LTLBench: Towards Benchmarks for Evaluating Temporal Logic Reasoning in Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Temporal reasoning (TR) is a critical component of artificial intelligence, encompassing understanding and processing temporal information and relationships between events. To discover and study the TR ability in Large Language Models (LLMs), various datasets have been constructed in different ways for evaluating various aspects of TR ability. Our work proposes a novel approach to design and develop a pipeline for constructing datasets to evaluate the TR ability of LLMs by leveraging random directed graph generation, LTL formula, and the NuSMV model checker. Based on the pipeline, we have also constructed a dataset as a benchmark, namely LTLBench, consisting of 2,000 TR challenges and evaluated six LLMs with it. Furthermore, we have conducted additional experiments to discover the impact of increasing the number of events and formula operators on the complexity of TR problems and the performance of LLMs. We have demonstrated that although LLMs exhibit some promise in handling TR challenges, they still struggle with complex TR. We expect this work can offer insights into TR ability in LLMs while also providing a valuable tool for future TR evaluations.[1]

1 Introduction

Temporal reasoning (TR) is a fundamental and critical aspect of artificial intelligence, that encompasses understanding, processing, and reasoning about the temporal information and relationships between events, which is essential for handling and solving problems in various scenarios Shoham & Goyal (1988); Chittaro & Montanari (2000); Vila (1994). Recently, Large Language Models (LLMs) have demonstrated and shown promise and emergence of various reasoning abilities, including but not limited to mathematical reasoning Gaur & Saunshi (2023); Wei et al. (2022); Imani et al. (2023), commonsense reasoning Bian et al. (2024); Wei et al. (2022), and theory of mind reasoning Strachan et al. (2024); Kosinski (2023); Tang & Belle (2024a;b). Nevertheless, there still remains a lack of definitive consensus regarding the emergence, performance, and robustness of their ability to tackle TR challenges. Several studies have demonstrated that although LLMs have shown some promise with TR ability, they still struggle with TR and there is a substantial gap on the performance of handling and solving TR challenges between the state-of-the-art LLMs and humans Chu et al. (2023); Wang & Zhao (2024); Beniwal. Furthermore, since TR ability is crucial for handling and processing temporal information and relationships between temporal events and TR problems are ubiquitous among many activities Vila (1994), for LLMs, in their frequently mentioned and utilized scenario, question and answering, correctly comprehending temporal information and handling temporal tasks is necessary and warranted to provide efficient and helpful responses. In addition, LLMs are also discussed and used in various ways such as embodied agent Liu et al. (2023b); Wang et al. (2023); Liu et al. (2023a) for which planning and decision making unavoidably intertwine with TR. Therefore, it is necessary and important to study and discover the TR ability of LLMs.

In order to discover and evaluate the TR ability of LLMs, various benchmarks have been developed in previous studies using various dataset construction approaches, aiming at evaluating the TR ability of LLMs from different perspectives and at different levels of complexity. For example, in Xiong et al. (2024), they utilize

[1]The source code of the study is open source and available at `https://anonymous.4open.science/r/LTLBench-DF3E`.

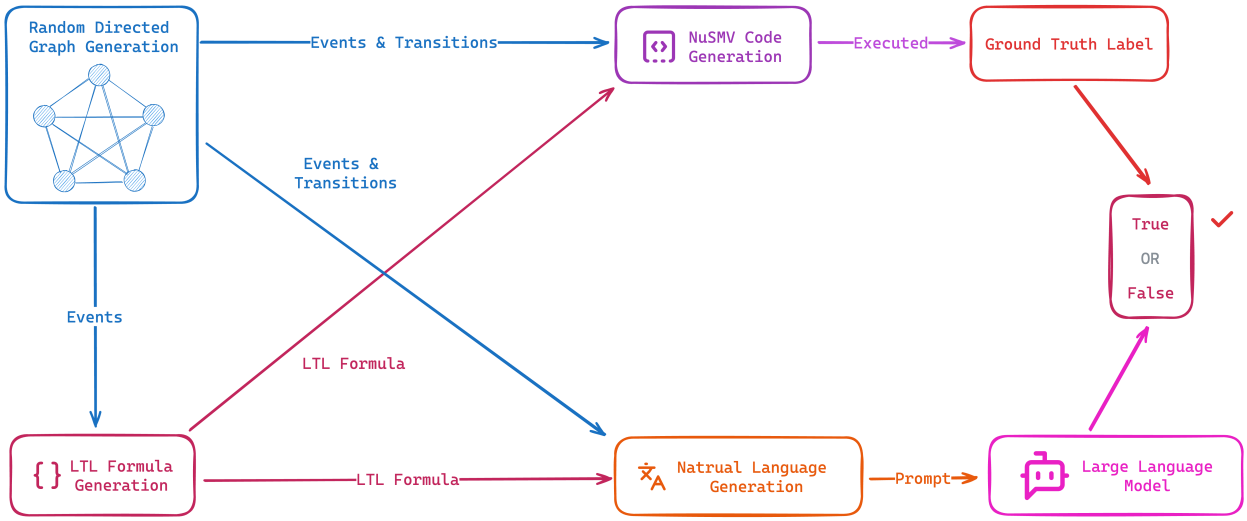


Figure 1: The overview of the TR problem generation pipeline.

GPT-3.5 to generate graph-based TR problems based on the temporal knowledge graph YAGO11k Dasgupta et al. (2018). Additionally, in Fatemi et al., they employ random graph generation as a foundation and preparation to form rule-based and different types of temporal facts and questions. In order to further discover the TR ability of LLMs and push the boundaries of evaluation of TR ability, we proposed and designed a novel pipeline for TR dataset construction which allows for the controllable and scalable generation of TR problems at any level of complexity and size. Based on it, we have also generated and constructed a dataset consisting of 2,000 TR problems as a benchmark, namely LTLBench.

Inspired by Fatemi et al., although our approach also involves random graph generation as the prerequisite and preparation for subsequent problem generations, the methodology differs a lot from it in terms of the usage of generated graphs and subsequent problem generations. In our approach, a generated TR problem mainly consists of a context that depicts the situation of a TR problem and a hypothesis requiring LLMs to judge its validity against the given context of the problem. In addition, the core components in the process of dataset generation involve a randomly generated directed graph which forms the preparation for subsequent problem generations, a random linear temporal logic (LTL) formula that serves as the hypothesis regarding a given problem context, and the NuSMV model checker Cimatti et al. (2002) which allows for running the code which represents the events, the transitions between events, and also the LTL formula to provide the ground truth label for the TR problem.

It is worth noting that although complex LTL is always discussed and used in the context of formal and program verification, basic and reasonably complex LTL tasks are ubiquitous in many daily tasks. For example, if people are out of milk, they will eventually buy it, which can be formalized in an LTL formula as $G(\text{OutOfMilk} \rightarrow F(\text{BuyMilk}))$, and also if the traffic light is green, it will then turn to yellow, which can be formalized as $G(\text{Green} \rightarrow X(\text{Yellow})$.[2] Therefore, LLMs are supposed to handle reasonably complex LTL problems in order to provide efficient and accurate responses. Therefore, we propose to use LTL, a subset of temporal logic, to form reasonably complex hypotheses of TR problems.

In addition, as shown in Figure 1, during the generation process for a TR problem, we first generate a random directed graph. Then, we adopted and slightly modified the LTL formulas generation algorithm designed by Zhu (2021) to generate an LTL formula based on the events given in the graph. Subsequently, we convert the information of events given in the graph and the LTL formula into NuSMV Cimatti et al. (2002) code and execute the code to obtain the ground truth label of the TR problem. Finally, the information of events and the LTL formula are converted into the TR problem in the form of natural language.

[2]Refer to Kröger & Merz (2008) and Goranko & Rumberg (2024) for more details on LTL syntax and semantics.

Furthermore, in order to intensively and comprehensively evaluate LLMs, we selected six models of which three are the models with large parameter sizes and three are the models with small parameter sizes, and evaluated them on LTLBench consisting of 2,000 generated TR challenges. We demonstrated that although LLMs have shown promise in handling TR challenges, they still struggle with complex TR. In addition, we have also taken additional experiments to discover how the number of events and operators may affect the complexity of TR problems and the performance of LLMs. The key contributions of our study are outlined as follows:

1. We have designed and developed a novel pipeline based on the random directed graph generation, LTL formula, and the NuSMV model checker for TR problem generation and dataset construction. Based on it, we have also generated and constructed a dataset, LTLBench, as a benchmark for evaluating the TR ability of LLMs. The data generation process in the pipeline is controllable and scalable, meaning it enables and allows for generating TR problems at any level of complexity and size;

2. We have also taken intensive and comprehensive experiments on six selected models for which three are models with large parameter sizes and three are in small parameter sizes and demonstrated that they still face significant challenges when tackling complex TR problems.

2 Related Work

2.1 TR in LLMs

Temporal reasoning has recently obtained substantial attention and study Xiong et al. (2024); Fatemi et al.; Beniwal; Chu et al. (2023); Hu et al.; Liu et al.; Vashishtha et al. (2020). In the work Beniwal, they point out the deficiencies of LLMs in terms of their ability to understand and handle temporal information and reasoning. Additionally, in Xiong et al. (2024), they introduce and propose a framework, TG-LLM, to improve the performance of LLMs in tackling TR tasks. Nevertheless, the evaluation and enhancement of TR ability in LLMs are still in progress and need more effort and study.

2.2 TR Benchmarks

To evaluate TR ability of LLMs, various TR datasets and benchmarks have been developed and constructed in different approaches for evaluating different aspects of TR ability of LLMs at varying levels of complexity Fatemi et al.; Wang & Zhao (2024); Xiong et al. (2024); Beniwal; Qin et al. (2021); Tan et al. (2023); Virgo et al.. For instance, in Xiong et al. (2024), a TR dataset is constructed by leveraging a large temporal knowledge graph, YAGO11k Dasgupta et al. (2018), and utilizing GPT-3.5 and rule-based Python script to generate TR challenges. However, in the work Fatemi et al., they leverage graph generation and use rule-based methods without introducing LLMs to generate TR tasks, highlighting and focusing on temporal semantics and arithmetic reasoning. Additionally, in Wang & Zhao (2024), they introduce a TR dataset consisting of various temporal aspects such as order, arithmetic, frequency, and duration. Our work introduces a novel pipeline for generating TR problems based on graph generation, LTL formula, and the NuSMV model checker, allowing for the control of the complexity and size of problems.

3 TR Problem Generation Pipeline

The pipeline to generate a single TR problem consists of four stages: (1) Random Directed Graph Generation, (2) LTL Formula Generation, (3) NuSMV Code Generation, and (4) Natural Language Generation. Figure 1 demonstrates the overview of the process for a TR problem generation.

3.1 Random Directed Graph Generation

During this stage, a directed graph is randomly generated with a given number of events n where $n > 1$ to ensure the formation of transitions between events.

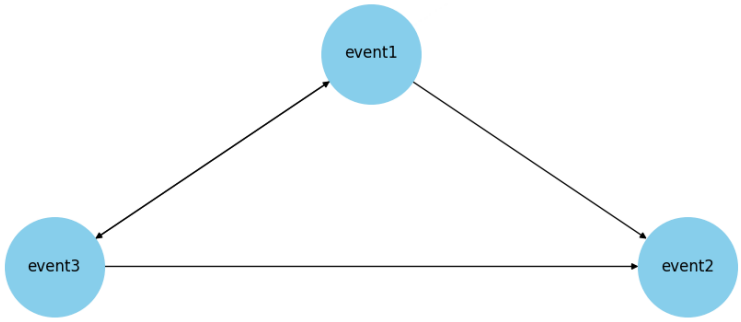


Figure 2: An example of a generated random directed graph.

In this graph, each $node_i$ represents an individual $event_i$, and each $edge_j^i$ is a directed edge pointing from $node_i$ to another $node_j$, which forms the relationships and transitions between events, indicating that $event_j$ represented by $node_j$ can occur after the $event_i$ represented by $node_i$. It is important to note that each $node_i$ within the graph can have multiple outgoing edges, signifying that several subsequent events can follow $event_i$, as well as multiple incoming edges, indicating that $event_i$ can be preceded by several other events.

As an example illustrated in Figure 2, given $n = 3$, three events are generated: $event_1$, $event_2$, and $event_3$. The case that $event_1$ points to $event_2$ indicates that $event_2$ can happen after $event_1$. The case that $event_1$ points to $event_3$ and $event_3$ also points to $event_1$ means that $event_1$ can happen after $event_3$ and $event_3$ can happen after $event_1$. In addition, $event_1$ not only points to $event_2$ but also $event_3$, indicating that both $event_2$ and $event_3$ can happen after $event_1$.

The generation of the random directed graph during this stage serves as the foundation and preparation for generating the LTL formula, NuSMV code, and TR problem represented in natural language by providing the information of events and also the transitions between events.

3.2 LTL Formula Generation

Based on the events generated in the graph, we employed and slightly modified the algorithm designed by Zhu (2021) to generate an LTL formula[3] with a given number of operators m where $m > 0$. The LTL operators include unary and binary operators. Unary operators, for example, include but are not limited to X which indicates that for a given event ϕ, $X\phi$ denotes that the event ϕ will occur at the next moment, and F for which $F\phi$ means that event ϕ will eventually occur at some point in the future. Binary operators include but are not limited to & representing logical AND and | representing logical OR. The given number of operators refers to the number of unary and binary LTL operators contained in an LTL formula. The algorithm for generating LTL formulas is detailed in Algorithm 1.

An example of a generated LTL formula is also provided and shown in Listing 1. The formula means that if $event_1$ happens, it will be globally true that, at some point in the future, $event_2$ will eventually happen.

```
(event1 -> (G (F event2)))
```

Listing 1: An example of a generated LTL formula.

The generation of the LTL formula aims to be the preparation for generating the LTLSPEC part of the NuSMV code and also to generate the hypothesis part of the TR problem represented in natural language.

[3]Note: only one LTL formula is generated for each given graph from the previous stage, meaning the *formulaCount* parameter in Algorithm 1 is always 1.

Algorithm 1 Generate LTL Formulas

procedure GenerateLTLFormulas($states, formulaLength, formulaCount$)
Input: $states$ - an array of atomic states representing events
Input: $formulaLength$ - the number of operators
Input: $formulaCount$ - the number of formulas to generate
Output: an array of LTL formulas
  $unaryOperators \leftarrow [X, G, F, !]$
  $binaryOperators \leftarrow [\&, |, \rightarrow]$
  $operators \leftarrow unaryOperators + binaryOperators$
  $B \leftarrow$ [[ ] of size $formulaLength + 1$]
  $B[0] \leftarrow states$
  $formulas \leftarrow$ [ ]
  for $i \leftarrow 1$ **to** $formulaCount$ **do**
    for $j \leftarrow 1$ **to** $formulaLength$ **do**
      $x \leftarrow$ randomly choose an operator from $operators$
      if $x \in unaryOperators$ **then**
        $y \leftarrow$ randomly sample a formula from $B[j-1]$
        $newFormula \leftarrow [x, y]$
      else
        $s \leftarrow$ randomly sample a integer from $[0, j)$
        $y1 \leftarrow$ randomly sample a formula from $B[s]$
        $y2 \leftarrow$ randomly sample a formula from $B[j-1-s]$
        $newFormula \leftarrow [y1, x, y2]$
      end if
      append $newFormula$ to $B[j]$
    end for
    $formula \leftarrow B[formulaLength][-1]$
    append $formula$ to $formulas$
  end for
  return $formulas$
end procedure

3.3 NuSMV Code Generation

Given the information of events from the graph and the LTL formula, this stage aims at converting and representing them in NuSMV code. The generation of the NuSMV code is divided into two parts: (1) context generation and (2) LTLSPEC generation. The context describes the situation of the TR problem while LTLSPEC represents a hypothesis regarding the context.

For the context generation, it includes event definitions, initial event setup, and event transitions setup. Based on the generated graph, the events and their transitions are converted into the context part of the NuSMV code while an initial event is set up randomly and also converted into the NuSMV code. As shown in Listing 2, Lines 2-3 define the three events, Line 5 sets up the initial event which is $event_3$, and Lines 6-12 construct the transitions of events in which, for instance, Line 7 shows that $event_2$ can follow $event_1$. In addition, Line 11 indicates that $event_2$ remains to itself and after $event_2$, no other events can happen, if there is no other transition specified from $event_2$ to $event_i$ where $i \neq 2$.

For LTLSPEC generation, we convert the generated LTL formula into code that complies with NuSMV, as illustrated at Line 13 in Listing 2 which represents the LTL formula shown in Listing 1.

The generation of the NuSMV code aims at providing the ground truth label for the TR problem. The generated code will be executed by the NuSMV model checker to give the ground truth label during the generation process.

```
MODULE main
VAR
    state : {event1, event2, event3};
ASSIGN
    init(state) := event3;
    next(state) := case
        state = event1 : event2;
        state = event1 : event3;
        state = event3 : event1;
        state = event3 : event2;
        state = event2 : event2;
    esac;
LTLSPEC ((state=event1) -> (G (F (state=event2))))
```

Listing 2: An example of NuSMV code.

```
1  === Context ===
2
3  Initially, event3 happened. After event1, event2 can happen. After event1, event3 can happen
       ↪ . After event2, no other events can happen. After event3, event1 can happen. After
       ↪ event3, event2 can happen.
4
5  === Hypothesis ===
6
7  C1: Event2 will happen eventually.
8  C2: C1 will always be true at any future time.
9  C3: That event1 happens implies that C2 holds.
10
11 C3 is True or False? Answer with "True" or "False" directly:
```

Listing 3: An example of TR problem in the form of natural language.

3.4 Natural Language Generation

During this stage, the events information in the graph and the LTL formula will be converted and represented in natural language. Similar to the NuSMV code generation, this stage can also be divided into two parts: (1) context generation and (2) hypothesis generation. The context describes the situation of the problem while the hypothesis is what the LLMs need to judge for validity against the given situation.

For context generation, based on the generated graph and the initial event, we convert the information of events and their transitions into natural language. As shown at Lines 1-4 in Listing 3, the initial event and events transitions are converted and described in natural language.

For hypothesis generation, we transform the generated LTL formula into natural language as shown at Lines 5-10 in Listing 3. Additionally, Line 11 is used to prompt LLMs to judge the validity of the hypothesis, requiring an answer in "True" or "False" format.

The TR problem represented in natural language generated in this stage is the core and final product of the generation process, which will be used to evaluate the TR ability of LLMs.

4 Experiment Settings

To evaluate the TR ability of LLMs, based on the pipeline, we generated and constructed a dataset, LTL-Bench, consisting of 2,000 problems. Each problem features the number of events $n = 3$ and the number of formula operators $m = 3$. Additionally, to explore the impact of changes in the number of formula operators, we conducted evaluations on newly generated problems with the fixed number of events $n = 2$ while varying the number of formula operators for which $m \in \{1, 2, 3, 4, 5, 7, 9\}$. For each (n, m_i), such as $(2, 1)$ indicating that the number of events is 2 and the number of operators is 1, there are 300 problems generated as a dataset for evaluation. Similarly, we fixed the number of formula operators $m = 2$ and varied the number of

Table 1: The metrics of LLMs on LTLBench

Models	Accuracy	F1	AUC
gpt-3.5-turbo	0.56	0.55	0.56
llama3:70b-instruct	0.59	**0.59**	0.59
qwen:72b-chat	**0.60**	**0.59**	**0.60**
gemma:7b-instruct	0.55	0.53	0.55
qwen:7b-chat	0.54	0.54	0.54
mistral:7b-instruct	0.54	0.50	0.54

events for which $n \in \{2, 3, 4, 5, 7, 9\}$ to examine the effects of changing the number of events while keeping the number of operators constant. In addition, for all generated datasets, their distributions of ground truth labels are meticulously balanced, meaning half of the problems are labeled as *True* while half of the problems are labeled as *False*.

Furthermore, for comprehensive evaluations, we selected six models. Three of them are in larger parameter sizes including GPT-3.5 Turbo noted as *gpt-3.5-turbo*, Llama 3 for which we choose its instruct model with 70 billion parameters noted as *llama3:70b-instruct*, and Qwen Bai et al. (2023) for which we utilize its chat model with 72 billion parameters noted as *qwen:72b-chat*. In addition, we also choose three models in smaller parameter sizes comprising Gemma Team et al. (2024) for which we use its instruct model with 7 billion parameters noted as *gemma:7b-instruct*, Mistral Jiang et al. (2023) for which we choose its instruct model with 7 billion parameters noted as *mistral:7b-instruct*, and Qwen Bai et al. (2023) for which we use its chat model with 7 billion parameters noted as *qwen:7b-chat*. This selection is intended for assessing the performance variability of handling TR challenges across models with different parameter sizes. In addition, the temperature hyperparameter for all selected models is set to 0 to keep the responses deterministic and also to ensure that the models consistently select the optimal next token.

5 Results and Analyses

5.1 Evaluation with LTLBench

We have evaluated the selected models on LTLBench which consists of 2,000 generated TR problems with the number of events $n = 3$ and the number of formula operators $m = 3$. The metrics consisting of Accuracy, F1, and AUC are shown in Table 1.

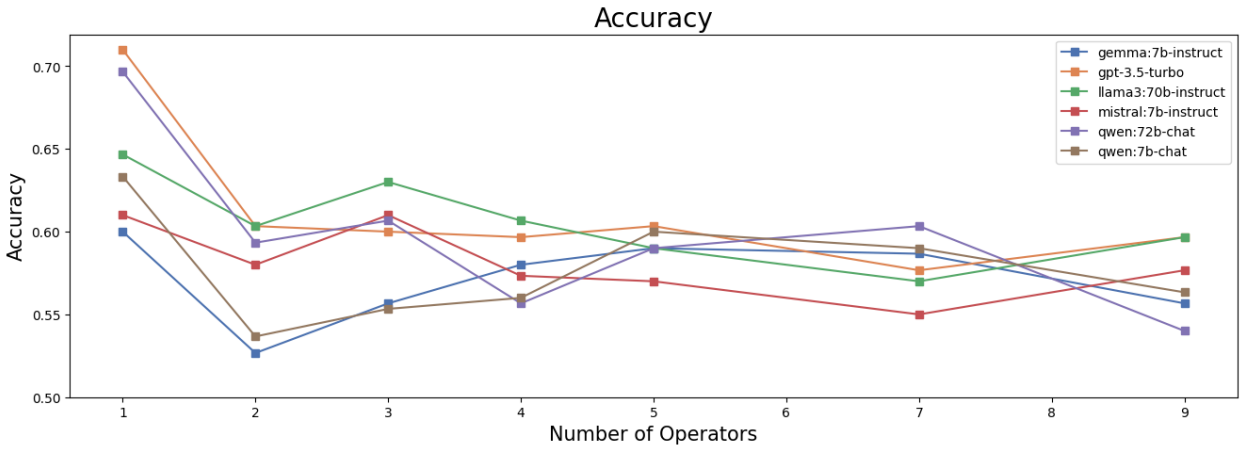
From the metrics, all Accuracy and AUC of LLMs are above 0.5. In addition, the model, *qwen:72b-chat*, demonstrates the highest Accuracy at 0.60, F1 at 0.59, and AUC at 0.60 while the *llama3:70b-instruct* also shows a similar performance. It may indicate that they show promise and ability to handle TR challenges. Nevertheless, the modest metrics signify that they still struggle with complex TR problems.

In addition, the averages of Accuracy, F1, and AUC for the models with large parameter sizes are 0.58, 0.58, and 0.58 respectively, while for the models with small parameter sizes, they are 0.54, 0.52, and 0.54 respectively. It unsurprisingly indicates that models with large parameter sizes may surpass the models with small parameter sizes when handling these complex TR challenges, although the difference between the performance is modest.

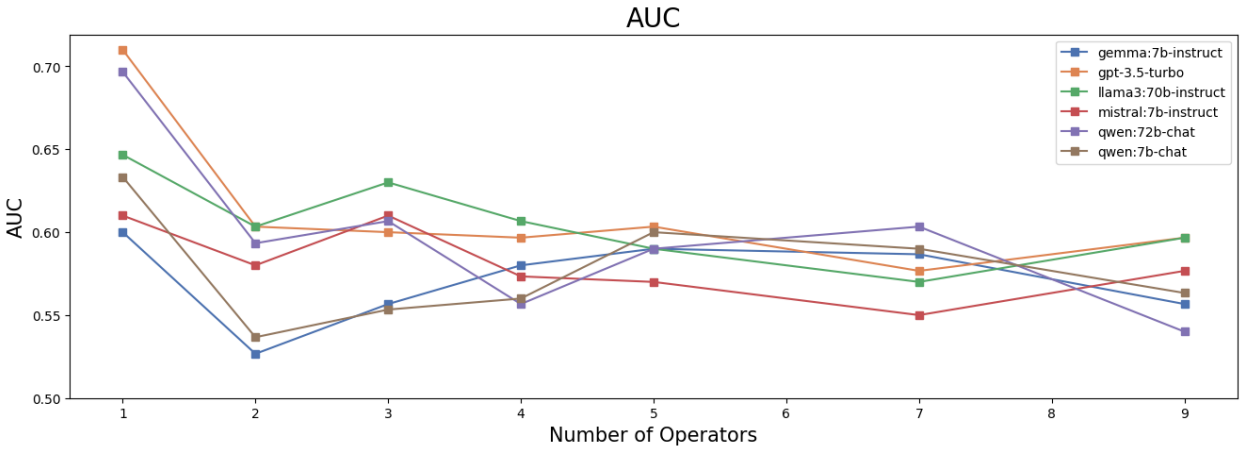
5.2 Impact of increasing m

The selected models are also evaluated on the additional constructed datasets, where the number of events is fixed to $n = 2$ while the number of formula operator m increases from 1 to 9, specifically $m \in \{1, 2, 3, 4, 5, 7, 9\}$. The performance in terms of Accuracy and AUC of the evaluated models is shown in Figure 3.

As shown in Figure 3a, the accuracy of models significantly drops from $m = 1$ to $m = 2$ regardless of the models with large parameter sizes or models with small parameter sizes. It is also the same case for



(a) Accuracy



(b) AUC

Figure 3: Accuracy and AUC when increasing the number of operators.

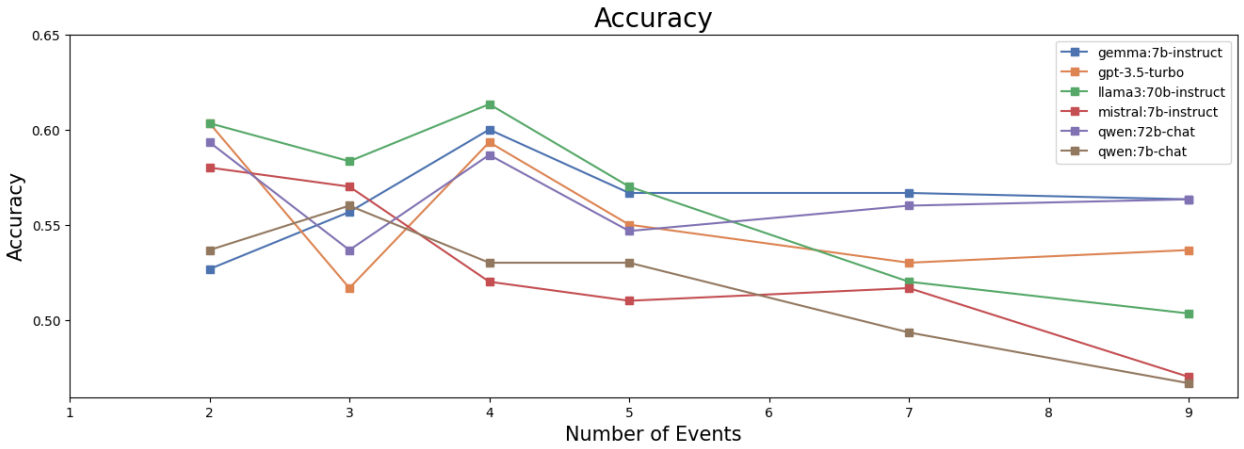
AUC as shown in Figure 3b. Although several models can have a good performance when $m = 1$ such as *gpt-3.5-turbo* and *qwen:72b-chat*, they suffer from TR tasks when m increases. It indicates that increasing the number of formula operators may significantly introduce more complexity of TR and the TR ability of LLMs is currently still lacking consistency and robustness when TR tasks become more complex.

Furthermore, it is obvious to note that models with large parameter sizes outperform all other models with small parameter sizes when $m = 1$. However, they show indifferent performance when m increases while the AUC decreases and approaches to 0.5, signifying that all models start random guessing.

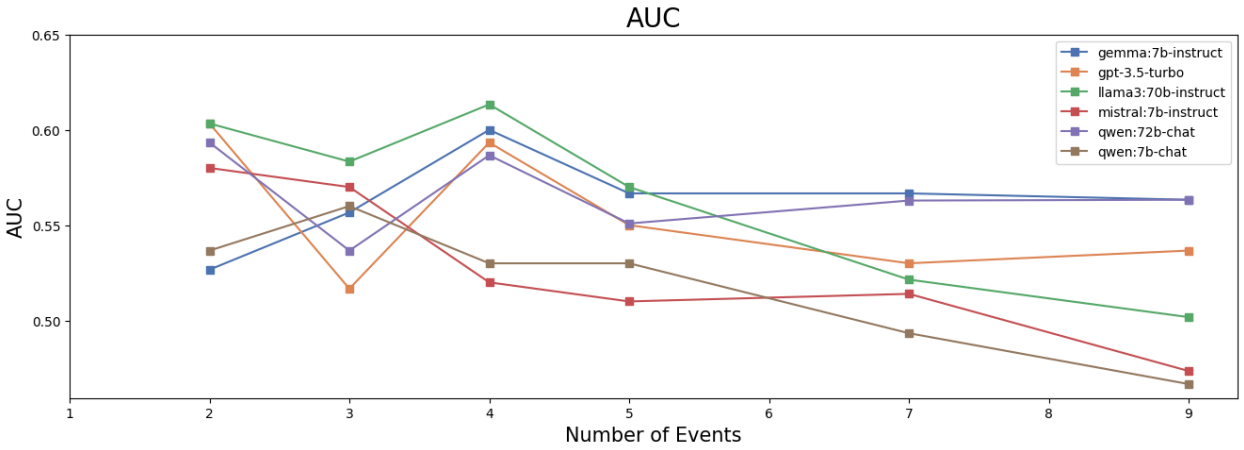
5.3 Impact of increasing n

In addition, the selected models are also evaluated with the number of formula operators fixed to $m = 2$ while the number of events n where $n > 1$ increases from 2 to 9, specifically $n \in \{2, 3, 4, 5, 7, 9\}$. The performance in terms of Accuracy and AUC of the evaluated models is shown in Figure 4.

As shown in Figure 4a and Figure 4b, the Accuracy and AUC of models show a trend to decrease while the number of events increases. Although it is not as obvious and significant as the case when n is fixed and m increases, it still indicates that increasing the number of events will also introduce a certain level of complexity. In addition, as the same case discussed in Section 5.2, as the number of events increases, models



(a) Accuracy



(b) AUC

Figure 4: Accuracy and AUC when increasing the number of events.

with large parameter sizes show indifferent performance to the models with small parameter sizes since their AUC approaches 0.5, indicating random guessing.

6 Implications

We designed a pipeline for constructing TR datasets to evaluate the TR ability of LLMs. The pipeline is controllable and scalable to be able to introduce any level of complexity of TR by modifying the number of events and the number of formula operators. Future work can leverage this pipeline to generate TR problems at a certain level of complexity depending on specific needs.

In addition, since TR is crucial in many scenarios, we hope this work can not only offer insights into the TR ability of LLMs but also provide a valuable tool to further evaluate the TR ability of future LLMs or other AI systems that are required to equip with a certain level of TR ability, thus, to enable confident deployment and application.

7 Limitations and Future Work

Considering the difficulty of generated TR problems, we only included a subset of LTL operators in the TR problem generation pipeline but excluded several LTL operators such as U (Until) and R (Release). However, adding and incorporating more LTL operators should be easy and straightforward to enable future work to evaluate the more complex TR ability of LLMs.

In addition, although we have intensively and comprehensively evaluated six models, due to the unavoidable heavy computation costs, we did not run the evaluation on GPT-4 and other LLMs exhaustively. Nevertheless, future work can easily leverage the pipeline and the constructed dateset, LTLBench, in this work to evaluate GPT-4 and other LLMs.

8 Conclusion

In this study, we designed a pipeline for generating and constructing the TR datasets based on the random graph generation, LTL formula, and the NuSMV model checker. Based on the pipeline, we have generated a dataset, LTLBench, consisting of 2,000 TR challenges and taken intensive and comprehensive evaluations on it with the six selected models in which three are the models with large parameter sizes while the other three models are in small parameter sizes. We have demonstrated that although LLMs show promise and emergence in handling the TR problem, they still struggle with it. We expect our work to offer insights into the TR ability of LLMs and to provide a valuable tool for TR evaluation while hoping it can pave the way for future more intelligent LLMs or AI systems requiring handling complex TR tasks.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. (arXiv:2309.16609), September 2023. URL `http://arxiv.org/abs/2309.16609`. arXiv:2309.16609 [cs].

Himanshu Beniwal. Remember this event that year? assessing temporal information and reasoning in large language models.

Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, Ben He, Shanshan Jiang, and Bin Dong. ChatGPT is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 3098–3110, Torino, Italia, May 2024. ELRA and ICCL. URL `https://aclanthology.org/2024.lrec-main.276`.

Luca Chittaro and Angelo Montanari. Temporal representation and reasoning in artificial intelligence: Issues and approaches. *Annals of Mathematics and Artificial Intelligence*, 28(1):47–106, 2000.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. Timebench: A comprehensive evaluation of temporal reasoning abilities in large language models. (arXiv:2311.17667), November 2023. URL `http://arxiv.org/abs/2311.17667`. arXiv:2311.17667 [cs].

A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark, July 2002. Springer.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2001–2011, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1225. URL https://aclanthology.org/D18-1225.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating llms on temporal reasoning.

Vedant Gaur and Nikunj Saunshi. Reasoning in large language models through symbolic math word problems. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 5889–5903, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.364. URL https://aclanthology.org/2023.findings-acl.364.

Valentin Goranko and Antje Rumberg. Temporal Logic. In Edward N. Zalta and Uri Nodelman (eds.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2024 edition, 2024.

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S Yu, and Zhijiang Guo. Do large language models know about facts?

Shima Imani, Liang Du, and Harsh Shrivastava. MathPrompter: Mathematical reasoning using large language models. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 37–42, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.4. URL https://aclanthology.org/2023.acl-industry.4.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. (arXiv:2310.06825), October 2023. URL http://arxiv.org/abs/2310.06825. arXiv:2310.06825 [cs].

Michal Kosinski. Evaluating large language models in theory of mind tasks. 2023. doi: 10.48550/ARXIV.2302.02083. URL https://arxiv.org/abs/2302.02083.

Fred Kröger and Stephan Merz. *Temporal logic and state systems*. Texts in theoretical computer science: An EATCS series. Springer, Berlin, 2008. ISBN 978-3-540-67401-6.

Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments.

Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. (arXiv:2302.11649), October 2023a. URL http://arxiv.org/abs/2302.11649. arXiv:2302.11649 [cs].

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. (arXiv:2308.03688), October 2023b. URL http://arxiv.org/abs/2308.03688. arXiv:2308.03688 [cs].

Lianhui Qin, Aditya Gupta, Shyam Upadhyay, Luheng He, Yejin Choi, and Manaal Faruqui. Timedial: Temporal commonsense reasoning in dialog. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7066–7076, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.549. URL https://aclanthology.org/2021.acl-long.549.

Yoav Shoham and Nita Goyal. Chapter 11 - temporal reasoning in artificial intelligence. In Howard E. Shrobe and the American Association for Artificial Intelligence (eds.), *Exploring Artificial Intelligence*, pp. 419–438. Morgan Kaufmann, 1988. ISBN 978-0-934613-67-5. doi: https://doi.org/10.1016/B978-0-934613-67-5.50015-0. URL https://www.sciencedirect.com/science/article/pii/B9780934613675500150.

James W. A. Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, Michael S. A. Graziano, and Cristina Becchio. Testing theory of mind in large language models and humans. *Nature Human Behaviour*, May 2024. ISSN 2397-3374. doi: 10.1038/s41562-024-01882-z. URL https://www.nature.com/articles/s41562-024-01882-z.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14820–14835, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.828. URL https://aclanthology.org/2023.acl-long.828.

Weizhi Tang and Vaishak Belle. Tom-lm: Delegating theory of mind reasoning to external symbolic executors in large language models. *arXiv preprint arXiv:2404.15515*, 2024a.

Weizhi Tang and Vaishak Belle. Zero, finite, and infinite belief history of theory of mind reasoning in large language models. *arXiv preprint arXiv:2406.04800*, 2024b.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L. Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu-hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology. (arXiv:2403.08295), April 2024. URL http://arxiv.org/abs/2403.08295. arXiv:2403.08295 [cs].

Siddharth Vashishtha, Adam Poliak, Yash Kumar Lal, Benjamin Van Durme, and Aaron Steven White. Temporal reasoning in natural language inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4070–4078, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.363. URL https://www.aclweb.org/anthology/2020.findings-emnlp.363.

Lluis Vila. A survey on temporal reasoning in artificial intelligence. *Ai Communications*, 7(1):4–28, 1994.

Felix Virgo, Fei Cheng, and Sadao Kurohashi. Improving event duration question answering by leveraging existing temporal information extraction data.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. (arXiv:2305.16291), October 2023. URL http://arxiv.org/abs/2305.16291. arXiv:2305.16291 [cs].

Yuqing Wang and Yun Zhao. Tram: Benchmarking temporal reasoning for large language models. (arXiv:2310.00835), May 2024. URL http://arxiv.org/abs/2310.00835. arXiv:2310.00835 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2022. doi: 10.48550/ARXIV.2201.11903. URL https://arxiv.org/abs/2201.11903.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. Large language models can learn temporal reasoning. (arXiv:2401.06853), June 2024. URL http://arxiv.org/abs/2401.06853. arXiv:2401.06853 [cs].

Weijun Zhu. Big data on linear temporal logic formulas. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 544–547, Chongqing, China, June 2021. IEEE. ISBN 978-1-72818-535-4. doi: 10.1109/IMCEC51613.2021.9482368. URL https://ieeexplore.ieee.org/document/9482368/.