ForestCast: Open-Ended Event Forecasting with Semantic News Forest

Zi Yu
Beijing Institute of Technology
Beijing, China
yuzi@bit.edu.cn

Shaoxiang Wang
Beijing Institute of Technology
Beijing, China
shaoxiangwang@bit.edu.cn

Guozheng Li
Beijing Institute of Technology
Beijing, China
guozheng.li@bit.edu.cn

Yu Zhang
University of Oxford
Oxford, United Kingdom
yuzhang94@outlook.com

Chi Harold Liu
Beijing Institute of Technology
Beijing, China
chiliu@bit.edu.cn

Abstract

Open-ended event forecasting (OEEF) seeks to predict future events from a given context without being restricted to a predefined scope or format. It plays a crucial role in domains such as risk management and financial decision making. Although large language models show potential for OEEF, existing approaches and datasets often overlook the complex relationships among events, and current research lacks comprehensive evaluation methods. To address these limitations, we propose ForestCast, a prediction pipeline that extracts forecast-relevant events from news data, organizes them into a story tree, and predicts subsequent events along each path. The pipeline comprises four steps: (1) clustering news into event nodes, (2) constructing a news story tree, (3) mining the semantic structure of the tree, and (4) predicting the next event node and evaluating prediction quality. To support this pipeline, we construct NewsForest, a dataset of 12,406 event chains, each representing a chronologically and logically linked sequence of news events. In addition, we introduce a comprehensive evaluation framework that measures both the accuracy and the quality of prediction. Experimental results demonstrate that ForestCast improves the ability of LLMs to forecast events in news data.

1 Introduction

Event prediction, the task of forecasting future events based on past and current information, has great potential in applications including policy making, risk management, and financial decision-making (Zhao, 2022). Accurate predictions can help decision makers anticipate challenges and seize opportunities (Zhao, 2022).

Conventional forecasting approaches include script event prediction (Chambers and Jurafsky, 2008) and temporal knowledge graph completion (TKGC) (Leblay and Chekol, 2018). These methods are restricted to predicting specific attributes and selecting answers from a finite range (Lin et al., 2022; Ma et al., 2024; Shi et al., 2023; Xu et al., 2023). However, real-world developments are rarely limited to such specific scopes, and critical information is not always captured by predefined attributes.

The text generation capabilities and open prediction space of large language models (LLMs) make them promising for open-ended event forecasting (OEEF). Thus, previous work has used LLMs for such forecasting tasks (Wang et al., 2025). However, OEEF presents two challenges. First, existing prediction methods and datasets typically handle massive forecast-related data by simply grouping and summarizing (Guan et al., 2024; Ma et al., 2024; Wang et al., 2025), overlooking the complex relationships between events in forecast-related data. Second, due to the open-ended characteristic of OEEF, existing evaluation methods (Ye et al., 2024; Wang et al., 2025) cannot reasonably evaluate prediction results.

To address these challenges, we introduce ForestCast, a method for the OEEF task that leverages LLMs and captures evolutionary structures within event data to forecast multiple potential evolutionary trajectories. ForestCast is built on the NewsForest dataset, which highlights key drivers of event progression and uncovers the logical relationships between them. To evaluate the performance of ForestCast, we propose an evaluation framework that measures both the accuracy and the quality of the predictions. Experimental results demonstrate improvements in both accuracy and quality after fine-tuning with the NewsForest dataset, including

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 12667–12681

a maximum pass@1 improvement of 6.00% and a 0.18 increase (on a five-point scale) in average quality metrics. The dataset and the code are available at https://github.com/bitvis2021/ForestCast.

The contributions of this work are as follows:

- We develop **ForestCast**, a method for open-ended event forecasting. This method organizes large-scale news into a news story tree and uses a fine-tuned model to predict the future evolution of the story tree.
- We construct **NewsForest**, a dataset for open-ended event forecasting that can be used to fine-tune LLMs. The dataset contains 12,406 prediction chains across four major domains: economics, politics, military, and social events.
- We introduce an evaluation framework for event prediction results. It evaluates both the guidance value of predictions and whether the predicted events occurred.

2 Related Work

This section reviews the literature on news story tree construction and event prediction.

2.1 News Story Tree Construction

The topic detection and tracking task (Allan et al., 1998) extracts key information from large-scale news corpora by thematically grouping and continuously tracking news events. However, this task ignores potential dependency relationships between events. To address this issue, researchers have proposed methods to capture the structural features of event evolution. Nallapati et al. (2004) quantify the dependency between two events based on temporal relationships and the TF-IDF vector distance. Yang et al. (2009) introduce the concept of event graphs to describe the relationship between events.

However, these studies only focus on pairwise event relationships and cannot fully represent the overall evolutionary structures of events. Shahaf et al. (2012) propose "metro maps" to describe evolutionary structures of events. Liu et al. (2018, 2020) introduce the news story tree, a structure more aligned with patterns of event development and user cognition, where dependencies between events are constructed through keyword-based maps. However, these methods remain limited to pruning operations on the graph structure and fail to effectively capture the internal cohesion within the same branch and the distinctiveness between different branches.

Moreover, existing methods for capturing evolutionary events rely mainly on low-level text feature analysis, such as keyword graphs (Liu et al., 2020), keyword reoccurrence rate (Shahaf et al., 2012), and TF-IDF vectors (Nallapati et al., 2004). In contrast, the method proposed in this paper leverages LLMs and pre-trained sentence encoders, enabling higher-level semantic analysis.

2.2 Event Prediction

Script event prediction (Chambers and Jurafsky, 2008) requires selecting the most likely subsequent event from a candidate list given an event context. Several studies have predicted event outcomes by building event chains (Wang et al., 2017, 2024; Radinsky and Horvitz, 2013), event evolution graphs (Ding et al., 2019; Li et al., 2018; Du et al., 2022), or deformation structures of event graphs (Zhou et al., 2021; Ma et al., 2023; Granroth-Wilding and Clark, 2016) as the evolutionary structures of the events. On the other hand, TKGC (Xia et al., 2024; Deng et al., 2020; Rong et al., 2025; Zhang et al., 2024a,b) addresses incomplete temporal knowledge graphs by learning representations of entities, relations, and timestamps to predict missing information.

Both script event prediction and TKGC are limited to predicting specific attributes and can only generate results within predefined scopes. When predicting real-world events, these methods offer insufficient guidance. To overcome these limitations, Guan et al. (2024) first proposes the OEEF task, which is characterized by

- diverse predictive questions covering different stages of event development and viewpoints, promoting comprehensive analysis;
- flexible prediction outputs with no restrictions on scope, format, or length, thus allowing semantically rich responses.

However, existing OEEF approaches also face challenges. Guan et al. (2024) propose a prediction method with a manually constructed test dataset, but it only clusters and summarizes forecast-related news, ignoring dependencies between events. Wang et al. (2025) propose a large OEEF dataset, but found that model performance decreases after fine-tuning on it. We hypothesize that this is because the dataset organizes topics by listing decades of history for a place or person, making it difficult to capture the hidden factors and underlying logic driving event development.

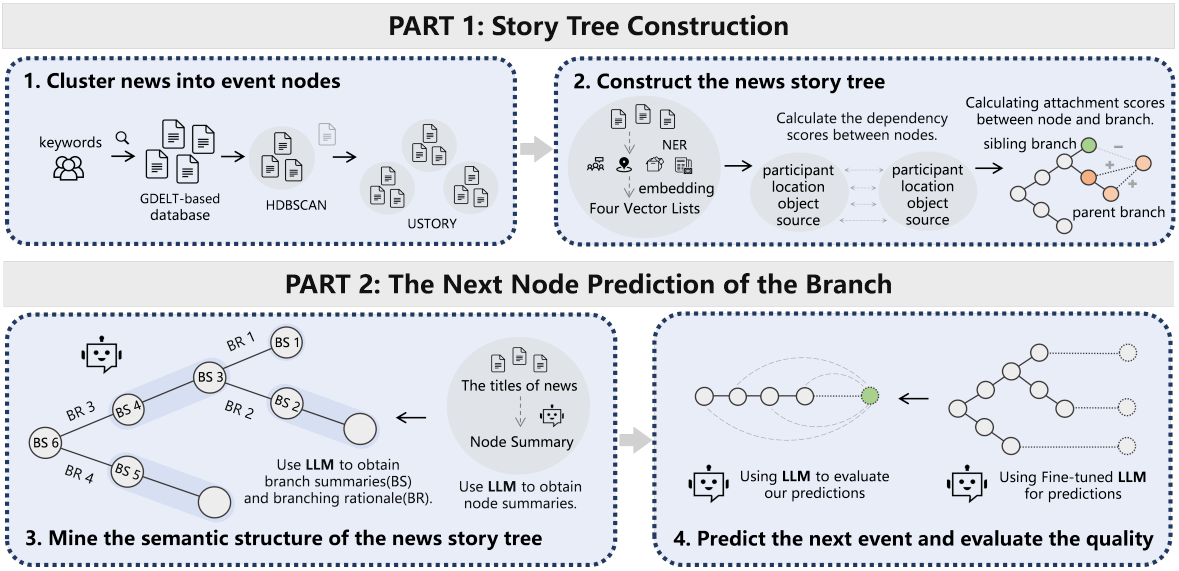

Figure 1: **ForestCast pipeline:** The first part of the pipeline constructs the story tree by grouping news into event nodes (step 1) and building a story tree (step 2). The second part predicts the next node along a branch by mining the semantic structure of the story tree (step 3) and predicting events while evaluating prediction quality (step 4).

Overall, existing work on forecast-related data processing remains focused on extracting event attributes and linking events through such attributes, while overlooking the developmental logic of information. In contrast, ForestCast mines this developmental logic and makes predictions based on it. We construct an OEEF dataset grounded in the developmental logic of events and employ a more comprehensive evaluation framework to examine the effectiveness of our method.

3 The ForestCast Method

We propose the ForestCast method, which forecasts events based on the organization of evolutionary structures. Our method requires users to provide topic-related keywords along with the start and end dates of the news collection period. For example, if a user wants to generate forests for the follow-up events of Trump's election campaign in February 2025, the user would specify the keyword (Trump, Election) and duration (20250201 to 20250228). Appendix A.1.4 describes the time complexity of the method and its time cost on a specific hardware setup.

We develop an interactive system (see Appendix Figure 5) for the user to provide such specification. The system automatically collects and analyzes relevant news, constructs the story tree, and produces predictions. We detail additional functions of the interactive system in Appendix A.1.1. As Figure 1 shows, the ForestCast method consists of four steps:

- **Step 1:** Cluster news into event nodes.
- **Step 2:** Build the news story tree.
- **Step 3:** Mine the semantic structure of the story tree.
- **Step 4:** Predict the next node and evaluate the prediction quality.

3.1 Cluster News into Event Nodes

Our data is sourced from the GDELT (Leetaru and Schrodt, 2013) database[1] to ensure comprehensive coverage and credibility. From GDELT, we obtain news headlines, links, publication times, and media sources. We then retrieve the full text of the news based on the links. We use keyword searches for news articles, deduplicate them, and cluster the articles describing the same event. The clusters serve as event nodes in the news story tree. For speed and accuracy considerations, we use semantic-based USTORY (Yoon et al., 2023) for event node clustering.

3.2 Construct the News Story Tree

In prior work on the topic detection and tracking task, the story trees have proven effective as visual representations that align with users' cognitive habits and the evolutionary structure of events. For

[1]The GDELT (Global Database of Events, Language and Tone) database is a real-time open database that monitors media coverage worldwide. URL: https://www.gdeltproject.org/

visual representations informative to human, its corresponding data structure may also be informative to LLM. Following this rationale, we build story trees to organize relevant news.

3.2.1 Calculate Dependency Scores

Inspired by Yang et al. (2009), we assume that the hidden dependency between events is determined by their participants, locations, objects, and media sources. Therefore, we use the text processing model en_core_web_sm of spaCy library (Honnibal and Montani, 2017) to extract features from news articles, deduplicate them, and disambiguate them, resulting in four sets of terms: $set_{\text{participant}}$, set_{location}, set_{object}, set_{source}. Since the importance of these features varies, we compute their frequencies and assign weights accordingly. After reordering features by weight, we obtain four weighted lists: $list_{\text{participant}}$, $list_{\text{location}}$, $list_{\text{object}}$, $list_{\text{source}}$. We further assume that each feature type contributes differently to the dependency relationship, and therefore set four weights: $\alpha_{\text{participant}}$, α_{location}, α_{object}, α_{source}.

To ensure that the construction is based on semantic relation, we use the pre-trained word encoder GloVe (Pennington et al., 2014) to encode the terms, producing weighted vector lists: $vec_{\text{participant}}$, vec_{location}, vec_{object}, vec_{source}. The dependency score between two nodes (denoted i and j) is then computed as the weighted sum of their cosine similarities:

$$\text{DepScore}_{i,j} = \sum_{k \in \{\text{part,loc,obj,src}\}} \alpha_k \cdot \text{sim}(v_{k,i}, v_{k,j})$$

Details of the hyperparameter settings (α, together with μ and λ introduced in the next section), are provided in Appendix A.1.3.

3.2.2 Calculate Attachment Scores

To ensure that event evolution forms a cohesive structure, our method enhances the distinction between different branches of the tree while reinforcing cohesion within the same branches.

When attaching a candidate node v to a potential attachment node u in the tree, we consider the dependency score $dep(v, u)$ between them, as well as the dependency scores between v and the nodes in u's parent branch (P_u) and sibling branches (S_u). The dependency score of the potential attachment node accounts for a proportion μ. The parent branch are defined as the trace back to the first ancestor node with multiple children of the potential attachment node. The sibling branch are defined as the other branches (not including the parent branch) of the first ancestor node with multiple children of the potential attachment node.

Within the parent branch, the weights (w_{p}) of the nodes decay as the path length to the potential attachment node increases. Within the sibling branches, all nodes are assigned the same weight. In addition, we introduce a penalty coefficient λ for sibling branches.

$$\begin{aligned}\text{AttachScore}_{v,u} = {} & \mu \text{dep}(v, u) \\ & + (1 - \mu) \sum_{p \in P_u} w_p \text{dep}(v, p) \\ & - (1 - \mu)\lambda \frac{1}{|S_u|} \sum_{s \in S_u} \text{dep}(v, s)\end{aligned}$$

where:

- P_u: The set of ancestor nodes from u up to, but not including, the first ancestor node with multiple children. $w_p = \exp(-\beta \cdot d(p, u))$, where $d(p, u)$ denotes the distance and $\beta > 0$.
- S_u: The set of u's siblings that share the same parent as u. $|S_u|$ is the number of siblings.

3.2.3 Connect Story Tree Nodes

With the attachment scores defined, we construct the story tree from the event nodes. We define the time of an event node as the average publication time of its associated news articles. The nodes are rearranged chronologically to form a sequence.

To prevent nodes that are temporally earlier but significantly offset from the entire story tree from disrupting its construction, we introduce a phase to remove such nodes. To remove these nodes, we select the first five nodes and connect the five nodes to a small tree. Then, we discard the nodes that have attachment scores lower than an adaptive threshold. The remaining of the first five nodes are added the node to the formal build sequence.

Then, we sequentially take each node from the formal build sequence as a candidate and evaluate its attachment scores to all nodes already in the tree. A candidate is attached to the node with the highest score. If the highest score falls below a threshold, the node is discarded.

3.3 Mine the Semantic Structure of the News Story Tree

Existing methods (Liu et al., 2020; Shahaf et al., 2012; Nallapati et al., 2004) typically attach nodes according to low-level textual features while neglecting semantics. We instead use LLMs to mine the semantic structure of the story tree after construction, which better captures the evolutionary structure of hidden events. In particular, we execute the following steps:

1. Use an LLM to generate event node summaries, branching rationale and branch summaries (Section 3.3.1).
2. Adjust the branches based on their summary distances (Section 3.3.2).

All prompts are provided in Appendix A.1.2.

3.3.1 Obtain Summary and Branching Rationale Using LLMs

LLMs are limited in handling long texts and tree structures. Therefore, we use a bottom-up data processing method to obtain the branching rationale using an LLM. First, we pass all news headlines in an event node to the LLM to generate a node summary. After obtaining the summary for each node, we perform a post-order traversal of the entire tree. Starting from the leaf nodes, we generate branch summaries and branching rationales (when applicable) for each node. Specifically, when the traversal reaches a node, one of the following processing is applied:

- **Option 1 - for a node v with zero or one child:** If v has no siblings or no children, no branch summary or branching rationale is stored at v. Otherwise, if node v has siblings and exactly one child, its branch summary is computed as follows. We first identify the chain of nodes from v to its closest descendant node w that has multiple children (both v and w are included in the chain). We then pass the branch summary (or the node summary, when the node has no branch summary) of each node in the chain to the LLM. The LLM is instructed to output branch summary of the chain, which is stored to node v.
- **Option 2 - for a node v with multiple children:** We pass the branch summary (or the node summary, when the node has no branch summary) of each child node along with v's node summary to the LLM. The LLM is instructed to output both the branching rationale and a branch summary. The branching rationale and branch summary are stored at v.

This process aggregates information layer by layer, ultimately yielding the branching rationale, branch summaries for each branch. At the root node, we obtain a summary of the entire tree.

3.3.2 Adjust Tree With Summary Distance

We encode node summaries using the pre-trained sentence encoder all-MiniLM-L6-v2 (Wang et al., 2021). We compute Euclidean distances between each node's node summary vector and its parent's. If the distance is above a threshold, we then examine the distance between the nodes' node summary vector and that of the root node. If the distance is below the threshold, this node and its corresponding branch is attached to the root node.

3.4 Predict Next Event and Evaluate Prediction Quality

After obtaining the story tree's semantic structure, we use an LLM to predict the next event based on the semantic structure. Each path from a leaf to the root is treated as a news development chain. The root node's branch summary is used as background context. We pass a news development chain and the background to the LLM, requiring it to predict the next node.

The predictions are attached to the end of the news story tree. Each prediction's quality is evaluated on five quality metrics in Table 1. The metrics evaluate different scopes:

- **Specificity** evaluates the node itself.
- **Validity** and **relevance** assess the relationship between the current node and its preceding nodes.
- **Chain consistency** and **causal insight** evaluate the entire chain.
- **Branch rationality** assesses the whole tree.

We instruct LLM to rate each of the six attributes on a 1 - 5 score by giving scoring examples.

Metric	Description
Specificity	Is the event description specific?
Validity	Does the event provide new information?
Relevance	Are the main participants consistent consecutively?
Chain Consistency	Is logic consistent throughout the chain?
Causal Insight	Are hidden relationships captured?
Branch Rationality	Are branches distinct/non-mergeable?

Table 1: Quality metrics for news story trees.

For prediction, we use Qwen2.5-7B fine-tuned on the NewsForest dataset (see Section 4). Section 6 shows that fine-tuning effectively improves the prediction accuracy and quality.

4 The NewsForest Dataset

The NewsForest dataset contains 12,406 news story chains. By masking the event chains, the dataset can be used to fine-tune LLMs to improve their predictive capabilities. An example of masked event chain is shown in Table 2.

4.1 Select and Clean News Data of Story Trees

We select four important domains to include in our dataset: economics, politics, military, and social events. The first three domains cover both long-term and short-term developments, whereas social events tend to be short-term. Table 3 shows example topics in the four domains.

Among the four domains, we manually select popular topics. We select keyword groups for popular topics from December 2024 to April 2025. Appendix A.2.1 describes the selection process. Our data is sourced from GDELT. Given keywords and time periods, we query all articles with matching keywords in titles.

To ensure topic coherence, we encode article headlines using all-MiniLM-L6-v2 (Wang et al., 2021). Then we use HDBSCAN (Rahman et al., 2016) to cluster the articles based on embedding of headlines and retain only the largest cluster as the final data source. For each topic, to cover both long-term and short-term prediction tasks (Wang et al., 2025), we set news article search windows of 20, 30, 40, 60, and 80 days. Within each time window, the start and end of the story tree are randomly determined. If insufficient articles are found, the window is shifted forward or backward by one-quarter of the span until the requirement is met.

4.2 Build Event Chains

After determining keywords and time periods, we construct event chains with three steps: cluster news into event nodes, construct the news story tree, and mine the semantic structure of the news story tree. This process is described in Section 3.1, 3.2, and 3.3.

4.3 Mask Event Chains

After constructing the news story tree, we treat the path from a leaf node to the root as a news development chain. The branch summary of the root node serves as contextual background. To enable LLMs to learn the evolutionary structure of hidden events, we mask nodes in each chain:

- For chains longer than six nodes, we mask the last one third of nodes.
- For chains of six nodes or fewer, we mask only the final node.

Masked nodes serve as the correct answers for the chain prediction. Long chains can generate two or more prediction chains. We connect the unmasked chain and one of the masked nodes as a predicted chain. Therefore, if n nodes in a story chain are masked, the story chain will generate n prediction chains. We use the DPO method (Rafailov et al., 2023) to fine-tune the model.

Prior work shows that the effectiveness of DPO depends primarily on the chosen answer (Pan et al., 2025). Thus, we only test two rejection schemes:

- **DPO-irrelevant:** Select the node with the lowest attachment score (i.e., the most irrelevant node) within the story tree as the reject entry.
- **DPO-fake:** Invoke an LLM to generate a fictitious node as the reject entry.

Both rejection schemes are tested in the Section 6. An example data point of the DPO-irrelevant scheme is shown in Table 2.

The final dataset for each rejection scheme contains 12,406 prediction chains, covering global news over six months. Table 4 shows the chain length distribution. Table 5 shows data volume across domains.

4.4 Evaluate the Dataset Quality

Following the evaluation method in Guan et al. (2024), we propose six quality metrics to evaluate the evolutionary structures of events (see Table 1). The evaluation results (Figure 2) show average scores from 2.57 to 3.74. Chain consistency and branch rationality scores 3.66 and 3.70, indicating good structural integrity.

5 Evaluation framework

Evaluating OEEF results is tricky because general predictions that do not specify event details are more likely to be true by chance. To address this issue, we evaluate predictions based on both accuracy and quality metrics.

Accuracy: To measure accuracy, we determine whether an event has occurred with online

Header	Content
Instruction	Task: Please predict the next node in the chain based on the background.
Input	**Background**: In New York, a federal judge kept a lawsuit against Trump alive, while Trump's legal team pushed for dismissals. Judges delayed or recused from Trump-related cases. Democrats confirmed judges pre-inauguration, and Republicans tried to stall Trump trials. **Chain**: 1. A federal judge denied throwing out a baseless lawsuit filed against Trump, maintaining its validity in court proceedings. 2. In Colorado, Republican judges dismissed Trump's election interference and secret documents cases, potentially setting Aileen Cannon as a future AG; these rulings could sway the outcome of Trump supporter candidate selection in court appointments. 3. Judges continue to cancel Trump's 2020 election case court deadlines post-presidential win, with a judge shutting down GOP challenge months after Trump's victory. 4. The judge paused Trump-related January 6 cases, citing special counsel's request and concerns about timing after Trump's election win and before his inauguration.
Chosen	After Joe Biden's victory in the US presidential election, a judge paused Donald Trump's legal challenges to the 2020 election results. This decision came shortly after the formal certification of the election outcome. Judge Jack Smith put the federal election interference case against Trump on hold.
Reject	Rioters sought charges dismissal unless Trump was also convicted, but lost in court, and are sentenced to prison following the Capitol riot.

Table 2: **Example of masked DPO-irrelevant data:** The background in the *input* summarizes of the entire story tree. The chain in the *input* is a masked story path, which is from a root node to a leaf node. The *chosen* entry is the masked node. The *reject* entry can be the least relevant node in the story tree or a fake node generated by an LLM.

Domain	Example Topics
Economics	(tariff, China), (stock, market)
Politics	(elect, republican), (trump, China)
Military	(NATO, Ukraine), (America, Iran)
Social Events	(Florida, hurricane), (police, racism)

Table 3: **Example topics from different domains in the NewsForest dataset:** Each topic consists of two or three keywords. Keyword groups are used in the stage of constructing in the dataset to search for news containing those keywords.

Attribute	Story Chain	Prediction Chain
Average Length	4.39	2.95
Max Length	14	9
Min Length	2	1

Table 4: Chain lengths before and after masking. Prediction chain is the result of masking story chain.

Domain	# Trees	# Story Chain	# Prediction chain
Economics	157	1726	2396
Politics	284	3613	5326
Military	217	2801	3942
Social Events	36	461	742
Total	**694**	**8601**	**12406**

Table 5: **Data volume per domain:** Number of trees, story chains, and prediction chains in each domain.

searches (Ye et al., 2024). The prediction results are first extracted from the forecasting model. We then search on-line for latest related news and provide this information on to an LLM for evaluation. The LLM decides whether the prediction has actually occurred. If so, the prediction is scored on the following quality metrics.

Quality metrics: To evaluate the informational value of predictions, we employ LLMs to assess prediction quality. Following Guan et al. (2024), we design five metrics (see the metrics in Table 1, with branch rationality excluded) and corresponding scoring criteria and examples to prompt LLMs to evaluate the quality of predictions. These five metrics are also used in the evaluation of dataset quality (see Section 4.4). Note that we use only five of the six metrics. We exclude branch rationality because it is specifically designed for tree structures and reflects the quality of a story tree rather than the quality of predictions.

6 Experiments

We evaluate the effectiveness of the ForestCast method through experiments conducted across multiple LLMs. Our goal is to examine whether the ForestCast method effectively captures hidden relationships that LLMs can learn.

We created a test dataset for the experiments, which includes both recent hot topics from April 13 to 20, 2025, and long-term popular topics from the past six months, such as the US tariff policy. For these topics, we constructed news story trees with time spans of 20, 30, 40, 60, and 80 days, similar

Score	Scoring Criteria	Example
1 (vague)	Mentions general idea only; lacks specifics (actions, people, results). Uses general words (e.g., "attention").	Chang'e 6 attracted widespread international attention.
2 (somewhat specific)	Little specific info, mostly vague. You may mention the event type/reaction, not the specific participants/actions/results.	After Chang'e 6 completed its important mission, it received some international feedback.
3 (moderately specific)	Some key details (core content), giving a general idea; may lack specifics (participants, results, background).	Chang'e 6 successfully collected and returned samples... attracting international attention.
4 (more specific)	Most key info (event, people/institutions, specific results/reactions). Core elements are relatively clear.	Chang'e 6 successfully brought back samples... space agencies from many countries expressed congratulations.
5 (very specific)	Clearly describes main events, identifies key participants, specific actions, results/reactions. Provides verifiable details.	Chang'e 6 successfully brought back samples... NASA Administrator Bill Nelson sent congratulations to China, expressing pleasure at the mission's achievement.

Table 6: Scoring criteria and example for the specificity metric introduced in Table 1. The scoring criteria are included in the prompt given to the LLM.

	Prediction Quality Metrics						Prediction Accuracy		
Model	**Spec.**	**Rel.**	**Valid.**	**Causal.**	**Consist.**	**Avg.**	**pass@1**	**pass@3**	**pass@5**
Qwen2.5-7B	2.91	2.55	2.50	2.83	3.14	2.79	16.22%	40.89%	56.22%
Qwen2.5-7B-DPO-irrelevant	**3.02**	**2.63**	2.56	**2.89**	3.21	2.86	**18.27%**	**46.00%**	**62.22%**
Qwen2.5-7B-DPO-fake	2.93	2.62	**2.72**	2.88	**3.27**	**2.88**	17.34%	42.44%	59.56%
LLaMA3.1-8B	2.90	2.53	2.49	2.77	3.10	2.76	18.00%	40.44%	55.33%
LLaMA3.1-8B-DPO-irrelevant	2.93	**2.57**	2.52	2.78	**3.13**	2.79	18.67%	46.89%	61.33%
LLaMA3.1-8B-DPO-fake	**2.97**	2.51	**2.56**	**2.86**	3.09	**2.80**	**24.00%**	**47.78%**	**64.66%**
Gemma2-9B	2.88	2.42	2.35	2.70	3.03	2.68	20.46%	43.56%	58.67%
Gemma2-9B-DPO-irrelevant	3.00	2.58	2.41	2.77	3.10	2.77	22.00%	**49.56%**	**66.00%**
Gemma2-9B-DPO-fake	**3.02**	**2.60**	**2.44**	**2.87**	**3.15**	**2.82**	**22.22%**	49.33%	65.11%

Table 7: **Model performance before and after fine-tuning with NewsForest:** Both the DPO-fake and DPO-irrelevant datasets are processed based on NewsForest. We use the same temperature and top_k for same LLM in the experiment. The quality metrics are described in Table 1 and are averaged for accurate predictions.

to the configuration of the NewsForest dataset, all concluding on April 20, 2025. Specifically, we reserved a 30-day period to ensure that all true predictions could unfold.

To examine whether the NewsForest dataset captures implicit real-world associations learnable by LLMs, we fine-tuned Qwen2.5-7B (Qwen Team, 2024), LLaMA3.1-8B (Meta AI, 2024), and Gemma2-9B (Gemma Team, 2024), using DPO-fake and DPO-irrelevant datasets. Appendix A.4 describes fine-tuning details. Table 7 shows that:

- After fine-tuning with datasets constructed using different rejection schemes, both the accuracy and quality of the predictions are improved.
- As mentioned in the existing studty (Pan et al., 2025), the fine-tuning efficiency of the DPO dataset depends on the answers chosen. The experimental results indicate that there is no significant difference in performance between the two DPO datasets.
- Among all quality metrics, validity shows the greatest improvement after fine-tuning.

We analyze the prediction accuracy and quality of the model at different time lengths before and after fine-tuning. Specific results are shown in Figure 3. We observe that LLMs exhibit a marked upward trend in predictive accuracy as the time span increases. Quality metrics show a slight decline with increasing time length. For near-term events, LLMs tend towards bold yet effective predictions, whereas for long-term events, LLMs lean towards stable yet unremarkable predictions.

7 Conclusion

This paper proposes ForestCast, a pipeline for open-ended event foresting that organizes large-scale news into story trees, mines semantic structures, and predicts future developments. We also intro-

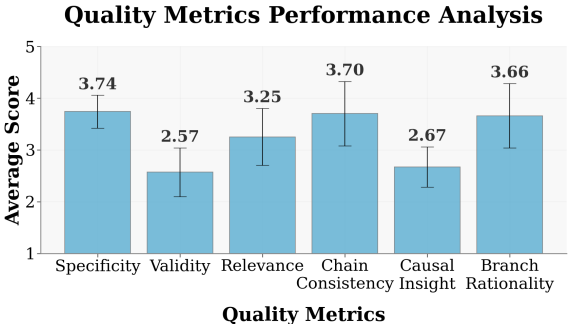

Figure 2: The mean and median scores of the NewsForest dataset across multiple metrics. The assessment is conducted by an LLM, with results categorized into five grades. Grade 5 denotes success, while Grade 1 indicates failure.

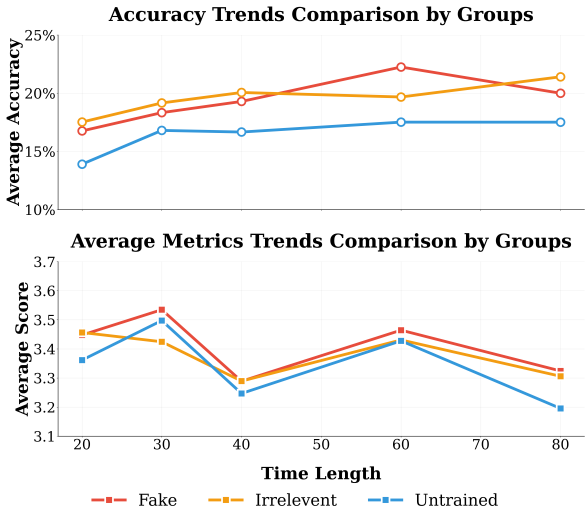

Figure 3: Pass@1 accuracy and average quality metrics comparison across prediction tasks of varying durations before and after model fine-tuning. The irregular group represents the average accuracy of LLaMA, Qwen, and Gemma fine-tuned on the DPO-irregular dataset. The same applies to the fake group.

duce NewsForest, a dataset of 12,406 event chains, and proposed an evaluation framework. Experiments on multiple LLMs show that ForestCast improves prediction accuracy and quality. This work provides a practical foundation for building more effective event forecasting systems. Our results imply that data structures corresponding to visualizations useful to humans are also useful to LLMs.

Limitations

First, our dataset is limited to news articles. Other data sources, such as social media, are also critical sources of information about event developments in contemporary society. Second, the automatic evaluation based on LLMs may not fully align with human evaluation, and future work will aim to bridge this gap. Third, due to concerns about knowledge leakage, our dataset covers only six months of data, which prevents us from evaluating the ability of LLMs to predict events beyond this period.

Ethics Statement

In this study, the ForestCast system was developed using open-source projects and the publicly available GDELT database. These resources have been widely used in prior studies, ensuring that no ethical standards are compromised. The GDELT data is available for unlimited and unrestricted use for academic, commercial, or governmental use of any kind without expense. Our derivative dataset, NewsForest, is used solely within this context and fully complies with GDELT's terms of use.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported by NSFC (Grant No. 62302038) and Young Elite Scientists Sponsorship Program by CAST (Grant No. 2023QNRC001). Guozheng Li is the corresponding author.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45. ACM.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1585–1595. ACM.

Xiao Ding, Zhongyang Li, Ting Liu, and Kuo Liao. 2019. ELG: an event logic graph. *CoRR*, abs/1907.08015.

Li Du, Xiao Ding, Yue Zhang, Ting Liu, and Bing Qin. 2022. A graph enhanced BERT model for event prediction.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2727–2733. AAAI Press.

Yong Guan, Hao Peng, Xiaozhi Wang, Lei Hou, and Juanzi Li. 2024. Openep: Open-ended future event prediction. *CoRR*, abs/2408.06578.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference*, pages 1771–1776. International World Wide Web Conferences Steering Committee.

Kalev Leetaru and Philip A. Schrodt. 2013. GDELT: Global data on events, location, and tone. *ISA Annual Convention*.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4201–4207. AAAI Press.

Li Lin, Yixin Cao, Lifu Huang, Shu'ang Li, Xuming Hu, Lijie Wen, and Jianmin Wang. 2022. What makes the story forward?: Inferring commonsense explanations as prompts for future event generation. In *The International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1098–1109. ACM.

Bang Liu, Fred X. Han, Di Niu, Linglong Kong, Kunfeng Lai, and Yu Xu. 2020. Story forest: Extracting events and telling stories from breaking news. *Proceedings of the ACM on Conference on Information and Knowledge Management*, 14(3):31:1–31:28.

Bang Liu, Di Niu, Kunfeng Lai, Linglong Kong, and Yu Xu. 2018. Growing story forest online from massive breaking news. *CoRR*, abs/1803.00189.

Yunshan Ma, Chenchen Ye, Zijian Wu, Xiang Wang, Yixin Cao, and Tat-Seng Chua. 2023. Context-aware event forecasting via graph disentanglement. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1643–1652. ACM.

Yunshan Ma, Chenchen Ye, Zijian Wu, Xiang Wang, Yixin Cao, Liang Pang, and Tat-Seng Chua. 2024. SCTc-TE: A comprehensive formulation and benchmark for temporal event forecasting.

Meta AI. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. 2004. Event threading within news topics. In *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*, pages 446–453. ACM.

Yu Pan, Zhongze Cai, Guanting Chen, Huaiyang Zhong, and Chonghuan Wang. 2025. What matters in data for dpo?

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the ACM international conference on Web search and data mining*, pages 255–264. ACM.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc.

Md Farhadur Rahman, Weimo Liu, Saad Bin Suhaim, Saravanan Thirumuruganathan, Nan Zhang, and Gautam Das. 2016. HDBSCAN: density based clustering over location based services. *CoRR*, abs/1602.03730.

Huan Rong, Zhongfeng Chen, Zhenyu Lu, Xiao-ke Xu, Kai Huang, and Victor S. Sheng. 2025. Pred-id: Future event prediction based on event type schema mining by graph induction and deduction. *Information Fusion*, 117:102819.

Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. 2012. Trains of thought: generating information maps. In *Proceedings of the World Wide Web Conference*, pages 899–908. ACM.

Xiaoming Shi, Siqiao Xue, Kangrui Wang, Fan Zhou, James Y. Zhang, Jun Zhou, Chenhao Tan, and Hongyuan Mei. 2023. Language models can improve event prediction by few-shot abductive reasoning. In *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics*, pages 2140–2151. Association for Computational Linguistics.

Zhen Wang, Xi Zhou, Yating Yang, Bo Ma, Lei Wang, Rui Dong, and Azmat Anwar. 2025. Openforecast: A large-scale open-ended event forecasting dataset.

In *Proceedings of the International Conference on Computational Linguistics*, pages 5273–5294. Association for Computational Linguistics.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 57–67. Association for Computational Linguistics.

Zikang Wang, Linjing Li, and Daniel Zeng. 2024. Integrating relational knowledge with text sequences for script event prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):9443–9454.

Yuwei Xia, Ding Wang, Qiang Liu, Liang Wang, Shu Wu, and Xiaoyu Zhang. 2024. Chain-of-history reasoning for temporal knowledge graph forecasting. In *Findings of the Association for Computational Linguistics*, pages 16144–16159. Association for Computational Linguistics.

Wenjie Xu, Ben Liu, Miao Peng, Xu Jia, and Min Peng. 2023. Pre-trained language model with prompts for temporal knowledge graph completion. In *Findings of the Association for Computational Linguistics*, pages 7790–7803. Association for Computational Linguistics.

Christopher C. Yang, Xiaodong Shi, and Chih-Ping Wei. 2009. Discovering event evolution graphs from news corpora. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(4):850–863.

Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. 2024. MIRAI: evaluating LLM agents for event forecasting. *CoRR*, abs/2407.01231.

Susik Yoon, Dongha Lee, Yunyi Zhang, and Jiawei Han. 2023. Unsupervised story discovery from continuous news streams via scalable thematic embedding. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 802–811. ACM.

Jinchuan Zhang, Bei Hui, Chong Mu, Ming Sun, and Ling Tian. 2024a. Historically relevant event structuring for temporal knowledge graph reasoning. *CoRR*, abs/2405.10621.

Jinchuan Zhang, Ming Sun, Qian Huang, and Ling Tian. 2024b. PLEASING: exploring the historical and potential events for temporal knowledge graph reasoning. *Neural networks*, 179:106516.

Liang Zhao. 2022. Event prediction in the big data era: A systematic survey. *ACM Computing Surveys*, 54(5):94:1–94:37.

Yucheng Zhou, Xiubo Geng, Tao Shen, Jian Pei, Wenqiang Zhang, and Daxin Jiang. 2021. Modeling event-pair relations in external knowledge graphs for script reasoning. In *Findings of the Association for Computational Linguistics*, pages 4586–4596. Association for Computational Linguistics.

A Appendix

A.1 ForestCast Method

Figure 5 shows the main interface. This section describes the additional functions, implementation details, and time complexity analysis.

A.1.1 Function Demonstration

Keyword Extraction. To help users quickly grasp the development structure of the story tree, we extract keywords for each node. Keyword extraction prioritizes terms with high recurrence rates within the same branch and low recurrence rates across branches. Keywords are derived from a deduplicated list of key entities associated with each node. An example is shown in Figure 4.

Tree Folding and Unfolding. When the number of news articles is large, the story tree may contain too many nodes to display clearly. To address this issue, we implement folding and unfolding functions, allowing users to focus on specific branches while maintaining an overview of the entire tree (Figure 6).

News Data Analysis. To give users both a global and detailed views of topic-related news, we analyze the raw news data. In the sidebar, we display the publication patterns of news outlets over time. Node colors represent different media sources. Clicking a node reveals the publication distribution of linked news articles in the sidebar (Figure 5).

Branch and Node Information Display. We present extracted semantic information to help users understand the event evolutionary structure. Hovering over a branch displays the branching rationale. Hovering over a node shows a node summary. Clicking a node reveals the original information and the article titles appear in the bottom-left panel. Clicking a title displays the full news article in the bottom-right panel. An example is shown in Figure 6.

A.1.2 Prompts

The system uses the following prompts. For Section 3.3.1, we use the prompt template "Node Summary" to generate node summary from news headlines. For Section 3.3.1 option 1, we use the prompt template "Non-Fork Branch Summary" to generate branch summaries. For Section 3.3.1 option 2, we use the prompt template "Branch Summary and Branching Rationale" to generate branch summary and branching rationale.

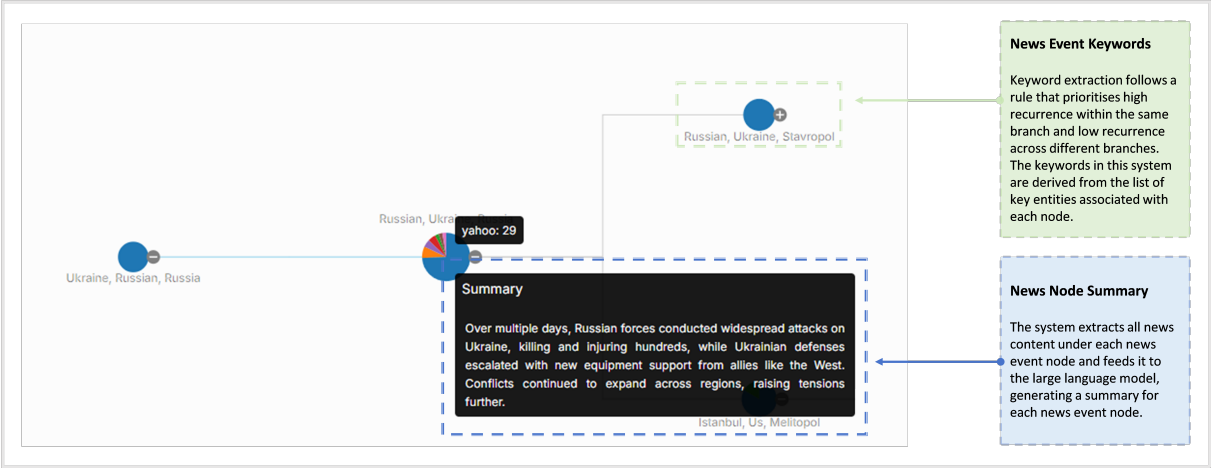

Figure 4: Node summary and display of keywords for event nodes in the user interface.

Node Summary

Suppose you are a journalist and you need to help a stranger sort out the development of an event and write a coherent summary of the event. In that case, your task is to read the following news headline(s) and summarize the events described in the news in a concise sentence of 80 words or fewer.
1. Pay special attention to the changes and development of the core entity. Strictly cover all header elements to ensure that the logical chain is complete, the dynamic process is clear, and the data is not lost.
2. The summary should include the cause, process, result, time, place, and people as much as possible.
News headlines are as follows: {News headlines}

Non-Fork Branch Summary

If you are a journalist and you are given a chain of multiple news stories, please give this news chain a 100-word summary in English. There can only be a summary in the answer, and no extra words are allowed.
News chain is as follows: {News chain}

Branch Summary and Branching Rationale

If you're a journalism person, I'm going to give you multiple follow-up stories and a central story, and the follow-up news describes different aspects of the central news. Your two tasks are:
1. Please give all follow-up news a differentiating 2-8 word English phrase to summarise the dependency relationship between follow-up news and central news, focusing on discovering the difference in their dependency relationship and the main subject in the news.
2. Please make a coherent English summary of the follow-up news and the central news in 80 words. The summary should include time, place, person, cause, process, and result as much as possible.
The answer template is as follows (the number varies according to the actual number):
The Relationship between Follow-up News 1 and Central News: Specifics
The Relationship between Follow-up News 2 and Central News: Specifics
Summary: Summary in 80 words
Follow-up news is as follows: {Follow-up news}
Central News is as follows: {Central News}

A.1.3 Hyperparameter Settings

We tuned hyperparameters manually through small-scale experiments. Users do not need to adjust hyperparameters in practice. The tuning was based on indicators such as number of branches, tree depth,

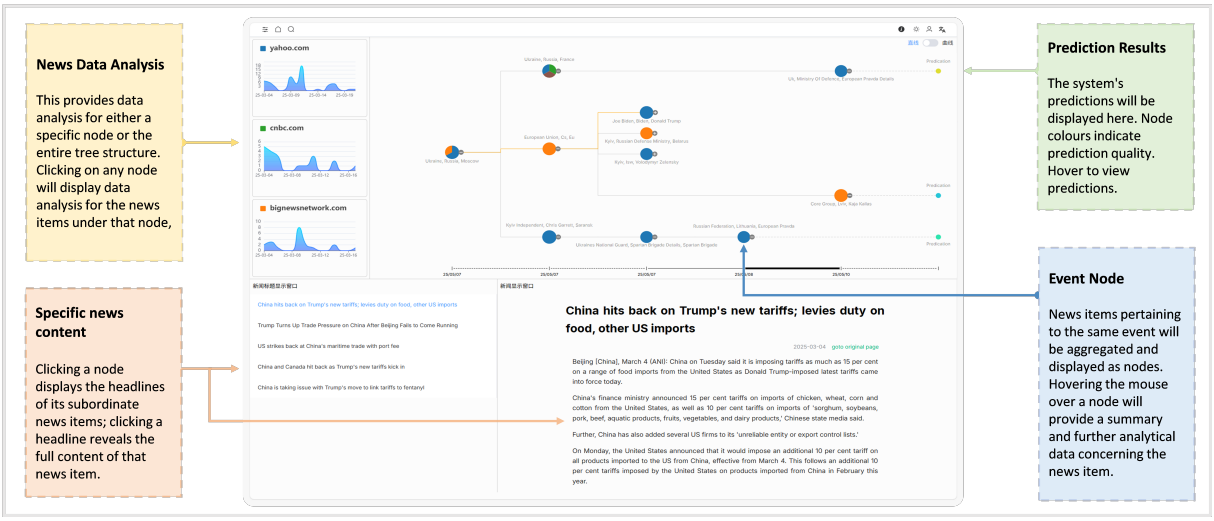

Figure 5: **System Interface:** The left and right parts show descriptions of each panel of the interface.

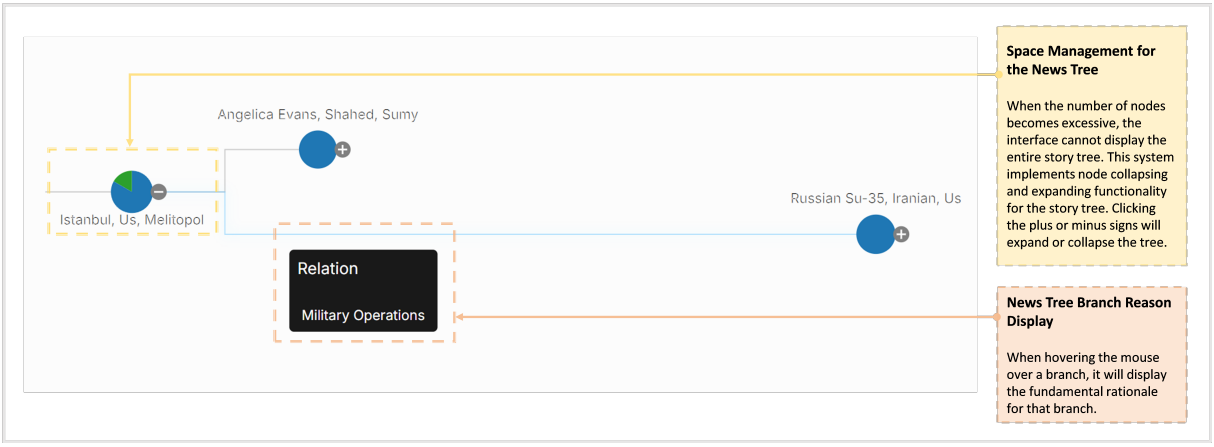

Figure 6: Demonstration of event node collapsing functionality and branch branching rationale in the user interface.

tree construction time, and user comprehension time. The following are the parameter settings:

- $\alpha_{\text{participants}} = 0.6$
- $\alpha_{\text{location}} = 0.2$
- $\alpha_{\text{object}} = 0.1$
- $\alpha_{\text{source}} = 0.1$
- $\lambda = 0.2$
- $\mu = 0.9$

A.1.4 Time Complexity of Story Tree Construction

Let m be the number of event nodes that are obtained from clustering news articles, and d be the dimension of the vector representations used for similarity computations. The following are the computational costs of key steps.

Computing Dependency Degrees. For each pair of event nodes v_i and v_j, a dependency score $dep(i, j)$ is calculated based on the similarities of lists of terms extracted from them. This involves computing vector similarities in the d-dimensional space. Since there are four lists, each similarity computation takes $O(d)$ time. Therefore, the total time complexity for this step is:

$$O(d)$$

Attaching Nodes to the Story Tree. The nodes are attached to the story tree sequentially. When attaching a node v to the story tree, an attachment score (AttachScore) is calculated. We assume that there are m nodes in the sibling branch and the parent branch. Since there are up to m potential attachment positions for each of the m nodes, the time complexity for this step is:

$$O(m^2 d)$$

Overall Time Complexity. The overall time complexity of the story tree construction is:

$$O(m^2 d)$$

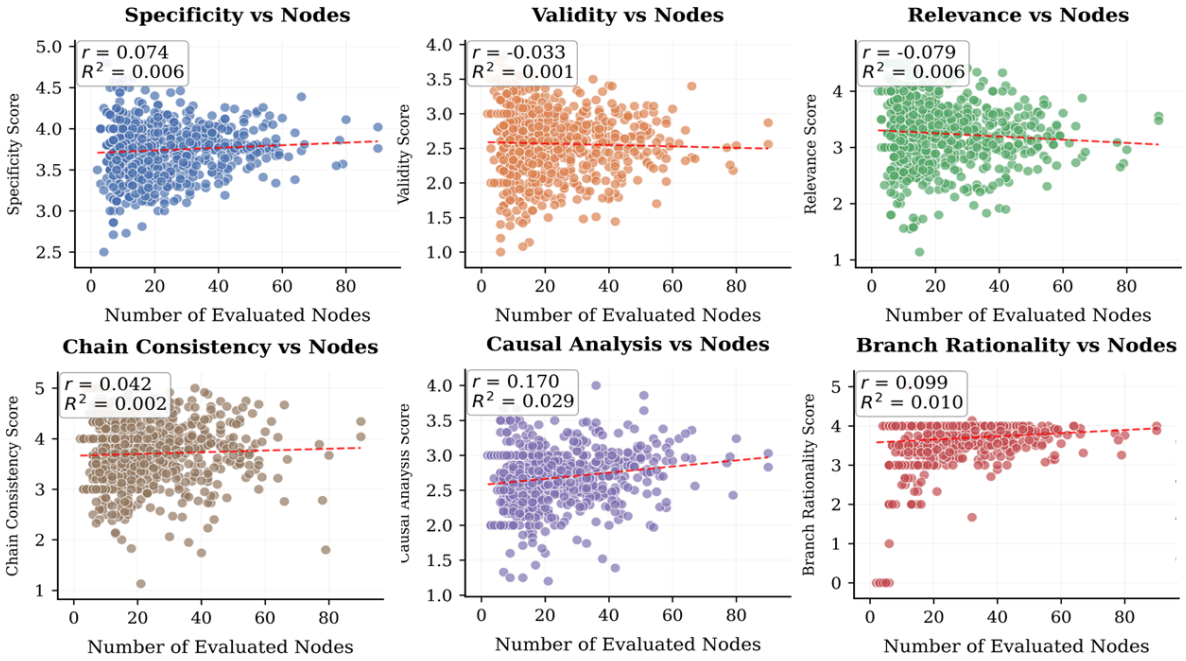

Figure 7: Change of metric values with regard to the number of nodes.

where m is the number of event nodes and d is the dimension of the term vectors.

Practical Efficiency. Our method is deployed on two A6000 servers. The entire process takes approximately 5 minutes for 800 articles, 3 minutes for 500 articles, and 1 minute for 100 articles.

A.2 NewsForest Dataset

A.2.1 Popular Topic Selection

The extraction of "popular topics" involves the following steps. First, the algorithm extracts high-frequency political entities (e.g., "United States") and event nouns (e.g., "election") from news articles from October 2024 to April 2025. Then, these terms are manually grouped into relevant topic phrases (e.g., "US election"). Finally, all generated topics are manually searched for relevant news and reviewed to ensure that they form a story tree with predictive significance.

A.3 Evaluation Method

In the experiment, we use the Tavily search API[2] to search for recent news. Specifically, the API is set to search for the ten most relevant news items within 10 days. We then pass the result to the Qwen2.5-7B to determine whether the prediction occurred. For the quality evaluation, we reuse the first five story tree quality metrics in Table 1.

[2]https://www.tavily.com/

A.4 Fine-Tuning Parameters

The fine-tuning library used in this work is LLama-Factory. The fine-tuning process is conducted on four A6000 servers with the following parameters:

Method Parameters

- **stage**: DPO
- **do_train**: true
- **finetuning_type**: lora
- **lora_rank**: 32
- **lora_target**: all
- **pref_beta**: 0.1
- **pref_loss**: sigmoid

Training Parameters

- **per_device_train_batch_size**: 1
- **gradient_accumulation_steps**: 16
- **learning_rate**: 5.0e-6
- **num_train_epochs**: 6.0
- **lr_scheduler_type**: cosine
- **warmup_ratio**: 0.15
- **bf16**: true
- **ddp_timeout**: 180000000
- **resume_from_checkpoint**: null

Dataset Parameters

- **dataset**: data
- **template**: qwen/llama/gemmma
- **overwrite_cache**: true
- **preprocessing_num_workers**: 16

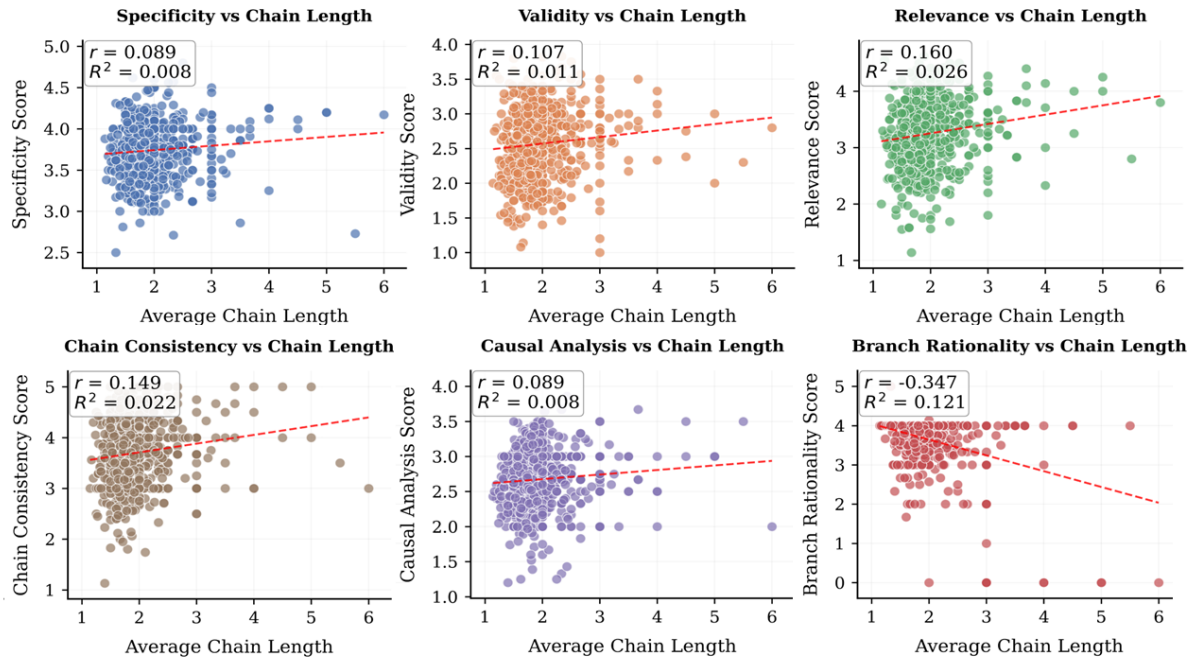

Figure 8: Change of metric values with regard to the average chain length.

- **dataloader_num_workers**: 4

Evaluation Parameters

- **val_size**: 0.1
- **per_device_eval_batch_size**: 1
- **eval_strategy**: steps
- **eval_steps**: 100