible combinations due to the extensive volume of recent literature on prompt engineering. There might still exist combinations of prompts that could further improve performance. However, we are almost certain that any potential combinations, if they exist, are likely to be more complex and thus less efficient for building large-scale datasets. For example, we did not add demonstrations in graph generation because the code template is already quite lengthy. Adding more documents could potentially exceed the context windows of some LLMs, making it challenging for them to interpret the instructions effectively.

Ethics Statement

Event relation graph generation is a powerful tool for understanding text. A potential misuse of the proposed method is mining user behaviours on their private data. For example, salient event relation graphs can be extracted from users' tweets to analyse their potential reactions to advertisements and scams. That could be a huge risk to social media users.

Another potential risk is that the saliency may introduce bias. LLMs may have their preferences in selecting a specific group of events as important events due to the data they were trained on. This is a question which requires further large-scale investigation. However, we think this risk is negligible in this study because we work on document-level information. There is little room for selection given that the news articles are already the products of choice and distillation. If the system is used to extract information from a border information source, such as social media, the risk must be carefully assessed.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Chunkit Chan, Jiayang Cheng, Weiqi Wang, Yuxin Jiang, Tianqing Fang, Xin Liu, and Yangqiu Song. 2023. Chatgpt evaluation on sentence level relations: A focus on temporal, causal, and discourse relations.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 340–345, New Orleans, Louisiana. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

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.

Jesse Dunietz and Daniel Gillick. 2014. A new entity salience task with millions of training examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 205–209, Gothenburg, Sweden. Association for Computational Linguistics.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892.

Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph Weischedel, and Nanyun Peng. 2019a. Deep structured neural network for event temporal relation extraction. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 666–106, Hong Kong, China. Association for Computational Linguistics.

Rujun Han, Qiang Ning, and Nanyun Peng. 2019b. Joint event and temporal relation extraction with shared representations and structured prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 434–444, Hong Kong, China. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Disha Jindal, Daniel Deutsch, and Dan Roth. 2020. Is killed more significant than fled? a contextual model for salient event detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 114–124, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard Hovy. 2018. Automatic event salience identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1226–1236, Brussels, Belgium. Association for Computational Linguistics.

Junru Lu, Xingwei Tan, Gabriele Pergola, Lin Gui, and Yulan He. 2022. Event-centric question answering via contrastive learning and invertible event transformation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2377–2389, Abu Dhabi, United Arab Emirates. 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 *Thirty-seventh Conference on Neural Information Processing Systems*.

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.

Bill McDowell, Nathanael Chambers, Alexander Ororbia II, and David Reitter. 2017. Event ordering with a generalized model for sieve prediction ranking. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 843–853, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Igor Melnyk, Pierre Dognin, and Payel Das. 2022. Knowledge graph generation from text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1610–1622, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Qiang Ning, Sanjay Subramanian, and Dan Roth. 2019. An improved neural baseline for temporal relation extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6203–6209, Hong Kong, China. Association for Computational Linguistics.

Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018. CogCompTime: A tool for understanding time in natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–77, Brussels, Belgium. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Rrubaa Panchendrarajan and Aravindh Amaresan. 2018. Bidirectional LSTM-CRF for named entity recognition. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Adithya Pratapa, Kevin Small, and Markus Dreyer. 2023. Background summarization of event timelines. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8111–8136, Singapore. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times Annotated Corpus.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Xingwei Tan, Gabriele Pergola, and Yulan He. 2021. Extracting event temporal relations via hyperbolic geometry. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8065–8077, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xingwei Tan, Yuxiang Zhou, Gabriele Pergola, and Yulan He. 2024a. Set-aligning framework for auto-regressive event temporal graph generation.

Xingwei Tan, Yuxiang Zhou, Gabriele Pergola, and Yulan He. 2024b. Set-aligning framework for auto-regressive event temporal graph generation. *arXiv preprint arXiv:2404.01532*.

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.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. Joint constrained learning for event-event relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 696–706, Online. Association for Computational Linguistics.

Xiaozhi Wang, Yulin Chen, Ning Ding, Hao Peng, Zimu Wang, Yankai Lin, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. MAVEN-ERE: A unified large-scale dataset for event coreference, temporal, causal, and subevent relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 926–941, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xingyao Wang, Sha Li, and Heng Ji. 2023. 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.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023. Causal reasoning of entities and events in procedural texts. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia. Association for Computational Linguistics.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. Benchmarking Large Language Models for News Summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Xiyang Zhang, Muhao Chen, and Jonathan May. 2021. Salience-aware event chain modeling for narrative understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1418–1428, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Runcong Zhao, Qinglin Zhu, Hainiu Xu, Jiazheng Li, Yuxiang Zhou, Yulan He, and Lin Gui. 2024. Large language models fall short: Understanding complex relationships in detective narratives. *arXiv preprint arXiv:2402.11051*.

A Additional Details of Dataset Construction

A.1 Document Selection

We select news documents from the NYT corpus based on the descriptors available. With regards to the generation of salient even graphs, the most relevant documents tend to be centered around event narratives, so that they could be rich in event relations. Tan et al. (2024a) investigated which descriptors are rich in event narrative using event frequency $\times$ inverse-descriptor frequency. We chose the documents using the same descriptors as them (e.g., "*airlines and airplanes*", "*united states international relations*", "*civil war and guerrilla warfare*", "*track and field*", "*soccer*", etc.).

We applied additional filtering based on the number of words in the documents. Documents with more than 8500 words or less than 100 words are excluded. Based on our preliminary observations, the extremely long documents are not typically news articles (only takes 0.02% in the entire NYT). They tend to be collections of articles over longer time spans, making them not suitable as focus of this study. Additionally, very long articles may affect the performance of open-source LLMs only due to limitations in the context length rather than their reasoning abilities. On the other hand, articles that are too short are less likely to contain complex event relation graphs, so we also exclude them. The final average word count of the selected 10347 documents is 780.

A.2 Frequent words and descriptors in the annotated dataset

	Test		Train	
Rank	Word	Count	Word	Count
1	win	41	win	$2,964$
2	express	15	make	$1,591$
3	play	14	face	$1,564$
4	make	13	express	$1,411$
5	defeat	12	include	$1,307$

Table 7: The top 5 most frequent trigger words in the human-annotated test set and the distant train set.

Table 7 reports the most frequent trigger words among the human-identified salient events and LLM-generated salient events after filtering out the light words (words that have no semantic meaning). We could see that "*win*", "*play*", and "*defeat*" are

	Test		Train	
Rank	Descriptor	Count	Descriptor	Count
1	U.S. International Relations	27	Terrorism	2,885
2	Terrorism	21	U.S. International Relations	2,574
3	Bombs and Explosives	17	Bombs and Explosives	1,727
4	U.S. Armament and Defense	15	U.S. Armament and Defense	1,717
5	Politics and Government	15	Politics and Government	1,649

Table 8: The top 5 most frequent descriptors in the human-annotated test set and the distant train set.

prominent triggers due to the sports topics within the dataset. These articles usually mention multiple events with these triggers. Triggers like "*express*", "*include*", and "*make*" are instead common across different scenarios.

Table 8 shows the most frequent descriptors in the human-annotated test set and the distant train set. These are the typical event-rich topics and are full of narratives.

A.3 Disclaimers of Risks

Consider that a large portion of the new articles in the New York Times corpus are about violent incidences, such as terrorist attacks and war. To prevent inflicting harm to traumatised victims, we show the information clearly in the recruitment description on the Prolific platform (Figure 3).

Figure 3: The recruitment descriptions.

A.4 Guidelines and User Interface

A well-designed user interface is essential for collecting high-quality data efficiently. We fully cooperate with participants to improve the user interface iteratively based on their feedback.

In the salient event identification stage (Figure 4), we show the title, abstract, and content of the article on the right side. We show candidate events, which are extracted through CAEVO and Mixtral, on the left sidebar. The shown CAEVO events are the top events ranked based on the saliency feature score. The participants can choose the candidate events which they think are accurate and salient. The guideline also informs them that if multiple options refer to the same event, they can only choose the most accurate and informative one. If a salient event is not present among the candidates, they could write it in the text input box and add it.

In the event relation identification stage (Figure 5), they could choose a source event, a relation type, and a target event to add a relation triplet. The source event and the target event need to be chosen from the salient event list from the first stage. We automatically detect and prevent any new event that will lead to duplication and contradiction. The participants can also deselect the added event if they change their minds. The participants were asked to finish the first stage first, and then annotate the second stage based on their own annotations in the first stage.

In the following are reported the screenshots of the guideline pages (Figure 6).

A.5 More details about the annotation

We started the annotation process by releasing several trial rounds, during which we chose participants based on their dedication and understanding of the terminologies. It required considerable communication efforts to ensure they had an accurate understanding of the task definition.

During training, we found a common mistake among the annotators was that they tended to overestimate the *is_subevent_of* relation. They often confused it with the *caused_by* relation or temporal inclusion.

We advised them that *is_subevent_of* pertains to two events on different granularity levels but referring to the same subject. To distinguish *is_subevent_of* from temporal overlap, they could check whether the actor in the subevent is the same as or a part of the actor or object in the parent event. For example, if a parent event is "*a team did something*" the subevent can be "*a member of the team did something*".

A.6 Information about the Annotators

The annotators were paid at the rate of 8£/h. We screened native English speakers from all over the world to ensure they could read English articles

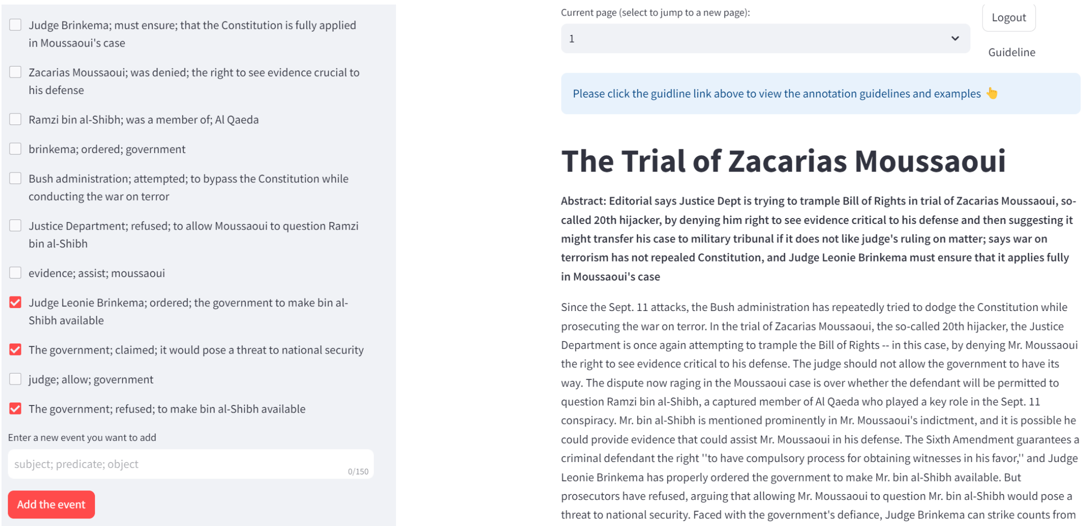


Figure 4: The user interface of salient event identification.

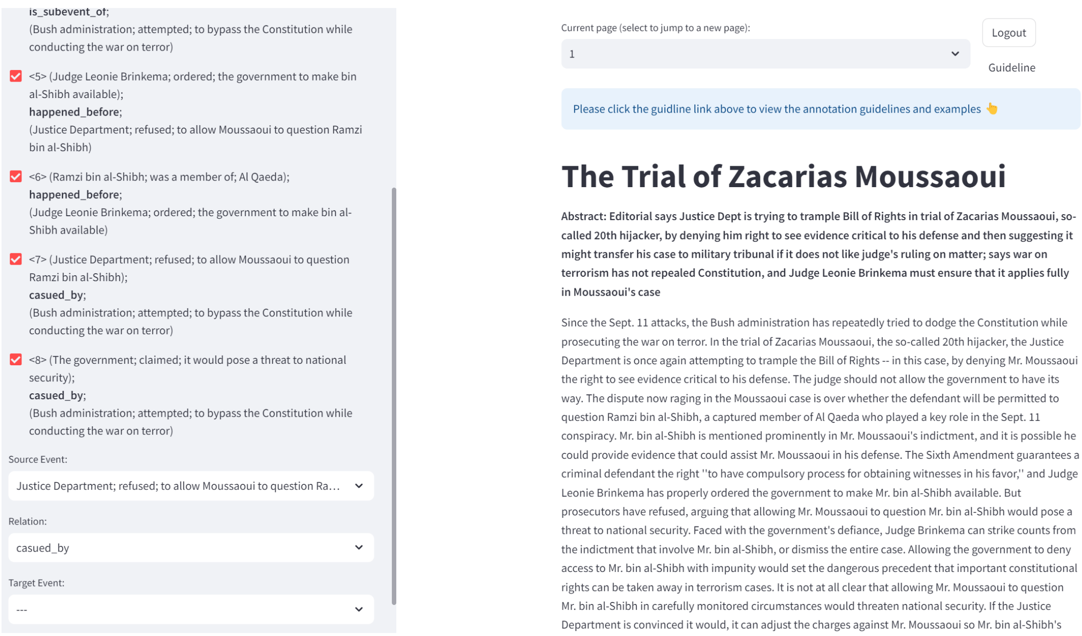


Figure 5: The user interface of event relation identification.

fluently. We also selected participants based on their previous submissions and approval rates to ensure they were familiar with the platform and were high-quality annotators.

Two of the final annotators are identified as male, and they both come from the UK. One of the final annotators is identified as female, and she comes from Canada. They all identified as white.

A.7 Dataset Licensing

The original NYT corpus is available for noncommercial research license. One of our authors has obtained the license. Based on the license, we could not include the original text in our dataset. Thus, we will only release the generated/annotated graphs. Our dataset will also be in noncommercial research license.

B Saliency Features

Inspired by (Choubey et al., 2018), we calculate the saliency features to show how our proposed method differs from previous methods in terms of event saliency. Unlike conventional computation methods, these saliency features are calculated on the sentence level to be comparable across documents of various lengths. These saliency features are:

Event frequency: A salient event tends to appear frequently in the document. Let $D = \{s_0, s_1, ..., s_{n-1}, s_n\}$ be the document and the list of sentences in the document. Let e be the event. Let $M(e) = \{s_i, s_j, ..., s_k\}, 0 \leqslant i < j < k \leqslant n$ be the list of sentences which mention the event e. The event frequency is calculated as:

$$frequency(D, e) = \frac{|M(e)|}{n+1}. \quad (1)$$

First appearance: News writers usually mention the salient event as early as possible to attract readers' attention. The first appearance of the event e is computed as:

$$first\_appearance(D, e) = \frac{i}{n}. \quad (2)$$

Stretch size: Salient events tend to be mentioned all across the document. The stretch size of event e is calculated as:

$$stretch\_size(D, e) = \frac{k-i}{n}. \quad (3)$$

To detect which sentences mention the event e, we first lemmatise the words in the document and the given event. Then, detect whether there is a matched substring the same as the given event in each sentence. However, the abstractive nature of LLM-based salient event generation makes exact matching not viable. To detect the event mention of LLM-generated events, we formulate a series of prompts. We first ask: “*Which sentence in the document below mentions the event "{event}"? Please enclose that sentence in () and show it. Document: """{doc_content}"""*”. Then, we employ iterative refinement in case the LLM misses any other sentences: “*Is there any other sentence in the document directly mentioning the event "{event}"? Please enclose that sentence in () and show it.*” Lastly, we collect the sentences from the responses.

We run the methods on the human-annotated dataset (100 documents). We compute the saliency features of the events in each document and take the average across the events. Lastly, all the values are averaged across all the documents.

C Prompting Details

The total time cost for prompting Llama3 to construct the training data (10247 documents) is about 2200 hours (total wall-clock time of all the machines). We run the models on multiple machines with different specifications, including one with 6× RTX 3090, one with an A100, and one with 2× A40.

Table 9 shows an example of the code prompt for hierarchical graph generation and the response from Llama3. Table 10 shows an example of the hallucination prompt and the response.

Algorithm 1 is the pseudo-code of the entire salient event graph generation process.

In the summarization prompt, we use a temperature of 0.8 and a top_p of 0.9. For the salient event generation prompt, we use a temperature of 0.5 and a top_p of 0.9. The relation graph generation prompt also uses a temperature of 0.5 and a top_p of 0.9. The hallucination grader prompt uses a temperature of 0.

Input to the LLMs	Response from the LLMs

Could you please finish the following code?
import networkx as nx
document = {document}
event_list = {event_list_str}
# This is a graph representing the hierarchical relation between the events in the document 	
# Each edge in the graph represents a subevent relation between the head and tail nodes which are events 	
# An edge means the head event is a subevent of the tail event. They are closely related but on different granularity levels. 	
hierarchical_graph = nx.DiGraph() # This is a directed acyclic graph. There should not be any cycle in the graph.
# Add events as nodes 	
for event in event_list:
    hierarchical_graph.add_node(event)
# Add hierarchical relations as edges to the graph using .add_edge() function. The function takes two strings as inputs. Each string represents an event. 	
# There should not be any cycle in the graph. Explain the reason for each added edge as a comment after each function call 	
	Here is the completed code:
	import networkx as nx
	hierarchical_graph = nx.DiGraph()
	# Add events as nodes
	for event in event_list:
	    hierarchical_graph.add_node(event)
	# Add hierarchical relations as edges to the graph
	hierarchical_graph.add_edge("the government responded by reducing the provincial civil service by 28%", "liberals cut spending")
	# The event "the government responded by reducing the provincial civil service by 28%" is a subevent of "liberals cut spending" because it is one way the Liberals cut spending.

Table 9: An example of the relation graph generation prompt input and the response from Llama3.

Input to the LLMs	Response from the LLMs
You are a grader assessing whether an answer is grounded in / supported by a set of facts. Give a binary score 'yes' or 'no' score to indicate whether the answer is grounded in or supported by a set of facts. Then, provide a short explanation. Here are the facts: {document} Here is the answer: Event "the government responded by reducing the provincial civil service by 28%" is a subevent of event "liberals cut spending".	Score: Yes Explanation: The answer is grounded in the facts because it accurately identifies a specific action taken by the government (reducing the provincial civil service by 28%) as a subevent of the broader event of cutting spending, which is mentioned in the text.

Table 10: An example of the hallucination prompt input and the response from Llama3.

Algorithm 1 CALLMSAE: CAscading Large Language Models for SAlient Event graph generation

Input: Document d, Max Refinement Round k

Output: An Event Relation Graph g

$summary \leftarrow$ Summary_Generation(d)
$salient\_events \leftarrow$ Event_Generation$(summary)$
$hierarchical\_graph \leftarrow$ null
$current\_round \leftarrow 0$
while $current\_round < n$ **do**
    $hierarchical\_graph \leftarrow$ Hierarchical_Graph_Generation$(d, salient_events, hierarchical_graph)$
    $hierarchical\_edges \leftarrow$ Get_Edges$(hierarchical_graph)$
    for $edge_i$ in $hierarchical\_edges$ **do**
        $remove\_edge \leftarrow$ Hallucination_Grader$(d, edge_i)$
        if $remove\_edge$ **then**
            $hierarchical\_graph \leftarrow$ Remove_edge$(hierarchical_graph, edge_i)$
        end if
    end for
    $current\_round \leftarrow current\_round + 1$
end while
$temporal\_graph \leftarrow$ null
$current\_round \leftarrow 0$
while $current\_round < n$ **do**
    $temporal\_graph \leftarrow$ Temporal_Graph_Generation$(d, salient_events, temporal_graph, hierarchical_graph)$
    $temporal\_edges \leftarrow$ Get_Edges$(temporal_graph)$
    for $edge_i$ in $temporal\_edges$ **do**
        $remove\_edge \leftarrow$ Hallucination_Grader$(d, edge_i)$
        if $remove\_edge$ **then**
            $temporal\_graph \leftarrow$ Remove_edge$(temporal_graph, edge_i)$
        end if
    end for
    $current\_round \leftarrow current\_round + 1$
end while
$causal\_graph \leftarrow$ null
$current\_round \leftarrow 0$
while $current\_round < n$ **do**
    $causal\_graph \leftarrow$ Causal_Graph_Generation$(d, salient_events, causal_graph, temporal_graph, hierarchical_graph)$
    $causal\_edges \leftarrow$ Get_Edges$(causal_graph)$
    for $edge_i$ in $causal\_edges$ **do**
        $remove\_edge \leftarrow$ Hallucination_Grader$(d, edge_i)$
        if $remove\_edge$ **then**
            $causal\_graph \leftarrow$ Remove_edge$(causal_graph, edge_i)$
        end if
    end for
    $current\_round \leftarrow current\_round + 1$
end while
$g \leftarrow \{hierarchical\_graph, temporal\_graph, causal\_graph\}$

Guideline

Back to annotation page

When you visit the annotation platform for the first time, there may be a ngrok confirmation page. Just click 'visit' to confirm. ngrok is a tool which we use for setting up the website.

Step one

Select a page from the dropdown menu to start annotating. You can also use the 'Previous' and 'Next' buttons at the bottom to navigate through the pages.

Step one

Each page shows a news article. The large font text is the title. The bold font text is the abstract. The rest is the main content.

You should primarily use the words and phrases from the main content to construct events. Please don't repeat events from the title or the abstract.

Step two

The sidebar on the left show a list of candidate salient event suggested by an algorithm. We ask you to do the followings:

Step two

1. If you think an event is salient, tick the checkbox next to it. Otherwise, untick the checkbox.

2. If you think there is an event that isn't listed, you can add it by entering the event in the text box.**The event should at least contain a subject and a trigger**.

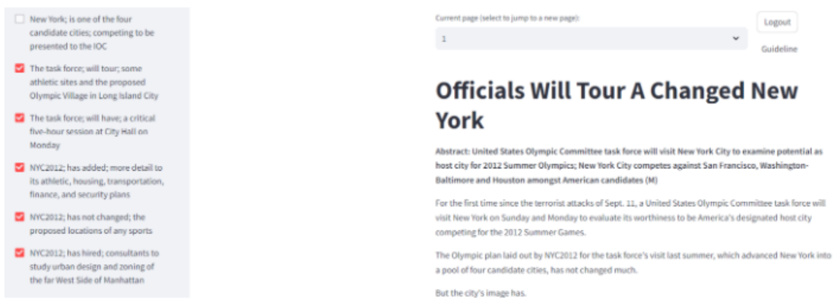
3. If you think a ticked event makes no sense, untick it. When two options are referring to the same event, untick the one you think is less informative.

Every modification will be saved automatically.

Definition of Event

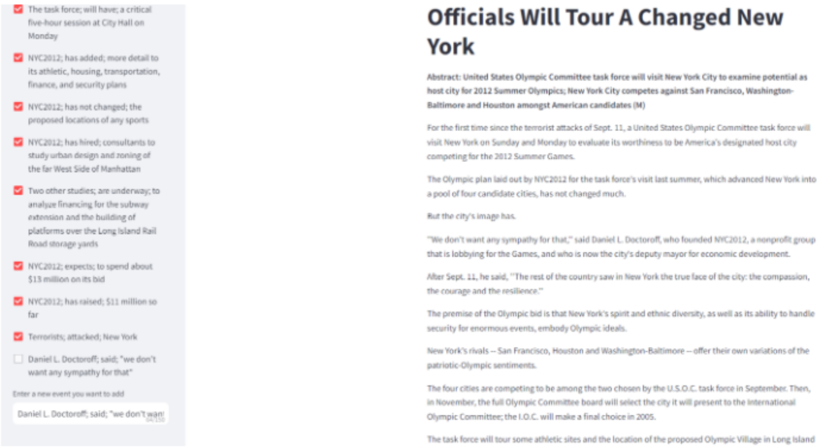
An event is anything that happens as described in the article. We represent the events in a structured format: **actor; trigger; target**. The actor of the event is usually the subject of a sentence. The trigger can be seen as the predicate of a sentence. The target is usually the object in the sentence which is optional.

Example



Example one

In example 1, *New York; is one of the four candidate cities; competing to be presented to the IOC* should not be chosen because the predicate isn't something that can be considered as an event. An event is essentially a change of state. Predicates like "is" is only describing one state. On the other hand, *New York; is competing; with three cities to be presented to the IOC* should be chosen because the predicate can indicate an event.



Example two

In example 2, *Daniel L. Doctoroff; said; we don't want any sympathy for that* is an event but not a salient event because simply describing someone said something isn't important enough in this article.

Example three

In example 3, *MetroStars; won; Major League* isn't selected because it is less accurate and informative than the second option. They are referring to the same event and we don't want duplication. The fourth and fifth options are not selected due to they are more about a description of state than a change of state. *Game; was marked; by physical play* isn't selected because there is no direct reference in the article that the game is marked by physical play.

Below are more annotated examples.

Figure 6: Annotation guidelines of salient event identification shown to the annotators.

Guideline

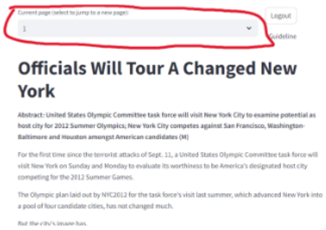
Back to annotation page

Step one

Select a page from the dropdown menu to start annotating. You can also use the 'Previous' and 'Next' buttons at the bottom to navigate through the pages.

Each page shows a news article. The large font text is the title. The bold font text is the abstract. The rest is the main content.

You should primarily use the words and phrases from the main content to construct events. Please don't repeat events from the title or the abstract.



Step one

Step two

You could add event relations to the left sidebar. To add a relation, first choose a source event, then choose a relation type and a target event. There are three options for relation type. Relation **is_caused_by** means the source event is strictly dependent on the target event. The source event wouldn't happen if the target event doesn't happen. Relation **happened_before** means the source event happened before the target event as described by the article. It's the temporal relation in the real world instead of the occurence order in the article. Relation **is_subevent_of** means the source event is the subevent of the target event.

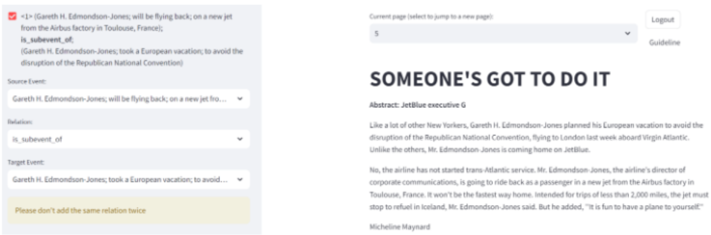
Step two

Please find all the relations that are desbribed in the article. Including the relations that are explicitly mentioned, and the relations that can be inferred based on the evidence presented in the article. If logical inference cannot fully support the relation, please don't include it. Every modification will be automatically saved on the server.

Example

Example one

In example 1, (The task force; will tour; some athletic sites and the proposed Olympic Village in Long Island City) will be on Sunday and (The task force; will have; a critical five-hour session at City Hall) will be on Monday. They are explicitly related by time.



Example two

In example 2, (Gareth H. Edmondson-Jones; will be flying back; on a new jet from the Airbus factory in Toulouse, France) is part of the vacation in the event (Gareth H. Edmondson-Jones; took a European vacation; to avoid the disruption of the Republican National Convention).

Figure 7: Annotation guidelines of relation identification shown to the annotators.