EventHopNLI: A Functional Dataset for Systematically Diagnosing Logical Failures in LLM Temporal Reasoning

Ved Mathai
University of Oxford, UK
ved.mathai@eng.ox.ac.uk

Janet B. Pierrehumbert
University of Oxford, UK
janet.pierrehumbert@oerc.ox.ac.uk

Abstract

This paper presents EventHopNLI, a simplified functional diagnostic dataset for the task of event temporal ordering. This paper uses this diagnostic dataset to improve the interpretability of the performance of attention-based language models on this task. Existing datasets based on natural data have multiple overlapping linguistic features. Simplifying and isolating these features improves interpretability. EventHopNLI is a programmatically-created NLI dataset that systematically varies over various complexity factors such as number of events, number of logical hops etc. Even though EventHopNLI is highly simplified, it still proves challenging to language models. Being functional, the dataset is dynamic. This reduces the risk that the data is available to language models during training. We ablate over the different complexity parameters and illustrate different shortcomings of attention-based models at this task. We discuss the performance of RoBERTa-large, Llama-405B and GPT-4o. The code and data is available at https://github.com/vedmathai/eventhopnli.

1 Introduction

Identifying events in time and reasoning about their relationships is critical for many NLP application areas such as text summarization and fact checking. However, off-the-shelf large language models perform rather poorly in temporal reasoning (Xiong et al., 2024; Wang and Zhao, 2023). Despite the advances they report in building explicit temporal graphs and applying chain-of-thought reasoning, the problem cannot be viewed as solved. One reason the task is difficult is that multiple linguistic factors need to be brought to bear concurrently. A related challenge is that human annotators find it difficult and sometimes confusing to supply explicit temporal annotations, and as a result annotated natural data is expensive and often noisy. A core challenge – and the one that the present paper addresses – is that reasoning about temporal relationships is a multi-hop reasoning problem on a graph of relationships, and classic algorithms for solving such problems exactly are recursive.

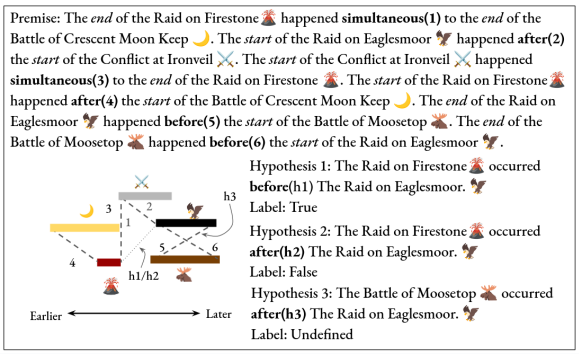


Figure 1: An example of the generated premise and an illustration of the timeline that the premise depicts. The actual dataset does not contain the emojis. They have been included in the figure to help readers parse the paragraph. The dataset is created to test the upper-bound of language models on the the temporal ordering task. The current naturalistic datasets are have confounding linguistic features making it hard to identify specific areas of improvements. Simplifying the task to this format helps us better understand models' ability to perform multi-hop reasoning for temporal reasoning.

Most annotated temporal datasets (UzZaman et al., 2013) are created by asking experts or crowdworkers to provide temporal labels on naturally occurring text articles. In general, the set of temporal relationships in a text has a $\mathcal{O}(n^2)$ complexity on the number of events. Annotating all relationships quickly becomes intractable as the text length increases, so it is necessary to select events for which relations are to be annotated. UzZaman et al. (2013) leaves the choice of the pair of events for which a relationship is to be identified to the annotators. As a result, these datasets are sparsely labeled. Later annotation efforts attempt to remedy the sparsity problems. For example, Cassidy et al. (2014) make annotators provide temporal relationship labels between all events in a two-sentence

Proceedings of the 2025 CLASP Conference on Language models And RePresentations (LARP), pages 11–27
September 8-9, 2025

window. Ning et al. (2018) perform two-rounds of annotations. The first classifies the member of events to multiple orthogonal axes, i.e., one axis for events that have happened and another axis for events that are only planned to happen but haven't as yet. The second round of annotation then places these events in temporal order on their respective axes. This discussion illustrates that the annotation completeness is already limited by the specific design of the annotation process.

One additional – and critical – limitation of current datasets is *Lack of diversity*. Most annotation efforts were carried out on news data, which is dominated by the past tense. This introduces biases which can limit performance on other types of text, such as planning documents. A second critical limitation is linguistic coverage. The materials are not balanced or controlled for the linguistic markers of temporal relations. This makes it impossible to carry out a diagnostic evaluation of which types of expressions are the causes of difficulties in performance.

Both of these points motivate EventHopNLI. The dataset focuses on analysing the performance of models when multiple logical hops are required to perform the temporal reasoning. We systematically vary the complexity in terms of number of events, relationships, hops and analyse the performance. An inventory of the full variety of linguistic features found in naturalistic data can be found in §2. A technical description of how the EventHopNLI dataset isolates and balances for the specific features in its scope is found in §3.

Our contributions are the following:

1. **The dataset**: EventHopNLI, which is presented in the form of a Natural Language Inference task. Each datapoint presents a paragraph (premise) and a corresponding claim (hypothesis) about the temporal order between two events in the paragraph. The task posed to the model is to decide whether the claim is true, false or undecidable due to logical inconsistencies in the text.

2. **An illustration of programmatic data generation for the task of temporal relationship extraction**: As discussed above, EventHopNLI is designed to alleviate the problems of imbalance and excessive overlap of linguistic phenomena by generating the data points programatically. This forms a functional test (Fan et al., 2024). Static benchmarks run the risk of being accessible to language models during their training. Evaluating on data that the language model has already seen during their training doesn't give us a true estimated of their expected behaviour on out-of-domain data. Functional tests create data that tests a particular functionality but has enough adaptation to the lexical representation of the data. Therefore, it guarantees that the evaluation provides a true estimate on the out-of-domain examples. We hope that this, along with the previously motivated argument to isolate linguistic phenomena when designing temporal datasets provides a template for future temporal datasets targeting other linguistic phenomena.

3. **Analysis of the model performance on this dataset.** We report the performance of three models on EventHopNLI. We find that even the best-performing model struggles on the dataset, for all but the simplest premises. Given that we have distilled down the dataset to a very simplified version with all the additional challenges removed, it is clear that reasoning capabilities beyond those provided by a transformer model using pure attention will be required to fully solve the problem of temporal-order classification.

The rest of the paper is as follows: §2 analyses the different linguistic features a model would need to have the ability to understand in order to perform the task of temporal-ordering. §3 describes the data and its creation. §4 describes the experiments performed on the dataset, followed by the presentation and analysis of the results in §5.

2 Desiderata

2.1 Target particular linguistic features while abstracting away from other features

Understanding temporal relationships involves understanding multiple linguistic features.

We argue that to systematically understand the failure modes of language models, datasets have to be designed such that linguistic features are isolated from each other in the datasets.

The specific linguistic feature that EventHopNLI focuses on are **temporal markers** such as *before*, *after*, *during* etc. that position events in relations to each other in time. EventHopNLI analyses how language models perform on chains of such relationships in text.

Below, we enumerate the linguistic features that often co-occur in temporal and event expressions. The text in EventHopNLI is designed such that the choice of label does not depend on any of the following features. In Appendix A, we take an exam-

ple from the TempEval dataset and show how these features often occur in overlapping ways. Note that each of the following are often studied substantially in isolation in theoretical linguistics. For each, we provide an explanation that highlights the main difficulty of that feature.

Events embedded under a speech-act verb: For example, *fly* in *John said that Mary will fly tomorrow*, is embedded under *said*. The temporal location of embedded events is understood in relation to that of the speech act.

Events presupposed by a corresponding state: Referring to an entity being *dead* presupposes that a *death* event happened sometime before.

Vagueness: Stating that World War II happened during the last century specifies a broader time window than saying that it took place from 1939 to 1945. Such vagueness can lead to uncertainty in inferences about temporal ordering.

Accomplishments/Achievements/Processes: If a process of drawing a circle stopped halfway then a circle is not drawn. But if the process of jogging is stopped halfway, it is still true that jogging took place.

Entity coreference resolution: The same real-world event may be referred to by different lexical descriptors. These co-references have to be correctly resolved to create the chain of event occurrences. A related yet important point is that a noun such as *recovery* alludes to a verbal event of an entity *recovering*.

Irrealis events are events that were or are planned to occur and haven't or are yet to occur. These are more difficult to model on a timeline because they may not occur or many simultaneous timelines of the future may exist.

General knowledge and commonsense: Inaugurating a new president of the USA entails that an election has taken place. Understanding this entailment involves the use of world knowledge about how elections occur.

In the next section, we show how the text of EventHopNLI is designed such that a model that has to solve EventHopNLI task (described in the next section) does not have to understand any of the linguistic features mentioned above.

2.2 Provide for rich ablations

Balancing the data across different parameters and labeling individual data points with the parameter value allows us to recognize patterns of failures in a model's behaviour. The data should be balanced for size and temporal relationships. The dataset should tackle questions such as: Are models able to decipher ambiguity in the facts stated? Do language models benefit from the temporal relationships being presented in a sorted order? It should also probe whether difficulties are intrinsic to the problem or to the lexical form of the temporal domain.

2.3 Avoid the need for human annotations

In §1 we discussed the general difficulty previous annotation efforts have faced since this task is difficult for humans too. By programmatically generating the data we can i) inexpensively create a large dataset ii) balance the dataset across a range of attributes and their values iii) guarantee the availability of a ground truth, correct, answer for each query. Inspired by (Kim and Schuster, 2023), we argue that it is beneficial to create data programmatically, with controllable parameters that systematically limit the number of linguistic factors that need to be simultaneously used.

Generating data does not preclude the need for annotated natural data. Performance metrics obtained on annotated natural data gives us a good understanding of how models will perform on data that is available in production. However, in order to improve their performance, it is important to diagnose where, why and how they fail so that future interventions can be made to either the algorithm or training data to improve performance. Testing on data that is isolated by linguistic features helps us understand the features that prove difficult for the model. Such data with isolated linguistic features can be obtained by filtering natural data, which is labour intensive and expensive. Alternatively, it can be obtained by generating data: the method used by this paper.

3 Data

Formally, we have a set of events $\mathcal{E}$; and relationship types $\mathcal{T}$ = {*before*, *after*, *simultaneous*}. Each event has a *start* and an *end*. The premise is a set of temporal relationships r that are in the format of a triple ($e_1^{start/end}$, t, $e_2^{start/end}$) where $e \in \mathcal{E}$ and $t \in \mathcal{T}$, the superscript of e indicates the extremity and the subscript indicates the index of e. The hypothesis is of the form (e_1, r, e_2) where $r \in \mathcal{R}$ and $\mathcal{R}$={*before*, *after*, *overlaps*}. The label $l \in \mathcal{L}$ and $\mathcal{L}$ = {*true*, *false*, *undefined*}. The task is a modification of a natural language inference (NLI) task.

Each data point consists of a premise, a hypothesis (or claim) and a label.

The premise is a set of n relationships between m events while the hypothesis makes a claim about a single relationship r. A simpler formulation of the problem would involve the temporal relationship between point events. However, we choose to test durative events because they form an inherently harder problem and one that forms a better representation of those events available in the natural texts. In our formulation, one has to keep track of the extremities of the event and compare all four points in order to find the ordering of events versus just the two required of point events. This includes keeping track of when the events mentions are under-specified, i.e., only the start or the end of an event is mentioned.

1) ***true*** labels cases where the hypothesis is true given the premise. 2) ***false*** labels cases where the hypothesis is false given the premise. A pair can be false either because an alternative claim is true or no claim can be derived. For example: if the premise makes a claim (e_1^{start}, before, e_2^{end}) and the hypothesis makes the claim (e_1^{start}, after, e_2^{end}). Then the pair is labeled as *false*. 3) ***undefined*** labels cases when there are two more logical paths that involve the two events that present in a cyclical chain. For example, i.e., both (e_1^{start}, before, e_2^{end}) and (e_1^{start}, after, e_2^{end}) exist or are derivable from the evidence available.

In general, there are 13 different temporal relationships (Allen, 1983). It is undesirable to tie the architecture of a model to the subset of relationships that need to be classified. Setting up an NLI task allows the set of labels to remain constant while the set of temporal relationships tested can be varied easily without affecting existing learning architectures.

Examples of generated data points are presented in Fig.1. The following are particular design choices and the particular desiderata that they address:

Dataset is agnostic to world knowledge: Our dataset has the flavour of a miniature language that could be employed in the context of a multi-player on-line game. By using fictional battle names such 'The Raid on Firestone', we guarantee that the proposed evaluation methodology is testing the model's ability to understand the logic of temporal relationship markers and not using any facts about real historical battles that it may have learned during its pre-training phase.

Minimalistic descriptions of temporal relationships: The premise is made up of only relationships in the form of start/end of event 1 after/before/simultaneous start/end of event 2. Using the full names of events in the specification of every relationship removes the need for models to apply entity coreference resolution that is more complex than simple lexical matching. No commonsense or world-knowledge beyond that of the temporal relationship markers has to be applied. There is no verbal event therefore there is no need to understand tense. All events are realis, (i.e, have actually occurred versus possibly occurring in the future) and are named events (i.e, there is no use of anaphora or events rooted in verbs). There are no descriptions of states.

Ablating over sizes: Ablating over the sizes of the premise in terms of relationships, events, logical hops[1] gives us an understanding of how increased complexity due to size and increased number of relationships affect language models' performance.

By ablating over the **temporal relationships** in the premise we can identify if any particular temporal relationship proves to be more difficult than others. We predict that *simultaneous* would be more difficult than that of *before* and *after* for the following reasons.

Finding the temporal order between events or event identifying contradictions in the premise involves performing a depth first search on the graph. Traversing a graph with only *after* and *before* relationships is computationally easier than traversing a graph with *simultaneous* relationships. Both traversals involve maintaining a stack. However, with *simultaneous* relationships the stack size at any given point can be larger.

We explain with an example: Assume event *A* is earlier than event *B*. It is not necessary to add events earlier than *A* or later than B to the stack since they do not provide useful information to the temporal chain that connects event *A* and event *B*. However, assume a simultaneous relationship between *A* and *C*. Now all of the nodes related to *C* will have to be added to the stack, which means that with many simultaneous relationships there are

[1]We define a **temporal chain** to be a sequence of two or more events ordered temporally such that traversing the chain provides the temporal relationship between the two events at its endpoints. We further define each individual relationship on the chain to be a logical hop.

higher chances of the size of the stack being larger than if there were no simultaneous relationships.

Sorting: Providing the temporal relationships in the premise in a sorted order resembles how events are often described in text, i.e., in temporal order. Sorting relationships reduces the number of permutations in which the same timeline can be expressed to just one. This would increase the probability that patterns between the train and test sets repeat.

Comparison with other domains: Testing on only the temporal domain raises the question of whether the difficulty of the task is intrinsic to the task of timeline resolution or to the lexical properties of the temporal relationships. The dataset consists of computationally equivalent tasks but in two following domains:

i) **Spatial domain**: Events are mapped to locations and the start and stop points are mapped to east and west edges and the temporal relationships are mapped to *east of* and *west of*. An example is provided in Table 4.

ii) **Logical axioms:** Relationships are mapped to $\{<, >, =\}$ and the event names to simple tokens. This formulation strips away the natural language aspect of the task.

Generalizability versus memoization: We expect a language model to generalize the logic of temporal relationships from the training data while not memoizing (learning by rote) event names, specific lexical forms or timelines. We create train-test pairs that individually keep event names, lexical forms or timelines common between the train and test datasets.

Impossible logical chains: A logically coherent temporal graph is a directed acyclic graph. If a cycle exists in a temporal graph then the temporal relationships among the events on the cycle are undefined. Annotated natural text data may have cyclic relations either because the text itself contains contradictions, or because the annotators made mistakes. A language model should recognize such cycles and report them. Our dataset contains instances of cycles only by design, when the claims in the text are contradictory.

A note on complexity of the task: Since the data points were created using an algorithm and natural language templates, it is trivial to write a parser that parses the data points back into a temporal graph to obtain 100% accuracy. This places performance of the language models in perspective. Kim and Schuster (2023) argues that their task of entity-tracking may be hard for humans when the stimuli, which are quite long, are presented orally. However, when provided with a scratchpad, humans were able to solve the task exactly. Our task is similar in that it can be solved exactly with a scratchpad and careful reasoning. Human annotators may not obtain a perfect score due to factors such as fatigue, carelessness or simple errors. However, when provided with a written version, a scratchpad and no time constraints, one of the authors was able to solve the question exactly. With these experiments, we provide similar standards to the language model. We ask if the language models can solve the task exactly if they had no time constraints. Language models are increasingly being deployed in applications that help students and researchers who expect a high level of correctness from the models especially on such logical problems that don't involve subjective decisions (Kooli, 2023), therefore it is instructive to understand the upper-limit of their logical performance.

3.1 Generation of the EventHopNLI dataset

The following are salient points about the implementation details of the program that generates the dataset, more information including pseudocode is included in Appendix 8.

The program has four sections: i) timeline data structure; ii) timeline generator; iii) verifier; iv) data-structure-to-natural-language translator.

Timeline data structure: The timeline itself is a graph of events connected by temporal relationships. The events are partially ordered and the dataset is balanced to have timelines with internal contradictions.

Timeline generator: The set of fictional battle names are generated using ChatGPT. We create a list of names for the test set mutually exclusive from the names in the train dataset. The extremities (start or end) of two events are selected at random and a relationship with a specific temporal relationship is generated between them.

Verifier: In some cases in the timeline graph there may be internal logical contradictions. A hypothesis is generated by choosing two events and assigning a relationship between them. The verifier traverses the generated timeline graph and decides whether the hypothesis is *true*, *false* or *undefined*.

It is possible for the relationship between the two

events in the hypothesis to be under-specified. For example, given two events A and B, the premise specifies that the start of event *A* is before the start of *B*, however, the premise does not specify if the relationship is between the end points. Assume the claim in the hypothesis is: event *A* being before, after or overlapping with *B*. It is unclear which one of these temporal orderings is true without knowing the exact temporal ordering between the end-points. Therefore we label the pair as *false*.

A closed cycle in the graph is evidence of a logical contradiction. Using a depth-first search traversal on a directed graph allows us to find cycles easily, because visiting a previously visited node would present a cycle.

Data-structure-to-natural-language translator: The temporal graph is converted to natural language descriptions using templates. Later, we describe how we create different sets of these templates for an ablation study.

3.2 Balancing EventHopNLI and defining its variations

We balance EventHopNLI across the following attributes: {number of events, number of relationships, number of hops, whether an internal logical contradiction exits, temporal relationship, label}.

The **number of events** vary over a exponentially increasing set of sizes, i.e., $\in \{4, 8, 16, 32\}$.

The **number of relationships** vary as function of the number of events. *number of relationships = min(max(number of events, 3), 32)* where *relationship_number = relationship_multiplier*$\times$ *number of events* and *relationship_multiplier* $\in \{0.5, 1, 2\}$.

The dataset represents a Cartesian product of the following form: *number of events* $\times$ *number of relationships* $\times$ *temporal relationship* $\times$ *whether there exists an internal contradiction* $\times$ *label*.

We create different train-set tests with the following strategies. Each of the strategies inform us about the models' capabilities.

Each of the train datasets are about 14,000 data points. The test set are 2,500 data points.

Strict: In this pairing, the names of events, natural language templates for the relationships and the timelines are mutually exclusive between the train and test set. Naturalistic data would have a high degree of mutual exclusivity between the train and test set, therefore an efficient model would be expected to not perform memoization.

Same names/templates/timelines: To systematically check for the models ability to generalize vs. memoizing (cf §3) we remove the mutual exclusivity between the train and test set and create the following three sets of train datasets. In each one of i) **event names** ii) the **natural language templates** used to create the premise iii) the set of **timeline graphs** are maintained as common information between the train and test set.

Random	0.323
RoBERTa standard	0.762
RoBERTa spatial domain	0.66
RoBERTa logical domain	0.79
Llama 405B strict	0.4
GPT-4o strict	0.36

Table 1: Main results reported on the standard specification of the dataset

Ablating temporal relationship type and sorting: We create the following deviations from the strict dataset by varying the relationship types allowed and sorting the relationships in the premise temporally:

i) only after and before relationships

ii) only after and simultaneous relationships

iii) only before and simultaneous relationships

iv) only before relationships: this is the same as (iii) but the *after* relationships are inverted to make them *before* relationships

v) All relationships sorted [2]

vi) Only before relationships sorted: same as (iv) but sorted temporally.

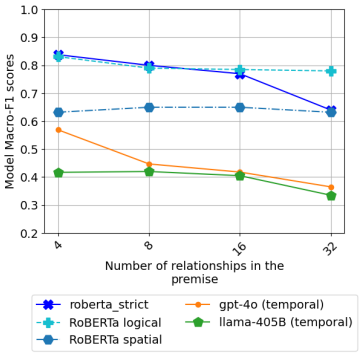
These ablations will inform us i) whether the *simultaneous* relationship-type is indeed more difficult than *after/before* ii) Whether sorting relationships temporally benefits learning because it reduces the entropy in terms of presentation of information.

Alternate domains: Following §3, we create two sets of test-train pairs. One for the spatial domain and the second for the logical domain.

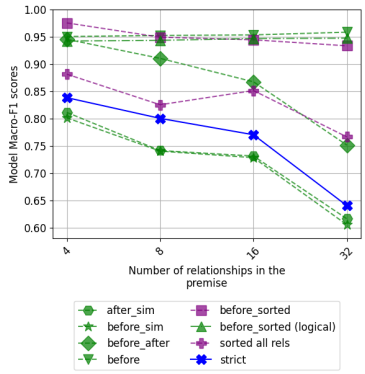
4 Experiments

Finetuning-based experiments: We experiment with fine-tuning a RoBERTa-large (Liu et al., 2019) model on the given data to see how well the model can learn this task with fine-tuning. We deliberately

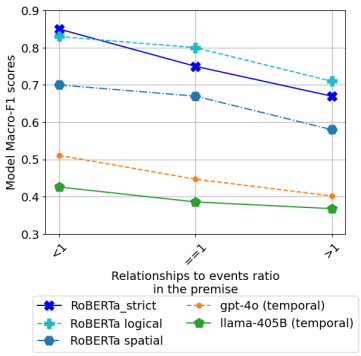
[2]All of the cases in which the premise is sorted, the relationships are sorted temporally by the earliest event present in the relationship. Arbitarily choosing events in the case of logical cycles.



(a) Performance of RoBERTa on different domains



(b) Ablation performed with RoBERTa on datasets systematically designed to maintain common information between the train and test set.



(c) Ablation study performed on the datasets with the data points partitioned by the relationship-to-event ratios in the premise.

Figure 2

use RoBERTa because it is an example of a large model which can easily be fully fine-tuned without the use of adapters.

We use a learning rate of 1e-6.

In-context demonstration: In this setting, the large language model is prompted with only the task description, label description and the expected response format as shown in Table 3. We prompt GPT-4o (Hurst et al., 2024), a large commercial closed model [3] and Llama-405b (Grattafiori et al., 2024), a large open-weight model.

It is instructive to evaluate the ability of these large-language on this task without any fine-tuning, with the understanding that a low performance on this task will automatically translate to a low performance on an even more complex task.

5 Results

We use macro-F1 as the basic metric for all of the results reported.

5.1 Simple Baselines

We report the **simple baseline** of random selection to show that the dataset is both not trivial and is balanced. An F1 score of 0.33 on the random selection shows that the dataset is balanced.

5.2 Main Results

Table 1 reports the main results. Each of the result reports the mean of five runs.

[3]We attempted experiments on GPT-o1, OpenAI's latest reasoning model and we found the the results varied drastically between multiple runs over multiple weeks. Therefore, we do not report the performance.

Performance drops as complexity increases Fig. 2a shows that as the number of relationships increases, the performance of the models deteriorates. This is an indicator of the expected performance of models on production data. Timeline creation is a direct downstream task of temporal relationship extraction and regardless of having a larger context window, lower performance can be expected from the models when the number of relationships that need to be parsed increase.

The rate of decrease in performance as the number of events increases (shown in Fig. 3a) is not as prominent as when the number of relationships increase, this indicates that it is the number of relationships and not the number of events is the source of complexity for the task.

Fig. 2c provides further evidence. It plots performance as a function of the relationships to events ratio. Decreasing performance as the ratio increases shows that the performance is more influenced by the number of relationships rather than events.

As the number of logical hops required increases, the performance first falls and then asymptotes out. (Fig. 3b). We attribute this to the fact that as the number of permutations in which smaller temporal chains are presented is exponentially smaller than the ways relationships with larger temporal chains can be presented. This makes it easier for the model to memoise the different patterns that represent the temporal chains.

Surprisingly, this effect is not seen in GPT-4o, suggesting that its higher number of parameters, larger training data and larger context window enable it to perform the same regardless of the number

of hops required to solve the particular data point, even though its overall performance is low.

Larger parameter size and pre-training data does not provide an advantage The performance of models with a larger parameter capacity on this simplified dataset shows that even in its highly-simplified formulation temporal relationship extraction is still computationally challenging for language models.

The fine-tuned RoBERTa model performs much better than the few-shot prompted larger models (Llama-405b and GPT-4o). This suggests that the data that very large language models learn from is too noisy for them to effectively learn temporal reasoning.

The complexity is intrinsic to the problem and does not arise from the lexical constructions.

In Fig. 2a, we see that the temporal domain and logical domain perform similarly, except for the case of 32 relationships, where the logical domain maintains its performance. This may be because the number of tokens is lower in the logical domain, and never exceeds the limit of the context window.

The spatial domain is intrinsically harder to parse than that of the temporal domain, which shows that the temporal domain is not a lower-limit for the task.

RoBERTa performs detrimental memoization The performance of `same_timelines` and `same_-names` are both lower than `RoBERTa_strict`. This shows that during fine-tuning RoBERTa is performing memoization (i.e., learning axioms by rote) rather than learning the patterns in a generalizable manner. (Fig. 3c). This is undesirable behaviour from a model that is expected to generalize to new information.

We see in the same figure that when the test set uses the same natural language templates as the train set, performance improves which means the fine-tuned model struggles to extend to different sentence constructions that express relationships.

Sorting the relationships temporally improves the performance of the NLI classification. By sorting the timelines, the number of permutations to describe the same timeline reduces to just one. This in turn increases the probability that patterns will repeat between the train and test sets.

These results allows us to predict the performance of models of different types of data that may be encountered in production: better performance can be expected on data that follows a chronological structure than those that don't.

The simultaneous relationship greatly increases the complexity of the problem. In Fig. 2b we see that the datasets *only after and simultaneous* and *only before and simultaneous* obtain similar scores. However, *only before and after* obtains a much better score than that of the standard specification.

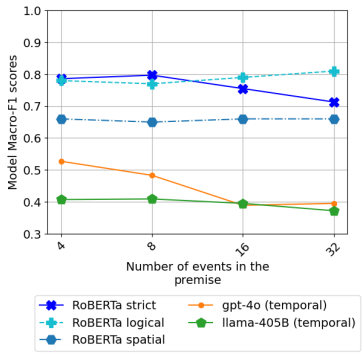
This result corroborates our intuition (cf. §3) that the formulation of the problem that included simultaneous relationships were computationally harder than those that didn't have a simultaneous relationship.

The formulation with *only before* is near ceiling Finetuned RoBERTa achieves 0.95 macro-F1 on the dataset which has simultaneous relationships removed and the *after* relationships to be inverted to be *before* relationships.

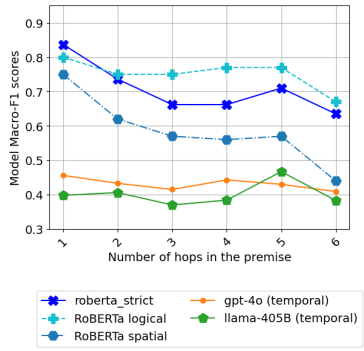
GPT is confused by the undefined class. RoBERTa is less so. Fig. 4a and Fig. 4b plots the confusion matrices for both GPT-4o and RoBERTa. Fig. 4a shows that GPT is prone to choosing the *True* or *False* labels while being averse to choosing the *Undefined* label. This shows a limitation in identifying when the premise contradicts itself. RoBERTa's confusion matrix (Fig. 4b) appears balanced in comparison. This means that an attention model is able to learn (by fine-tuning) generalizable patterns when given enough examples.

6 Related Work

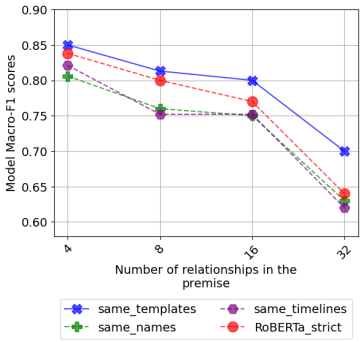
Recent studies on the performance of LLMs on temporal data (Xiong et al., 2024; Wang and Zhao, 2023) show that the problem is far from solved on test suites such as TimeBench (Chu et al., 2024). Both (Wang et al., 2024b) and (Xiong et al., 2024) attempt to improve performance by using temporal graphs. Xiong et al. (2024) show that converting a textual description into a temporal graph and performing chain-of-thought reasoning increases performance. However, the study does not systematically vary the complexity of the reasoning required, or diagnose the error patterns. We provide a systematic set of experiments and analysis in understanding the failure modes of the language models. We show that even the simplest descriptions of events are not fully understood by language models. Holtermann et al. (2025) find that models are able to satisfactorily perform temporal reasoning over timezones, they are not able to perform the same reasoning when asked to reason over both



(a) Ablation on event count performed with different models

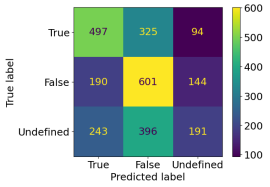


(b) Ablation study performed on the datasets partitioned by the number of logical hops in the premise (cf 2).

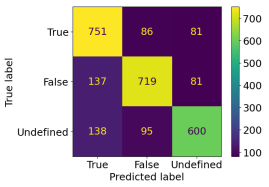


(c) Ablation performed with RoBERTa on datasets systematically designed to maintain common information between the train and test set.

Figure 3



(a) Confusion matrix on the labels for GPT-4o



(b) Confusion matrix on the labels for RoBERTa-large

Figure 4: Confusion Matrices

timezone and geographical locations.

Our work focuses on multi-hop reasoning on the specific domain of temporal graphs. Other projects (Lin et al., 2018; Misra et al., 2023) explore multi-hop reasoning on knowledge graphs. Wang et al. (2024a); Lin et al. (2024) study LLM capabilities in reasoning about graph datastructures that arise in other application areas. Their results resemble ours in showing deterioration of performance with increases in graph complexity. Works such as Han et al. (2024), explore the language models' ability to perform multi-hop reasoning on datasets that involve first-order logic (FOL) while Chen et al. (2020) explores multi-hop reasoning on textual information for the task of question answering.

Qi et al. (2024) explores how well language models are able to solve problems of different theoretical computational complexity. They show how performance degrades as the complexity increases. While the differences in complexities in their examples are more marked, the differences in complexity for temporal reasoning, as we have shown, are softer.

7 Conclusion

This paper presents EventHopNLI, a dataset that systematically identifies the error patterns of large language models on the task of temporal order. The dataset is designed to perform ablations across multiple factors (such as size of texts, temporal relationships, number of logical hops) while isolating specific linguistic features. We use the dataset to diagnose the error modes on examples of two paradigms of language models.

The results show us that there are limitations to language models' ability to traverse temporal graphs represented in texts. This prompts future research to investigate whether the use of a logical theorem solver (Pan et al., 2023; Olausson et al., 2023) can help obtain better results on the temporal ordering task.

Hopefully, this study will help everyday users of such models understand the expected limitations when they are applied to their specific data.

8 Limitations

As a result of our design goals, the dataset is limited to a specific set of linguistic features and temporal relations. It does not cover the further features described in §1 and §2. We leave it to future studies to create equally controlled diagnostic datasets that systematically include more of the linguistic features described in §2 such that the gap between natural data and this synthetic data is closed. We only evaluate general-purpose large language models, and do not evaluate approaches that explicitly construct temporal graphs or use scratchpads or Chain-of-Thought reasoning.

Acknowledgments

This work was funded by the Engineering and Physical Sciences Research Council (grant EP/T023333/1 awarded to University of Oxford)

References

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

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. 2024. TimeBench: A comprehensive evaluation of temporal reasoning abilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1228, Bangkok, Thailand. Association for Computational Linguistics.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. 2024. NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4092–4114, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Carolin Holtermann, Paul Röttger, and Anne Lauscher. 2025. Around the world in 24 hours: Probing llm knowledge of time and place. *arXiv preprint arXiv:2506.03984*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855.

Chokri Kooli. 2023. Chatbots in education and research: A critical examination of ethical implications and solutions. *Sustainability*, 15(7).

Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony G Cohn, and Janet B Pierrehumbert. 2024. Graph-enhanced large language models in asynchronous plan reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 30108–30134.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kanishka Misra, Cicero Nogueira dos Santos, and Siamak Shakeri. 2023. Triggering multi-hop reasoning for question answering in language models using soft prompts and random walks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 972–985, Toronto, Canada. Association for Computational Linguistics.

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

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the*

2023 Conference on Empirical Methods in Natural Language Processing, pages 5153–5176, Singapore. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Zhenting Qi, Hongyin Luo, Xuliang Huang, Zhuokai Zhao, Yibo Jiang, Xiangjun Fan, Himabindu Lakkaraju, and James Glass. 2024. Quantifying generalization complexity for large language models.

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

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024a. InstructGraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13492–13510, Bangkok, Thailand. Association for Computational Linguistics.

Jiapu Wang, Kai Sun, Linhao Luo, Wei Wei, Yongli Hu, Alan Wee-Chung Liew, Shirui Pan, and Baocai Yin. 2024b. Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 8384–8410. Curran Associates, Inc.

Yuqing Wang and Yun Zhao. 2023. Tram: Benchmarking temporal reasoning for large language models. *arXiv preprint arXiv:2310.00835*.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10452–10470.

A Analysis of the general complexity of temporal relationship extraction

In this section, we analyse a single paragraph from the TempEval dataset and illustrate the general complexity of the temporal relationship extraction task, both for models and for the human-annotation efforts. We identify and list below linguistic features or structures that provide the complexity.

Embedded under a speech-act verb: The event *sent* at (3) is embedded under the speech-act verb (1). The time of occurrence of the (3) is dependent on the time of occurrence of (1).

Expression of states indicate events that caused them: (5) indicates a current state of the expert, i.e., dead. This state indicates that a death had occurred sometime before. The *death* event remains implicit in the paragraph.

Vagueness: Compare temporal expressions (6) and event (14). The modifier *almost* introduces vagueness to the temporal expression. (14) is a temporal expression signifying a particular particular month.

Accomplishments/Achievements/Processes: The *recovery* at (7) is an accomplishment which culminates a period of searching which in itself is an implicit durative process. This implicit link will have to be processed by annotators and models.

Irrealis events: (7) is an infinitive subordinate of 'hope'. This places it on the irrealis axis. This means that though the event is reported it is unknown if the event definitely happened if it is in the past, or if is going to happen for sure if it is in the future. Many annotation schemes ask annotators to ignore such events since it is hard to provide a label for events that may take place. Similar problems exist for negative events.

Entity coreference resolution: (4) refers to a person (an entity) by their occupation while (8) refers to the entity by their name, without drawing an explicit connection between the two references. Entity coreference is itself a complex unsolved problem in NLP. Its complexity is inherited in the general task of temporal relationship classification.

Temporal markers: (9), (12) and (13) marks an explicit temporal relationship between an event and time. Obviously, having explicit temporal markers between events greatly simplifies the problem as compared to using implicit knowledge; however, an aim of this paper is to stress-test models' understanding given materials with only these explicit markers.

The top commander of a Cambodian resistance force said[1] Thursday[2] he has sent[3] a team to recover the remains of a British mine removal expert[4] kidnapped and presumed killed[5] by Khmer Rouge guerrillas almost two years ago[6]. Gen. Nhek Bunchhay ... said in an interview with The Associated Press at his hilltop headquarters that he hopes to recover[7] the remains of Christopher Howes[8] within[9] the next two weeks. Howes had been working[10] for the Britain-based Mines Advisory Group when[11] he was abducted[12] with his Cambodian interpreter Houn Hourth in[13] March 1996[14].

Table 2: An abridged example of text and events from the TempEval dataset. A popular dataset for evaluating temporal relationship extraction. This paragraph has been taken from article APW19980219.0476.tml. Each of the linguistic features and a discussion for them is presented in §2.

General knowledge and commonsense: (12) *working* can only happen before (7) because a person can only be working if they are alive. This is an example of general knowledge and commonsense being applied.

Therefore the task of temporal relationship extraction involves the application of multiple linguistic faculty. These features interact deeply with each other further complicating the task.

Algorithm 1 Find if event_1 overlaps event_2: Check if ep^1_{start} before ep^2_{start} and ep^1_{end} is after ep^2_{start}. Note that this assumes that e1 is before e2. The function will have to be called twice with swapped parameters to check all conditions of overlap.

```
function DOES_OVERLAP_FORWARDS(e1, e2)
    s1_s2 ← is_ep1_before_ep2(e1.sp, e2.sp)
    s2_s1 ← is_ep1_before_ep2(e2.sp, e1.sp)
    e1_s1 ← is_ep1_before_ep2(e1.ep, e1.sp)
    s2_e1 ← is_ep1_before_ep2(e2.sp, e1.ep)
    e2_s2 ← is_ep1_before_ep2(e2.ep, e2.sp)
    check ← (NOT e1_s1) AND (NOT e2_s2) AND ((NOT s2_s1) OR s1_s2) AND s2_e1
return check

end function
```

Algorithm 2 Find if event_point_1 occurs before but not simultaneous to event_point_2: Follow the relations and adding events to the queue if they happen after or simultaneous to those already in the queue

```
function IS_EP1_BEFORE_EP2(ep1, ep2)
    eps ← [(ep1, True)]
    seen ← set()
    while eps.length > 0 do
        ep, is_sim ← eps.pop()
        if ep in seen then
            continue
        end if
        seen.add(ep)
        for rel in ep.rels do
            if rel.reltype == 'after' AND ep == rel.ep2 then
                eps.append((rel.ep1, False))
            end if
            if rel.reltype() == 'before' AND ep == rel.ep1 then
                eps.append((rel.ep2(), False))
            end if
            if rel.reltype == 'simultaneous' then
                eps.append((rel.other_point(ep), is_sim))
            end if
            if (ep2, False) in eps then return True
            end if
        end for
    end while
     return False
end function
```

Algorithm 3 Find if the relationship between event_1 and event_2 cannot be determined because there is contradictory evidence. Check if there is contradictory evidence between all pairs of extremities

```
function IS_CONTRADICTORY_EVENT_PAIR(e1, e2)
    s1_s2 ← is_contra_eps(e1.sp, e2.sp)
    s1_e1 ← is_contra_eps(e1.sp, e1.ep)
    s1_e2 ← is_contra_eps(e1.sp, e2.ep)
    s2_e2 ← is_contra_eps(e2.sp, e2.ep)
    check ← s1_s2 or s1_e1 or s1_e2 or s2_e2
return check

end function
```

Algorithm 4 Find if event_point_1 and event_point_2 have contradictory relationships: Check if both ep1 is before and after ep2 as long as they are not simultaneous. If they are of the same event then make sure the end is not before start

```
function IS_CONTRADICTORY_EVENT_POINTS(ep1, ep2)
    same_event_check ← (ep2.event == ep2.event AND (ep1.event.stp == ep1) AND
            (ep2.event.enp == ep2)
    check_forwards ← is_ep1_before_ep2(ep1, ep2)
    check_backwards ← is_ep1_before_ep2(ep2, ep1)
    check_simultaneous ← is_simul_eps(ep1, ep2)
    check ← check_forwards AND check_backwards AND NOT check_simultaneous and
            NOT (same_event_check and check_backwards))
     return check

end function
```

Algorithm 5 Find if event_1 overlaps with event_2: Check if the events are not contradictory and if they overlap.

```
function IS_OVERLAP_EVENTS(e1, e2)
    is_contradictory ← is_contradictory_event_pair(e1, e2)
    e1_e2 ← self.does_overlap_forwards(e1, e2)
    e2_e1 ← self.does_overlap_forwards(e2, e1)
    check ← NOT is_contradictory AND (e1_e2 OR e2_e1)
     return check
end function
```

Algorithm 6 Check if ep1 and ep2 are simultaneous. Accumulate events in a queue by adding event points to the queue that are simultaneous to those already in the queue.

```
function IS_SIMUL_EPS(ep1, ep2)
    eps ← [event_point1]
    seen ← set()
    while eps.length > 0 do
        ep = eps.pop()
        seen.add(ep)
        for rel in ep.rels do
            if rel.reltype == 'simultaneous' then
                other_point ← rel.other_point(ep)
                if other_point NOT in seen then eps.append(other_point)
                end if
            end if
            if ep2 in eps then return True
            end if
        end for
    end whilereturn False
end function
```

Algorithm 7 Find if event_1 occurs strictly before event_point_2: check if ep^1_{start} and e^1_{end} are both before ep^2_{start} and e^1_{end} is not simultaneous to e^1_{end}

```
function IS_STRICTLY_BEFORE(e1, e2)
    s1_s2 ← is_ep1_before_ep2(e1.startp, e2.startp)
    e1_s2 ← is_ep1_before_ep2(e1.endp, e2.startp)
    e1_s2_is_simul ← is_simul_eps(e1.endp, e2.startp)
    check ← s1_s2 AND (e1_s2 OR e1_s2_is_simul)
     return check
end function
```

[INST] «SYS»
The premise is a set of battles and their temporal relationships
The hypothesis is a claim of the temporal relationship between two battles.

There are three answer choices:
1) True: The hypothesis is true given the premise
2) False: The hypothesis is False given the premise
3) Undefined: There is logically contradictory evidence in the premise regarding the events in the hypothesis. So no claim can be made.

The first five are examples with the labels provided.

Your task is to predict the label for the given examples. Do not provide reasoning and provide in the format of 'answer: index: label'.

Examples: <examples>

«/SYS»

Provide the labels for the following sentences in the format of 'answer: index: label'.

<uid> premise: <premise>
hypothesis: <hypothesis>

[INST]

Table 3: The baseline prompt used for Llama. The tokens in <> are replaced by actual values from the dataset.

Western edge of Stormforge is located to the east of eastern edge of Bloodmoon Keep. Eastern edge of Stormforge is located to the east of eastern edge of Bloodmoon Keep. Western edge of Sunfire Canyon is located to the west of western edge of Bloodmoon Keep. Eastern edge of Frostfang Pass is located to the west of western edge of Ravenloft. Eastern edge of Bloodmoon Keep is located to the west of eastern edge of Ravenloft. Eastern edge of Frostfang Pass is located to the east of western edge of Sunfire Canyon. Western edge of Sunfire Canyon is on the same longitude as western edge of Frostfang Pass. Eastern edge of Bloodmoon Keep is on the same longitude as western edge of Ravenloft. Western edge of Stormforge is located to the east of eastern edge of Ravenloft.

Table 4: Example of the NLI data in the spatial domain.