FACTTRACK: Time-Aware World State Tracking in Story Outlines

Zhiheng Lyu[1] **Kevin Yang**[2] **Lingpeng Kong**[1] **Daniel Klein**[2]
[1]The University of Hong Kong, [2]UC Berkeley
`{zhlyu,lpk}@cs.hku.hk, {yangk,klein}@berkeley.edu`

Abstract

While accurately detecting and correcting factual contradictions in language model outputs has become increasingly important as their capabilities improve, doing so is highly challenging. We propose a novel method, FACTTRACK, for tracking atomic facts and addressing factual contradictions. Crucially, FACTTRACK also maintains time-aware validity intervals for each fact, allowing for change over time. At a high level, FACTTRACK consists of a four-step pipeline to update a world state data structure for each new event: (1) decompose the event into directional atomic facts; (2) determine the validity interval of each atomic fact using the world state; (3) detect contradictions with existing facts in the world state; and finally (4) add new facts to the world state and update existing atomic facts. When we apply FACTTRACK to contradiction detection on structured story outlines, we find that FACTTRACK using LLaMA2-7B-Chat substantially outperforms a fair baseline using LLaMA2-7B-Chat, and achieves performance comparable to a GPT4 baseline. Moreover, when using GPT4, FACTTRACK significantly outperforms the GPT4 baseline. [1]

1 Introduction

Large language models (LLMs) have recently surpassed human performance across a wide range of tasks (Ouyang et al., 2022; OpenAI, 2023), yet generating long-form text remains fraught with challenges compared to tasks with shorter outputs. Even when models are trained to support context windows of hundreds of thousands of tokens, they may still struggle to retrieve and reason over such long context (Liu et al., 2023). The most advanced existing language models still take long context generation as a direction for further improvement in the future.

[1]Our code and data are at `https://github.com/cogito233/fact-track`.

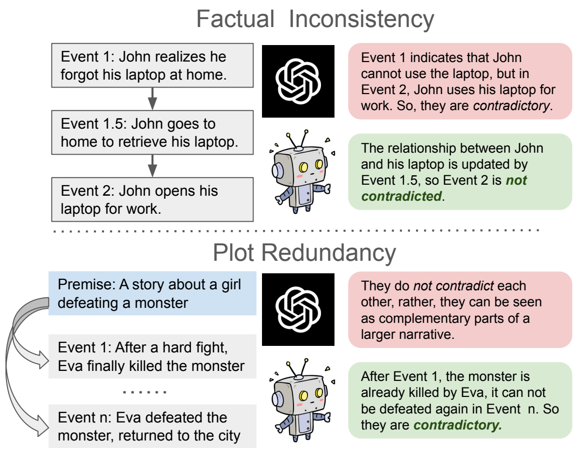
Factual Inconsistency
Event 1: John realizes he forgot his laptop at home.
Event 1.5: John goes to home to retrieve his laptop.
Event 2: John opens his laptop for work.
Event 1 indicates that John cannot use the laptop, but in Event 2, John uses his laptop for work. So, they are *contradictory*.
The relationship between John and his laptop is updated by Event 1.5, so Event 2 is ***not contradicted***.
Plot Redundancy
Premise: A story about a girl defeating a monster
Event 1: After a hard fight, Eva finally killed the monster
......
Event n: Eva defeated the monster, returned to the city
They do *not contradict* each other, rather, they can be seen as complementary parts of a larger narrative.
After Event 1, the monster is already killed by Eva, it can not be defeated again in Event n. So they are ***contradictory***.

Figure 1: FACTTRACK tackles the problems of factual inconsistency and plot redundancy. Note that those problems are based on our observations, to provide a clearer understanding of the problem, with both issues considered together in our pipeline and evaluation. For factual inconsistency detection, FACTTRACK tracks a validity interval for each fact to distinguish legitimate contradictions from facts simply changing over time. For plot redundancy detection, our method can represent the timeline in more structured form, making detection easier.

Existing works using LLMs in hierarchical generation have explored structured approaches to maintain strong internal coherence within extensive texts of thousands of words (Yang et al., 2022a,b). However, challenges remain in maintaining factual consistency and avoiding hallucinations during the generation. Especially when these problems occur in a high-level planning stage, they can greatly damage performance on downstream tasks. Therefore, a system capable of detecting factual contradictions and correcting them is essential. In Figure 1, we take story planning as an example, depicting two common issues encountered: *factual inconsistency* and *plot redundancy*.

Such issues are expected across various domains, as training corpus documents often suffer from length limitations and originate from fragmented sources, impairing the model's ability to establish long-distance, multi-state connections. Moreover, generalizing current sentence-level fact verification models (de Marneffe et al., 2008; Cai et al., 2021) to accommodate longer texts remains a challenge. We highlight two main distinctions that arise when comparing sentence-level fact statements to lengthier texts: the complexity of multi-fact contexts, and the dynamic, stateful nature of content over time.

To tackle these challenges, Section 3 introduces the concept of directional atomic facts, establishing a universal framework for time-aware contradiction detection and state tracking. Section 3.1 details the analysis of structures to identify atomic facts and detect contradictions around key events, employing LLMs for event decomposition into *pre-facts* and *post-facts*. The concept of pre-facts and post-facts is akin to preconditions and postconditions but formulated in natural text, and implying truthfulness for the entire validity time interval. This approach facilitates tracking the state changes over time prompted by events. Maintaining a comprehensive non-contradictory set of facts (*world state*) also allows for detection of contradictions by verifying and updating the atomic facts within the world state. Section 3.2 explores the broader implications of atomic facts through a more flexible and time-aware framework, introducing a *timeline* for events and atomic facts to identify contradictions or updates through their temporal overlaps.

In Section 4, we describe the main implementation of our method, **FACTTRACK**. As shown in Figure 5, we first maintain a list of pre-facts and a list of post-facts as our data structure. To update our data structure for a new event, we use a four-step pipeline: Decompose-Determine-Contradiction-Update. With this data structure, we can detect whether two events contradict each other and identify the specific pair of atomic facts in conflict, guiding the subsequent correction procedure. Furthermore, the validity interval of facts may be useful structural information in and of itself for downstream tasks.

To measure the effectiveness of our approach, we introduce a task for detecting contradictions in the planning procedure, with story outlines serving as our test domain. For event pairs flagged by a method as contradictory, we use GPT-4 to annotate whether they are actually contradictory, on a 1 to 5 scale. Experimental results show that contradictions flagged by FACTTRACK (using LLaMA2-7B-Chat as a base LLM) are judged to be real contradictions by a score margin of 0.384 more than a fair baseline using LLaMA2-7B-Chat. Additionally, when FACTTRACK is run on GPT4, the performance significantly surpasses all baselines, including the one using GPT-4 as a base model.

In summary, our contributions are as follows:

1. We introduce a framework for decomposing events into atomic facts and tracking their validity intervals on a timeline.
2. Based on that framework, we develop a method, FACTTRACK, for detecting time-aware factual contradictions in outlines.
3. We apply FACTTRACK to story outline generation, defining a task and LLM-based evaluation metrics for detecting contradictions. The results confirm our approach's empirical effectiveness.

2 Related Work

Fact Verification. Fact verification is a task widely studied in natural language processing, ranging from verifying scientific claims (Thorne et al., 2018; Wadden et al., 2020) to validating fake news (Wang, 2017; Augenstein et al., 2019). Unlike verifying claims against a database of facts, existing work has also demonstrated the feasibility of performing verification within context (Mihaylova et al., 2019; Shaar et al., 2022; Li et al., 2023). We also draw inspiration from efforts to decompose complex sentences into multiple atomic facts using LLMs (Fan et al., 2020; Kamoi et al., 2023; Min et al., 2023). Compared with prior works, our approach specifically operates on temporal structures, maintaining time-dependent fact validity intervals to handle the *dynamic* nature of the world state as facts change over time.

State Tracking. Prior work on state tracking ranges from dialogue tracking (Thomson and Young, 2010; Chao and Lane, 2019) and memory and entities networks (Sukhbaatar et al., 2015; Henaff et al., 2017) to using neural checklists (Kiddon et al., 2016) and story planning (Rashkin et al., 2020). With the advent of LLMs, state tracking in long-form story planning and generation has shifted towards explicit tracking using natural language, ranging from unstructured text (Zhou et al., 2023) to structured dictionaries (Yang et al., 2022b). Additionally, there also exists some work on generating better temporal fact validity by predicting the validity interval (Zhang and Choi, 2023), jointly modeling text with its timestamp (Dhingra et al., 2022), or utilizing Wikipedia timestamps (Jang et al., 2023). Our approach is related to the latter, but the main difference is our method operates on the generated output from a language model, and focuses on post-hoc detection rather than the calibration of a language model.

Hierarchical Generation. Hierarchical generation is widely applicable across various domains of long-form content creation, such as story generation. It can be implemented through the internal hidden states of the model (Li et al., 2015; Shen et al., 2019; Guo et al., 2021), or explicitly via natural language text or structured schema (Yao et al., 2019; Rashkin et al., 2020; Tian and Peng, 2022; Mirowski et al., 2022; Yang et al., 2022b,a). The paradigm of hierarchical generation brings both benefits and challenges to our work. On one hand, it facilitates the resolution of contradictions at higher levels compared to those detected at the bottom level or during sequential generation. On the other hand, it may require a more complex data structure to maintain the validity intervals of facts.

3 Time-Aware Atomic Facts

Existing work focuses on leveraging the semantic decomposition capabilities of LLMs for fine-grained fact-checking (Min et al., 2023). In this session, we propose an event-centric, time-aware atomic fact decomposition method. We track facts on a timeline, building toward the design of our FactTrack system. By analyzing the structure of an event and the representation of the world, we develop a method to better decompose events into atomic facts (Section 3.1) with forward and backward directions. We then discuss how to use knowledge of contradictions between atomic facts to determine the validity intervals of these atomic facts on the timeline (Section 3.2). Through this decomposition of facts and by maintaining validity intervals, our method is particularly effective for time-varying facts as is common in domains such as story writing. Examples illustrating the concepts in this section are provided in Appendix A.

3.1 Fact Decomposition

Events and Directional Atomic Facts. In narrative theory, *events* represent transitions in the *world state*, reflecting shifts that impact characters, settings, or the overarching scenario (Hühn et al., 2009). We model these transitions using *directional atomic facts*, capturing the essence of events as transformations from one state to another. Atomic fact means a basic and concise statement that conveys a single piece of information (Min et al., 2023), see examples in Figure 2. Then we employ a novel strategy using LLMs to decompose these events into distinct atomic facts with directions and a time validity interval as shown in Figure 2. These facts are then compared pairwise to assess their interrelations and impacts on the narrative structure. Recognizing that events represent transformations in the world state, we classify these atomic facts into three categories: *pre-facts*, *post-facts*, and *static facts*. pre-facts are those truths that exist before an event takes place, post-facts are truths that emerge following an event, and static facts are those truths that remain unchanged throughout.[2] To simplify the problem, we interpret static facts as a straightforward amalgamation of a pre-fact and a post-fact, and henceforth concern ourselves with only the latter two categories. Figure 2 shows an example of how we decompose event $\mathcal{B}$ into pre-facts and post-facts.

World State. The term *world state* corresponds to a set of facts that hold at a particular point in time. Compared with object-based tracking, fact-based tracking provides a more flexible state space. The world state at any given moment represents the maximum set of non-contradicting facts. For example, in a process that moves forward in time, at each new event, the world state is used to cross-check with pre-facts and is updated with post-facts. If a fact in the world state contradicts with a new post-fact, then we drop the former (Figure 2).

3.2 State Tracking on Timeline

Event Time Interval. Any given event can be decomposed into multiple sub-events according to the author's desired level of detail. For instance, previous works use this strategy to generate a story outline by doing this decomposition recursively (Yang et al., 2022a; Wang et al., 2023). To effectively model this hierarchical temporal structure,

[2]Traditional AI methods like STRIPS (Fikes and Nilsson, 1971) discuss *preconditions* and *postconditions*. Similar to our *pre-facts* and *post-facts*, they model the requirements of the world state before an action and the changes to the world state after an action. We prefer the terms pre-fact and post-fact to better express our meaning of tracking time-varying natural language facts with validity intervals on a timeline.

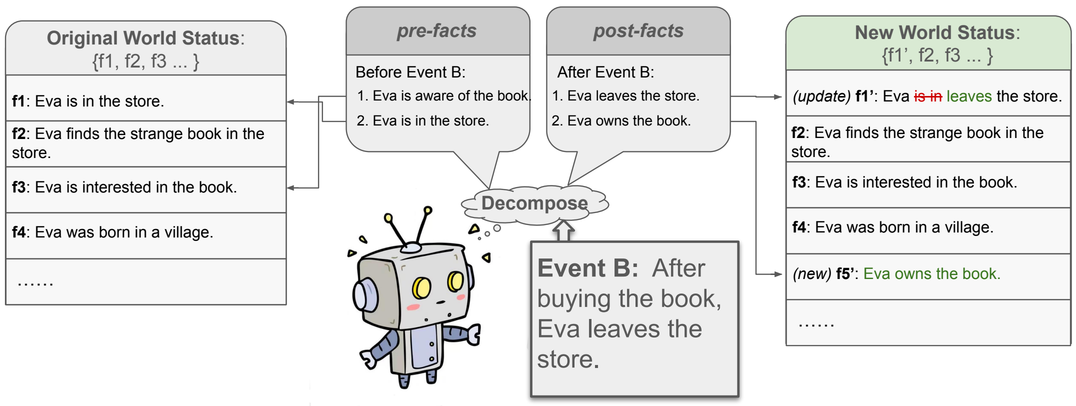
Original World Status: {f1, f2, f3 ... }
f1: Eva is in the store.
f2: Eva finds the strange book in the store.
f3: Eva is interested in the book.
f4: Eva was born in a village.
......
pre-facts
Before Event B:
1. Eva is aware of the book.
2. Eva is in the store.
post-facts
After Event B:
1. Eva leaves the store.
2. Eva owns the book.
Decompose
Event B: After buying the book, Eva leaves the store.
New World Status: {f1', f2, f3 ... }
(update) f1': Eva ~~is in~~ leaves the store.
f2: Eva finds the strange book in the store.
f3: Eva is interested in the book.
f4: Eva was born in a village.
(new) f5': Eva owns the book.
......

Figure 2: Decomposition of an event, to *verify* and *update* the world state for the event. Moving forward on the arrow of time in this example, we first retrieve all facts corresponding to pre-facts from the world state to check for any conflicting fact pairs (*Verification*). We then replace any fact in the world state that contradicts a post-fact with the corresponding new post-fact (*Update*).

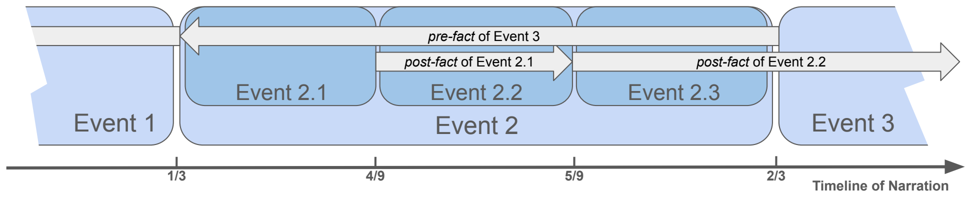

Figure 3: The timeline of narration. The start time and end time of any given event can be split recursively into sub-events. pre-facts begin at the left boundary and point to the left; post-facts begin at the right boundary and point to the right.

we define the time interval of an event as a continuous segment along the narration timeline (See Figure 3). For simplicity, we set the time interval of the entire narrative (e.g., story outline) to be $[0, 1]$. When one event ($\mathcal{B}$) is a subevent of another event ($\mathcal{A}$), $\mathcal{B}$'s time interval is contained within $\mathcal{A}$'s time interval, with adjacent events possessing non-intersecting intervals. That is, for each level of a partial hierarchical outline, if we want to generate k subevents, we have:

$$\forall i \in 1..k :$$

$$l_i = l + (i-1) \cdot \frac{r-l}{k} \quad (1)$$

$$r_i = l + i \cdot \frac{r-l}{k} \quad (2)$$

Fact Validity Interval. We now consider an event with a validity interval $[l, r]$. We assume a pre-fact f is from some time point x in the event, i.e., $x \in [l, r]$. Then we define the *validity interval* of f as $(-\inf, x]$. This implies f is valid before the time point x, but not afterwards. Similar logic applies to post-facts, where the default validity interval is $[x, \inf)$. However, as it is unclear how to pinpoint the exact value of x, we simplify the problem by assigning the default validity interval to be $(-\inf, l]$ for pre-facts and $[r, \inf)$ for post-facts.

Update Condition. To handle alterations in the world state, we introduce a mechanism that updates the interval when two facts with the same direction (either both pre-facts or both post-facts) contradict each other. The premise here is that the more "up-to-date" fact is deemed more reliable and reflective of the updated world state on the interval where they overlap. For example, consider two post-facts: "Eva is in the store" and "Eva left the store." Assuming they have the validity intervals $[\frac{4}{9}, \inf)$ and $[\frac{5}{9}, \inf)$ respectively, and are contradictory, we would need to adjust the first validity interval to be $[\frac{4}{9}, \frac{5}{9})$. As exemplified in the post-fact of Event 2.1 and the post-fact of Event 2.2 in Figure 3, we will update the former fact to make the two validity intervals not overlap with each other.

Contradiction Condition. We flag a contradiction between a post-fact from an earlier event and a pre-fact from a later event if they overlap in time and contradict each other. Suppose the validity interval of the pre-fact is $(l_1, r_1]$ and the interval of post-fact is $[l_2, r_2)$, where $l_2 < r_1$ since the pre-fact is from the later event. Figure 4 shows five possible relationships between these two facts with respect to the overlap in their validity intervals. As shown in Figure 4, a *checkpoint* indicates the boundary of an event where one of the two facts

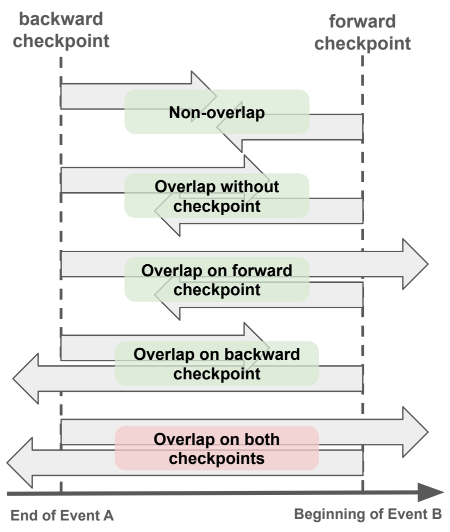
backward checkpoint
forward checkpoint
Non-overlap
Overlap without checkpoint
Overlap on forward checkpoint
Overlap on backward checkpoint
Overlap on both checkpoints
End of Event A
Beginning of Event B

Figure 4: Five possible situations for a pre-fact and post-fact contradicting each other on different points or intervals, depending on their respective validity intervals. In our implementation, we only flag a contradiction when a contradiction is detected on both checkpoints (the last situation) to maximize confidence in our predictions.

begins. Although different choices are possible, in our approach, we only flag contradictions for the case "Overlap on both checkpoints."Thus the constraint we use for flagging contradictions is:

$$l_1 \leq l_2 \leq r_1 \leq r_2 \tag{3}$$

Similar ideas also can be found in Allen's Interval Algebra (Allen, 1983), Our constraint corresponds to Allen's Interval Algebra is: $Fact_1 \; o \; Fact_2$, but there is slightly different since because different operations dictate the directionality of facts as we shown in Figure 5.

4 FACTTRACK

We are now ready to introduce our method, FACTTRACK (Figure 5). FACTTRACK can be understood as a data structure for representing a world state (Section 4.1) coupled with a pipeline of operations for updating this data structure given a new event that we want to insert (Section 4.2). Notably, events do not need to be inserted in chronological order, which is useful for hierarchical inputs such as story outlines. In our pipeline, we start by breaking down the event into several pre-facts and post-facts. Then for each fact, we conduct a series of interval operations to determine its validity interval, identify any contradictions, and finally update the validity interval of facts in world state. We also discuss how our method recognizes contradictions between two atomic facts (Section 4.3).

4.1 World State Maintaining

As shown in the light blue block in Figure 5, we maintain two lists to keep track of all pre-facts and post-facts. For each atomic fact, we store its content s_i, start time $t_{i,begin}$, and end time $t_{i,end}$.

4.2 Operation Pipeline

As shown in Figure 5, our operation pipeline consists of four steps which we execute in order:

1. *Decompose Events.* decompose a new event into pre-facts and post-facts.
2. *Determine Validity Interval.* Use the world state to find the validity interval for each atomic pre-fact or post-fact.
3. *Detect Contradictions.* Check if the current fact's validity interval contradict with existing facts.
4. *Update World State.* If needed, update existing facts as necessary and add the current fact to the world state.

Pseudocode is shown in Appendix B.3.

4.2.1 Decompose Events

In the orange block in Figure 5, we decompose the event into pre-facts, post-facts, and static facts. We use zero-shot prompting with an LLM and parse the output for structure afterward; see details in Appendix B.1. After this step, we execute the following three steps in sequence for each atomic fact.

4.2.2 Determine Validity Interval

Given a new atomic fact, we use all the facts in the world state to determine its validity interval (purple block in Figure 5). Taking a post-fact as an example, our initial default validity interval is $[l, \inf)$, where l is the right boundary of the event it came from. We then check the facts in the world state whose left boundary is greater than l, in order from left to right, until we find the first contradiction. Then, we set the right boundary of the current fact as the left boundary of the detected fact and return.

4.2.3 Detect Contradictions

As shown in the pink block in Figure 5, we retrieve all facts that overlap with the current fact, and check for contradictions. Note that this process can operate forward in time, as shown in Figure 2, as well as backward along the arrow of time. There are multiple ways to define the "overlap" of two directional segments. In our approach, we

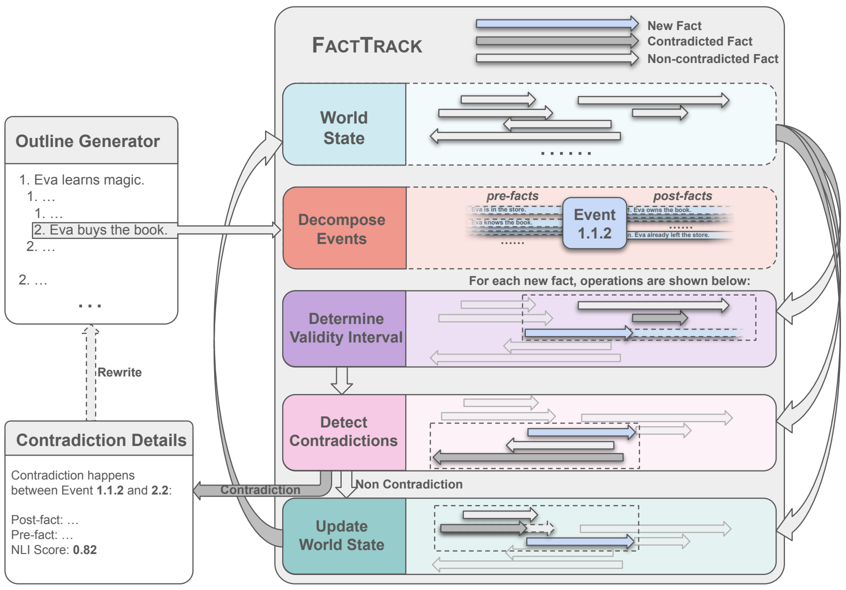

Figure 5: The general pipeline for how we maintain our data structure. We begin with a new event (e.g., plot point in a story outline), which we decompose into several pre-facts and post-facts (*Decompose Events*). For each fact, we determine its validity interval based on the world state (*Determine Validity Interval*), and then detect any contradictions with existing facts in the world state (*Detect Contradictions*). If the fact does not contradict any existing fact in the world state, then we update the world state with the new fact (*Update World State*). Otherwise, we write down details about the contradiction, and rewrite the new event conditioned on the preexisting event and details about the contradiction. Note that *Determine Validity Interval* and *Update World State* are only between facts in the same direction, while *Detect Contradictions* are only between facts in different directions.

use the strictest constraint in Figure 4, since we only care about whether there are contradictions on the *checkpoints*, where the two directional facts begin. By requiring contradiction on both checkpoints, as depicted in Figure 4, our method gains better robustness against errors in fact decomposition or the retrieval process, eliminating many false positives. After this step, if there is a contradiction detected, we can print the contradiction information as shown in the examples in Appendix F, otherwise we directly update the world state.

4.2.4 Update World State

As shown in the green block in Figure 5, if we decide to accept the event when no contradiction is detected, we will use this step to update the world state. We first use the current fact to update the validity intervals of existing facts in the world state. Taking a post-fact with interval $[l, r)$ as an example, to update the validity intervals, we retrieve all facts $[L, R)$ satisfying $L \leq l \leq R \leq r$. If those two facts contradict, then we update the validity interval $[L, R)$ to be $[L, l)$. Finally, we add the new fact $[l, r)$ to the world state.

4.3 Contradiction Recognition

We use an NLI model finetuned on outputs annotated by GPT4 (Appendix B.4). By using that model, we can give every pair of facts a contradiction score from 0 to 1 corresponding to how likely they are to contradict each other. If this score is over a set threshold, we say that the two facts contradict in our method. For the update condition and contradiction condition, we use different thresholds since false negatives for the update condition are less harmful than for the contradiction condition—it may not cause any major problems if we miss an update to a fact's validity interval, but we do not want to ignore an actual contradiction. Additionally, we also use a retrieval model as a filter to reduce the computational cost, see filtering details in Appendix B.3.

5 Evaluation

Experiment Setup. To examine our method's empirical effectiveness, we apply FACTTRACK to the task of detecting the contradictions in a story outline. While our method can also work online, detecting problems during the outline generation process and providing feedback, we evaluate only offline in this work. We employ an outline generator similar to the detailed outliner from Yang et al. (2022a), modifying the prompt to include more factual information (Appendix C.2). In particular, we follow their paradigm of breadth-first outline expansion. To balance length and knowledge density,

Method	PAIRWISE SCORE	CONTEXT SCORE
FULL OUTLINE DETECTION (GPT-4)	2.355 $\pm$ 0.163	2.859 $\pm$ 0.149
FACTTRACK (LLaMA2-7B-Chat, top 300)	2.393 $\pm$ 0.164	2.777 $\pm$ 0.146
FACTTRACK (GPT-4, top 300)	**2.599** $\pm$ 0.148	**3.133** $\pm$ 0.123
Random	1.419 $\pm$ 0.075	1.62 $\pm$ 0.087
PAIRWISE DETECTION (LLaMA2-7B-Chat)	1.452 $\pm$ 0.080	1.64 $\pm$ 0.088
FULL OUTLINE DETECTION (GPT-3.5-Turbo)	1.556 $\pm$ 0.068	1.902 $\pm$ 0.078
FACTTRACK (LLaMA2-7B-Chat, random 500)	**1.836** $\pm$ 0.092	**2.046** $\pm$ 0.092

Table 1: Comparison results of FACTTRACK (based on LLaMA2-7B-Chat) against corresponding baselines on 90 depth-3 story outlines. The contradiction scores range from 1 to 5, as annotated by GPT-4; higher is better. PAIRWISE SCORE indicates that the GPT-4 evaluator only sees the events corresponding to the two facts being checked, while in CONTEXT SCORE, GPT-4 sees the whole story outline as context. We downsample FACTTRACK's detections either randomly (e.g., random 500) or based on top contradiction scores to match the number of detections with baselines for fairer comparison; see Appendices C.3 and D for experiment details. FACTTRACK based on LLaMA2-7B-Chat achieves performance comparable to GPT4-Turbo while significantly outperforming other baselines. Additionally, when FACTTRACK is run on GPT4, the performance surpasses all baselines.

Method	PAIRWISE SCORE	CONTEXT SCORE
CoT (*without decompose events*)	2.355	2.859
FACTTRACK	**2.599**	**3.133**
FACTTRACK (*without track facts*)	2.380	2.364
FACTTRACK	**2.412**	**2.386**
FACTTRACK (few-shot GPT4)	**2.212**	**2.216**
FACTTRACK (few-shot LLaMA-7B-Chat)	2.186	2.196
FACTTRACK (NLI Model, default)	1.836	2.046

Table 2: Ablations on individual components of FACTTRACK, including removing *Decompose Events*, *Track Facts* (combining *Determine Validity Interval* and *Update World State*), and replacing *Detect Contradictions* with other methods. Both *Decompose Events* and *Track Facts* are critical to performance, while *Detect Contradictions* can be improved by changing its internal modules and thresholds.

we use story outlines with three layers, with each event having three sub-events, for 39 events in total. We estimate there are 3-5 true contradictions per outline on average (Appendix B.5). In the evaluation process, we randomly select 90 premises from the WritingPrompts dataset (Fan et al., 2018) and use LLaMA2-7B-Chat (Touvron et al., 2023) to generate the outlines; note that we run FACTTRACK on LLaMA2-7B-Chat as well. The task can be understood as a detection task: given the outline, we want to retrieve some candidate event pairs, indicating that the model considers them to be contradictory. Outline statistics are shown in Appendix D.

$$\text{Input}: [premise, event_1, event_{1.1}, \cdots event_{3.3.3}]$$
$$\text{Output}: [(id_{1,1}.id_{1,2}), \cdots, (id_{n,1}, id_{n,2})]$$

Baselines. With no directly applicable prior methods, we designed two baselines:

1. **PAIRWISE DETECTION on LLaMA2-7B-Chat**: The model compares two retrieved events to make predictions of contradictions. This approach scales well but struggles with maintaining temporal validity.
2. **FULL OUTLINE DETECTION on GPT series**: The model reviews the entire text using Chain of Thought (CoT) to identify contradictory event nodes. Limited by context window size, this method processes an outline of 39 events under our setting, averaging about 4800 tokens, using GPT-3.5-Turbo and GPT-4.

Due to varying numbers of positive detections, results were downsampled (see Appendix D).

Metrics. Due to the complexity and context-dependent nature of annotating event contradictions, we haven't found a clear method to label ground truth data, even with human input (see Appendix C.1). Upon review, GPT-4 (OpenAI, 2023) annotations proved to be higher quality and less noisy. Since contradictions aren't binary and can be nuanced and depend on context, we score them from 1 to 5 (Appendix C.3). We evaluate methods by the average contradiction score in detected pairs—a higher score indicates better detection. We developed two metric variations using GPT-4 context:

1. **PAIRWISE SCORE** directly labels contradictions by examining pairs of events directly.
2. **CONTEXT SCORE** labels event pairs within

the full outline context.

Classical metrics aren't applicable without gold labels, but we estimate precision and recall using our metrics in Appendix D.

Evaluation Results. Table 1 shows the results of our experiment. FACTTRACK on LLaMA2-7B-Chat is significantly better on the two metrics than both FULL OUTLINE DETECTION using GPT-3.5-Turbo and PAIRWISE DETECTION on LLaMA2-7B-Chat. We also achieve comparable performance with GPT4-Turbo, despite only using LLaMA2-7B-Chat in our method. When we run FACTTRACK on GPT4, the performance significantly surpasses all baselines[3] These results confirm that FACTTRACK effectively enhances contradiction detection in story planning by decomposing events and maintaining validity intervals of atomic facts. Examples are in Appendix F.

Qualitative inspection shows that event decomposition remains a bottleneck, as language ambiguity can lead to misunderstandings before decomposition. Appendix G illustrates a failure case due to ambiguity. Additionally, our method only detects binary contradictions and doesn't address more complex scenarios. Detailed error analysis is in Appendix G.

5.1 Ablation Study

We examine the roles and alternatives of the modules in FACTTRACK: *Decompose Events*, *Track Facts* (*Determine Validity Interval* + *Update World State*), and *Detect Contradictions*.

1. *Without Decompose Events*: To assess performance without the *Decompose Events* module, we substitute it with the Chain of Thought, as detailed in Table 8.
2. *Without Track Facts*: This variant excludes the *Determine Validity Interval* and *Update World State* steps, treating all facts as valid by default.
3. *Replacing the NLI Model in Detecting Contradictions*: We use the NLI model to lower the pipeline's computational cost. We explore replacing it with few-shot LLMs and estimate the potential improvements quantitatively.

[3]Note that while we used GPT-4 to generate the fact contradiction data to finetune our NLI model to work on more complex text, GPT-4 is not directly used in the pipeline of FACTTRACK on on LLaMA2-7B-Chat. See details in Appendix B.4.

Experiment	Precision	Recall	F1
GPT4o-mini	**89.29%**	5.57%	10.48%
FACTTRACK	52.71%	**62.81%**	**57.32%**

Table 3: Performance of the baseline and FACTTRACK in the Binary Judgement experiment in ContraDoc. FACTTRACK was implemented using GPT4o-mini, with splitting events by sentences conducted via the nltk package.

Results Table 2 shows that both the *Decompose Events* and *Track Facts* modules are critical to performance. However, the *Detect Contradictions* module can be improved by using few-shot versions of GPT-4 and LLaMA-7B-Chat, suggesting that more fine-grained annotation and knowledge distillation could enhance performance further. This indicates that while FACTTRACK has already substantially outperformed the baselines, there is potential for even better performance.

5.2 Document-level contradict detection

We extended the application of FACTTRACK to the ContraDoc dataset (Li et al., 2023), a dataset for evaluating document-level contradictions. In alignment with the original study, we employed a binary judgment framework, which involves directly prompting large language models (LLMs) as a baseline. Our approach, FACTTRACK, demonstrated a better F1-score compared to the baseline models. The observed reduction in precision relative to the baseline can be attributed to the presence of borderline contradictions within the documents and the ambiguity in the decomposition process. The results are detailed in Table 3.

6 Conclusion and Future Work

As the complexity of texts generated by LLMs increases, understanding their structures and tracking time-varying factual information becomes an increasingly important bottleneck in long-form generation. In this work, we have introduced FACTTRACK, a fact-tracking framework that decomposes events into pre-facts and post-facts and maps them onto a timeline, facilitating the tracking of narrative progress and detecting contradictions between events. Experimental results show that when we apply FACTTRACK to contradiction detection on structured story outlines using LLaMA2-7B-Chat as the base model, our method achieves performance comparable to GPT4 and significantly outperforms baselines using the same base LLM. Furthermore, when we run our method on GPT4, we find that its performance significantly outperforms all baselines.

By effectively maintaining factual consistency over extended contexts, we envision FACTTRACK serving not only as a fact-tracking module for complex content planning but also as an efficient automatic evaluation metric for contradictions in extensive texts. We additionally envision several further possibilities for improving our system, such as enhancing decomposition accuracy, structuring atomic facts more effectively (for instance, based on entities), and/or maintaining timelines in narratives that do not follow a strict chronological order. Moreover, although we only experiment on story outlines in this work, FACTTRACK is in principle generally applicable to other domains, and we hope that our framework's capacity for managing time-specific knowledge could be of use in other areas as well, such as detecting fake news and dynamically updating knowledge bases.

Limitations

The difficulty of identifying all contradictions and partial contradictions significantly constrained our evaluations (see Appendix C.1), limiting us to obtaining gold labels of contradictions. Therefore, we used GPT-4 to annotate the contradicting score as a proxy for the evaluation task.

The context window size of baseline and evaluation metrics also restricted us from running experiments on outlines much longer than the current 2000-3000 words. While FACTTRACK is capable of detecting contradictions in such outlines with near-linear cost growth, we did not evaluate them in this work due to the potential performance degradation of GPT-4 in longer contexts.

Additionally, the challenge of conducting thorough evaluations impacted the system's development. Many decisions, such as prompt design, the choice of models for each module (e.g., the few-shot language model outperforms the NLI model as shown in Table 2), and the selection of hyperparameters (e.g., thresholds for the contriver and NLI models), were made manually rather than through rigorous validation. As a result, there may be considerable room for improvement in the detailed design of individual modules.

Finally, FACTTRACK's performance may decrease on LLMs that lack strong generation and instruction-following capabilities.

Ethics Statement

Since FACTTRACK is built upon existing LLMs, we may inherit any potential biases and harms from those systems. However, in FACTTRACK, we focus on tracking facts and detecting factual inconsistencies during the process of creative story generation, with an emphasis on the interpretability of the world state within narrative structures. Our focus on factuality limits the potential abuse of language models and may be a useful tool for mitigating such abuses in the first place.

FACTTRACK is also currently designed only for English, although translating our prompts to other languages shouldn't be difficult in principle. However, performance might suffer in lower-resource languages, depending on the base LLM.

Reproducibility Statement

We saved all intermediate computation results, including our NLI dataset as well as results from the decomposition and inference steps. The finetuning process of the NLI model is also described in the appendix; otherwise, we use existing open-source or API-based LLMs for inference with temperature 0. Thus we believe our work is highly reproducible, and all of our code and data will be open-sourced upon publication. However, as we use the OpenAI API in some of our experimental comparisons, we cannot rule out the possibility of minor differences in inference results due to future API updates.

References

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843. 5

Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. 2019. MultiFC: A real-world multi-domain dataset for evidence-based fact checking of claims. 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 4685–4697, Hong Kong, China. Association for Computational Linguistics. 2

Deng Cai, Yizhe Zhang, Yichen Huang, Wai Lam, and Bill Dolan. 2021. Narrative incoherence detection. 2

Guan-Lin Chao and Ian Lane. 2019. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. 2