QUACER-C: QUANTITATIVE CERTIFICATION OF KNOWLEDGE COMPREHENSION IN LLMS

Isha Chaudhary[1], **Vedaant V. Jain**[1] **& Gagandeep Singh**[1,2]
[1]University of Illinois Urbana-Champaign, USA
[2]VMware Research, USA
{isha4, vvjain3, ggnds}@illinois.edu

ABSTRACT

Large Language Models (LLMs) have demonstrated impressive performance on several benchmarks. However, traditional studies do not provide formal guarantees on the performance of LLMs. In this work, we propose a novel certification framework for LLM, QuaCer-C[1], wherein we formally certify the knowledge-comprehension capabilities of popular LLMs. Our certificates are quantitative — they consist of high-confidence, tight bounds on the probability that the target LLM gives the correct answer on any relevant knowledge comprehension prompt. We certify popular LLMs to give provable and generalizable bounds on their knowledge comprehension capabilities on the Wikidata knowledge graph.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated human-level performance for several real-world downstream tasks (Yang et al., 2023; Liang et al., 2023; Bommasani et al., 2022). An important use case for LLMs is knowledge comprehension (Lazaridou et al., 2022; Khattab et al., 2023), that is, they are often used to summarize long texts, respond to user queries based on the context, and serve as adaptive task decomposers for reasoning-based retrieval augmented generation tasks (Yao et al., 2023). Large context windows of millions of tokens in models like Gemini v1.5 (Gemini Team, 2024) reduce reliance on large knowledge corpora of RAG systems and parametric knowledge held by LLMs. Users increasingly provide extensive references during inference to guide responses. This makes analyzing the knowledge-comprehension and reasoning capabilities of popular LLMs crucial.

There have been several studies to benchmark the performance of LLMs for knowledge comprehension using multihop question-answering datasets (Liang et al., 2023; Chen et al., 2021; Yang et al., 2018; Wang et al., 2023; Trivedi et al., 2022; Tang & Yang, 2024). These datasets consist of questions that require several reasoning steps to obtain the correct answer. Thus, benchmarking knowledge comprehension involves analyzing whether the target LLM can combine multiple pieces of information in meaningful ways and reason its way to the correct answer in the prompt, without deviating or hallucinating. However, prior studies lack a formal analysis of the knowledge-comprehension capability of the LLM. While they can inform us of the high-level trends in the performance of popular LLMs, they do not guarantee the risk involved in using a given LLM for knowledge comprehension tasks. Such guarantees are crucial when deploying LLMs in large-scale knowledge-comprehension tasks in critical domains such as medicine or finance, as they can give more confidence about the trustworthiness of LLMs before deployment. Our work aims to bridge this gap by introducing a novel formal certification method, QuaCer-C, for obtaining certified bounds on the knowledge comprehension capability of LLMs.

Key challenges. We face the following challenges when developing a formal certification framework for LLMs. (1) We need formal expressions capturing the knowledge comprehension property, amenable to certification. Such expressions should capture the diverse prompts for the property that the LLM should correctly answer. (2) Giving provable guarantees on LLMs is a hard, open problem, due to the high number of parameters of the models, for which the traditional certifiers (Singh et al., 2019; Shi et al., 2020; Bonaert et al., 2021) would lose significant precision leading to inconclusive

[1]**Qua**ntitative **Cer**tification of **C**omprehension

analysis. Moreover, failure examples where the desirable property does not hold are fairly easy to construct for popular LLMs by appropriate prompt tuning (Xu et al., 2024; Vega et al., 2023), making binary certificates that indicate whether an LLM satisfies a specification pointless. The number of prompts over which we desire the target LLMs to correctly reason is also large, making enumeration-based benchmarking intractable to generate any formal guarantees.

Our approach. We first conceptualize the knowledge comprehension property as a novel formal specification using a knowledge graph. We develop quantitative properties to obtain measures for the knowledge-comprehension capability of the target LLM. We then propose a model-agnostic, black-box quantitative certification approach, QuaCer-C, which circumvents the issues that traditional approaches have with the number of parameters in LLMs and can even work for closed-source LLMs. QuaCer-C generates high-confidence bounds on the quantitative property using queries leveraging confidence intervals (Clopper & Pearson, 1934). While formal analysis has been conducted on the individual generations of LLMs in prior work (Quach et al., 2024), there is no analysis for the average-case risk of LLMs in knowledge comprehension, over large, real-world knowledge graphs. Our certificates contain provable bounds on the probability of getting correct responses for a large number of knowledge comprehension tasks defined as specifications over knowledge graphs.

Contributions. We make the following contributions:

1. We specify the knowledge comprehension property desirable from the LLM responses as a formal expression. Our specifications use popular knowledge graphs such as Wikidata5m (Wang et al., 2021) that are augmented with supporting information about each of their entities. The specifications represent a large set of knowledge comprehension prompts with their respective correct answers, expected from any target LLM.
2. We model certification in a target LLM as a probability estimation problem and leverage Clopper-Pearson confidence intervals to generate provable, high-confidence bounds on the quantitative property of interest. Our implementation and data are open-sourced at `https://github.com/uiuc-focal-lab/QuaCer-C`.
3. We generate the proposed certificates for the popular LLMs: Llama 7B and 13B, Vicuna 7B and 13B, Mistral-7B, and Gemini-Pro. We observe that as the number of model parameters increases, the knowledge comprehension capability of the LLM improves. On comparing the different model classes, we see Gemini performing significantly better than others.

Our work is the first step towards providing guarantees on the knowledge comprehension and reasoning capabilities of LLMs, to ameliorate the caution needed when using LLMs (Shanahan, 2023) in a systematic way. We anticipate it to go a long way in making LLMs faithful to traditional notions of reasoning, and hence more trustworthy for deployment in critical domains.

2 CERTIFYING KNOWLEDGE COMPREHENSION

Consider an LLM $\mathcal{L} : \chi \to \mathcal{Y}$ that takes prompts $p \in \chi$ as input and generates responses $r \in \mathcal{Y}$, where χ and $\mathcal{Y}$ are sets of all possible prompts to the LLM and the responses generated by it, respectively. We formally assess $\mathcal{L}$ for its capability to comprehend knowledge given in its prompt and answer a multi-hop query involving multiple reasoning steps based on the given knowledge in the prompt. We define the knowledge comprehension property of LLMs next.

2.1 FORMAL SPECIFICATION

The property is defined over a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ having nodes $\mathcal{V}$ that correspond to distinct entities (e.g., name of a country) that are augmented with textual information about them and directed edges $\mathcal{E}$ between pairs of nodes that are labeled by the relation between the nodes. There can be multiple names aka *aliases* to identify each entity and each label for edges. An example of such a knowledge graph is Wikidata5m (Wang et al., 2021) which consists of nodes for entities having Wikipedia pages, along with the abstracts from the respective pages. Two nodes (v_1, v_2) are connected by a labeled edge if there is a link in the supporting document for v_1 to that for v_2.

Definition 2.1. (Path in a Knowledge Graph). A path $\Pi = [v_1, v_2 \ldots, v_l]$ is an ordered collection of nodes in a given knowledge graph $\mathcal{G}$, where $l > 1$, such that $\forall i \in [l-1]. v_i \in \mathcal{V}$ and $(v_i, v_{i+1}) \in \mathcal{E}$.

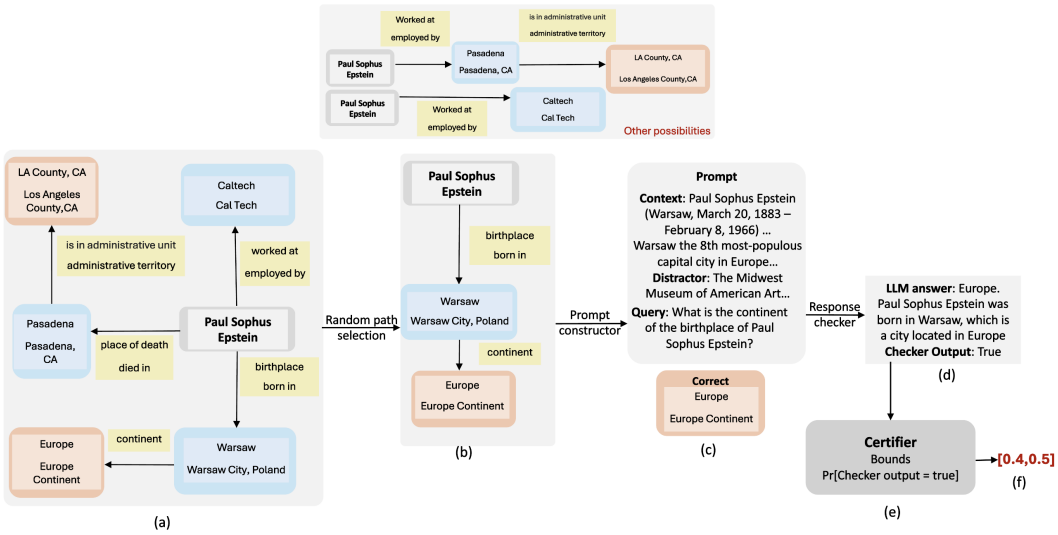

Figure 1: Overview of QuaCer-C. (a) A knowledge graph $\mathcal{G}$ pivoted on some node, in this case on 'Paul Sophus Epstein'. (b) A randomly chosen path originating at the pivot node from the various possibilities in $\mathcal{G}$. (c) A prompt created by our prompt constructor using the selected path and context from the Wikidata5m corpus of the entities (nodes) involved, along with a distractor context and the final query. (d) The target LLM's output to the prompt, validated using the response checker. (e) Certifier obtains bounds on the probability of correct response and we iterate till the range of the bounds falls below a threshold. (f) The final bounds contained in the certificate.

Next, we describe two primitive functions — *prompt constructor* and *response checker* — with which we specify the knowledge comprehension property. Let $\Pi_{\mathcal{G}}$ denote all paths in $\mathcal{G}$, a prompt constructor $\mathcal{Q}_{\mathcal{G}}$ takes a path (Definition 2.1) $\Pi \in \Pi_{\mathcal{G}}$ of finite but arbitrary length as input, and randomly generates a query-based prompt and the correct answer. Each prompt produced from a given path is a concatenation of some textual information about nodes and a query. The textual information consists of randomly organized supporting context for all the nodes in Π and other randomly selected *distractor texts* from nodes not in the path. The query in the prompt is on the entity corresponding to the tail of Π and it describes the rest of the path. The correct answer to the prompt is an alias of the tail of Π. The specific prompt constructor function is user-defined and can determine the difficulty level of the prompts with which the target LLM $\mathcal{L}$ is certified. A stochastic prompt constructor enables us to develop a general specification that takes into account the effect of common aliases and different ordering of information in the prompt on $\mathcal{L}$'s response. A response checker $\mathcal{C}_{\mathcal{L}}(p, r)$ is a boolean function that evaluates to true if $\mathcal{L}$'s response for prompt p matches any of the aliases of the given node r.

Our knowledge comprehension property requires that the target LLM $\mathcal{L}$ correctly answers queries contained in the prompts generated by the prompt constructor $\mathcal{Q}_{\mathcal{G}}$ on all paths in $\Pi_{\mathcal{G}}$, as identified by the response checker $\mathcal{C}$. We reformulate the property to make it quantitative by measuring the probability that the target LLM $\mathcal{L}$ correctly answers the query in a prompt over a randomly selected path from $\Pi_{\mathcal{G}}$ (Eq. 1), where $Dist(\Pi_{\mathcal{G}})$ is a user-defined probability distribution over all the paths $\Pi \in \Pi_{\mathcal{G}}$. A higher probability value indicates better knowledge comprehension capability of $\mathcal{L}$.

$$Pr_{\Pi \sim Dist(\Pi_{\mathcal{G}})} \mathcal{C}_{\mathcal{L}}(\mathcal{Q}_{\mathcal{G}}(\Pi)) = T \tag{1}$$

From a practical standpoint, longer paths can become semantically meaningless, and thus shorter path lengths are considered in popular multi-hop question-answering datasets such as (Yang et al., 2018; Chen et al., 2021; Trivedi et al., 2022). Thus, we upper-bound the lengths of paths certified, which we define as the number of nodes in the path, by a hyperparameter ρ. Let $\Pi_{\mathcal{G},\rho} \subseteq \Pi_{\mathcal{G}}$ denote all the paths, where each path has a maximum length ρ. Therefore, our knowledge comprehension property becomes Eq. 2 for any given maximum path length ρ.

$$Pr_{\Pi \sim Dist(\Pi_{\mathcal{G},\rho})} \mathcal{C}_{\mathcal{L}}(\mathcal{Q}_{\mathcal{G}}(\Pi)) = T \tag{2}$$

Global certification over the entire $\mathcal{G}$ involves a large number of queries and may not scale for real-world graphs having millions of nodes. Hence, we scope the probabilistic property for $\mathcal{L}$ (Eq. 2) into a local subgraph level property, wherein each subgraph $\mathcal{G}(v)$ is defined by a randomly selected *pivot* node v and consists of all paths Π_v originating from v having maximum length of ρ. Hence, the local probabilistic property becomes Eq. 3.

$$Pr_{\Pi \sim Dist(\Pi_v)} \mathcal{C}_{\mathcal{L}}(\mathcal{Q}_{\mathcal{G}}(\Pi)) = T \tag{3}$$

2.2 Certification method

Figure 1 gives an overview of our certification framework, QuaCer-C. QuaCer-C certifies the target LLM $\mathcal{L}$ by computing an interval $[p_l, p_u]$ containing the value of the property (Eq. 3) for a given pivot node v, prompt constructor $\mathcal{Q}_{\mathcal{G}}$, response checker $\mathcal{C}$, and distribution $Dist(\Pi_v)$ of paths in the subgraph of interest. We want tight intervals, i.e., a low value of $p_u - p_l$, for precise certification. For reliability, we construct intervals that bound the value of interest with high confidence. We determine bounds with higher confidence than the user-specified confidence level $1-\delta$, where δ is a small positive constant, using the Clopper-Pearson confidence intervals (Clopper & Pearson, 1934; Brown et al., 2001; Kurz et al., 2014; Collins, 2010) based on samples of $\mathcal{L}$'s responses. The bounds obtained are conservative, i.e., they are lower and upper bounds for the target property's value p with a confidence of at least $1-\delta$.

$$Pr[p_l \leq p \leq p_u] \geq 1 - \delta \tag{4}$$

We model the value p of the property in 3 as the probability of setting the underlying boolean random variable $\mathcal{R} = \mathcal{C}_{\mathcal{L}}(\mathcal{Q}_{\mathcal{G}}(\Pi))$ to true (success). We model $\mathcal{R}$ as a Bernoulli random variable with an unknown, constant probability of success, denoted by p. This assumption is based on the expectation that specifications, as defined on local subgraphs, will capture similar queries that elicit similar responses from $\mathcal{L}$ and, consequently, a consistent probability of success across different sampled prompts. To compute high-confidence tight bounds on p, we make n random observations of $\mathcal{R}$, out of which we obtain $k \leq n$ successes. With the n observations and $1-\delta$ confidence, we use the Clopper-Pearson confidence intervals to bound p with upper and lower bound p_u, p_l respectively. We make an adaptive number of observations n till the range between the bounds $p_u - p_l$ falls below a prespecified threshold α, which should be small for precise certificates. However, n increases on decreasing α, therefore α can be tuned according to the certification budget and desired precision.

3 Experiments

We generate the proposed certificate for popular LLMs and derive insights into their knowledge comprehension capability when compared with each other.

Experimental setup. We certify the 7B and 13B parameter Llama-2-chat models (Touvron et al., 2023) that have been instruction-tuned and safety-aligned, Vicuna 7B,13B v1.5 (16k context window size) (Chiang et al., 2023), Mistral 7B-Instruct-v0.2 (Jiang et al., 2023), and Gemini-Pro models. We use Wikidata5m (Wang et al., 2021) as our base knowledge graph after preprocessing (check Appendix B for details) from which we develop a test set of 50 specifications, each pivoted at a randomly selected node in Wikidata5m having a high number of possible paths, making enumerative certification (where all possibilities are tested for satisfaction of the specification, similar to traditional benchmarking) impractical. We select the largest maximum path length parameter $\rho = 5$, which could still form a sensible query. A path is sampled from our distribution over all paths in a subgraph by first uniformly randomly selecting a path length from $[1, \rho]$ and then uniformly randomly selecting a path having the chosen path length. We detail our prompt constructor prepended with the path sampler from the distribution of paths in a subgraph in Appendix A.1. We evaluate the responses of LLMs using string matching and another LLM (Mistral-7B-Instruct-v0.2) as our response checker function. The detailed design and evaluation of our response checker is presented in Appendix A.2. QuaCer-C generates certificates with confidence $1 - \delta = 95\%$ and certify till the range of the bounds falls below a threshold $\alpha = 0.1$. We also keep a maximum cap on the number of samples, which we fix as 500 samples, and report the bounds obtained if the threshold is reached. We conduct our experiments on 4 A100 GPUs hosting the open-source LLMs.

Certificates. QuaCer-C generates certificates providing tight, high-confidence lower and upper bounds on the probability of a correct LLM response to a random prompt obtained from the prompt

constructor applied on a given subgraph. We report the average value of the lower and upper bounds, over the test set of properties that QuaCer-C certifies for each LLM, in Table 1. We also report the average empirical probability (the ratio of correct responses to the total number of prompts, n, for each certificate), averaged over the test set.

We also study the robustness of the target LLMs in correctly answering random queries generated by just perturbing the aliases in a query corresponding to a path. The average alias robustness over all the paths considered in the certificates is $(60 \pm 5)\%$ for each target model.

Table 1: Certification results for different LLMs

Model	Avg. lower bound	Avg. upper bound	Avg. empirical value
Vicuna-7b-v1.5-16k	0.57 ± 0.08	0.67 ± 0.08	0.62 ± 0.08
Llama-7b-chat	0.61 ± 0.07	0.71 ± 0.07	0.66 ± 0.07
Vicuna-13b-v1.5-16k	0.63 ± 0.08	0.73 ± 0.08	0.68 ± 0.08
Llama-13b-chat	0.69 ± 0.06	0.79 ± 0.06	0.74 ± 0.07
Mistral-7b-Instruct-v0.2	0.64 ± 0.07	0.74 ± 0.07	0.69 ± 0.07
Gemini-Pro	0.75 ± 0.11	0.84 ± 0.11	0.80 ± 0.11

Discussion. The certificates provide provable bounds on the knowledge comprehension capability of a target LLM in an average case. QuaCer-C's certification results indicate an improvement in the knowledge comprehension capability as the number of parameters increases. This is evident for both the Llama and Vicuna models with their average values of the bounds and also for Gemini, which is the largest model certified overall. Across model classes, we observe that Mistral performs the best among the 7B parameter models, similar to the observation in previous benchmarks (Jiang et al., 2023; Beeching et al., 2023). The particular advantage of our certification method is highlighted in the results of Gemini-Pro (Gemini Team, 2024), wherein the range of certification bounds obtained with 95% confidence is entirely higher than those of other models, indicating that Gemini's worst performance in terms of knowledge comprehension will on average be better than the best performance of the other models, making it more reliable for high-risk situations.

4 Acknowledgement

We thank the anonymous reviewers for their insightful comments. This work was supported in part by NSF Grants No. CCF-2238079, CCF-2316233, CNS-2148583, and Google Research Scholar award.

References

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré,

Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin Vechev. Fast and precise certification of transformers. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pp. 466–481, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383912. doi: 10.1145/3453483.3454056. URL https://doi.org/10.1145/3453483.3454056.

Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval Estimation for a Binomial Proportion. *Statistical Science*, 16(2):101 – 133, 2001. doi: 10.1214/ss/1009213286. URL https://doi.org/10.1214/ss/1009213286.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data, 2021.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

C. J. Clopper and E. S. Pearson. The Use Of Confidence Or Fiducial Limits Illustrated In The Case Of The Binomial. *Biometrika*, 26(4):404–413, 12 1934. ISSN 0006-3444. doi: 10.1093/biomet/26.4.404. URL https://doi.org/10.1093/biomet/26.4.404.

Joseph Collins. Binomial distribution: Hypothesis testing, confidence intervals (ci), and reliability with implementation in s-plus. pp. 43, 06 2010.

Google Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

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, 2023.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023.

Daniel Kurz, Horst Lewitschnig, and Jürgen Pilz. Decision-theoretical model for failures which are tackled by countermeasures. *IEEE Transactions on Reliability*, 63(2):583–592, 2014. doi: 10.1109/TR.2014.2315952.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering, 2022.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models, 2023.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi Jaakkola, and Regina Barzilay. Conformal language modeling. 2024. URL https://arxiv.org/abs/2306.10193.

Murray Shanahan. Talking about large language models, 2023.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers, 2020.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019. doi: 10.1145/3290354. URL https://doi.org/10.1145/3290354.

Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Jason Vega, Isha Chaudhary, Changming Xu, and Gagandeep Singh. Bypassing the safety training of open-source llms with priming attacks, 2023.

Jinyuan Wang, Junlong Li, and Hai Zhao. Self-prompted chain-of-thought on large language models for open-domain multi-hop reasoning, 2023.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VVgGbB9TNV.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. Exploring the limits of chatgpt for query or aspect-based text summarization, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

A CERTIFICATION FRAMEWORK COMPONENTS

A.1 PROMPT CONSTRUCTOR

We use a prompt constructor designed to generate queries that facilitate a large language model's (LLM) navigation through a knowledge graph. The navigation process adheres to the constraint, ρ which we will refer to as k in this section, that limits the maximum number of inferential steps the LLM can take in its reasoning path. Our approach emphasizes explicit multi-hop reasoning while operating within the specified step count.

Algorithm 1 outlines the construction process. The input for the algorithm in the query_path we get from employing the PathGeneration detailed in Algorithm 2. This path is then transformed into a guiding query by the QueryGenerationPath function 3 that guides the LLM's reasoning. To provide essential context, the FormContext function gathers relevant Wikidata5m entries for all entities in the path and includes a distractor context to enhance task complexity. Additionally, during prompt generation, aliases for relations and entity in question are chosen at random using the GetAlias function. In order to support alias substitution and provide relevant information, the alias information for the target entity is added to the query context. The GetAlias function returns a randomly chosen alias(alternative name) for an input using the data from the wikidata5m dataset which is the basis of the knowledge graph we are using.

This procedure of prompt construction emphasizes traversing a graph within a constrained step count to construct a query.

Algorithm 1 Random Prompt Generation

Input: $query\_path$
Output: Query
$query, entity\_alias \leftarrow$ **QueryGenerationPath**($query\_path$, "")
$context \leftarrow$ **FormContext(**$query\_path$**)**
$context \leftarrow$ context + $query\_path[0]$ + "is also known as " $entity\_alias$
$prompt \leftarrow$ "Given Context $\{0\}$, answer $\{1\}$".format($context, query$)
return $prompt$

The path generation algorithm is outlined in 2. The algorithm commences by using the source node on which the subgraph is pivoted and subsequently chooses a random value (k_choice) between 1 and k, inclusive. This value dictates the length of the reasoning path. Next, a depth-first search generates a path of length k_choice from the source node.

Algorithm 2 PathGeneration

Input: Graph G, Integer k, Vertex $source$
Output: $query\_path$
Initialize $path$ to empty
$k\_choice \leftarrow$ **RandomChoice**([1, 2, ..., k])
$query\_path \leftarrow$ **DepthFirstSearch**(G, $source$, k_choice)
return $query\_path$

QUERYGENERATIONPATH

Algorithm 3 QueryGenerationPath

Require: $path$, cur_query
  if cur_query is empty **then**
    $cur\_query \leftarrow$ "What is "

    return **QueryGeneratePath**($path$, cur_query)
  else if length($path$) == 1 **then**
    $entity\_alias \leftarrow$ **GetAlias**($path$[0])
    $cur\_query \leftarrow cur\_query + entity\_alias$

    return $(cur\_query, entity_alias)$
  else
    $new\_query \leftarrow cur\_query +$ "the "+ **GetAlias**(relation(path[-2], path[-1])) + " of "

    return **QueryGeneratePath**($path[:-2]$, new_query)
  end if

The function initializes the query if cur_query is empty. If the path contains only a single node, that node is appended to the query. Otherwise, the function appends the descriptive relation between the last two vertices of the path to the current query and continues recursively with the remaining path.

EXAMPLE

Consider the following scenario within a graph: a path from Chandler Bing to Matthew Perry, then to 19 August 1969, with the relationships between Chandler Bing and Matthew Perry defined as "Actor" and between Matthew Perry and 19 August 1969 as "Birth Date." The algorithm would construct the query: "What is the birth date of the actor of Chandler Bing?" This query guides the LLM to reason through the defined relationships, demonstrating the algorithm's practical application.

A.2 RESPONSE CHECKER FUNCTION

To ensure the correctness of answers generated by a language model (LLM_A), we employ a two-step validation process.

Step 1: Alias Validation (String Matching)

Firstly, we leverage the Wikidata5m dataset to compile a comprehensive list of aliases associated with the answer entity. If any of these aliases are found within LLM_A's response, the answer is directly approved else, we move to step 2.

Step 2: Evaluation with LLM_Checker

Secondly, if no aliases match, we engage a separate language model (LLM_Checker) to validate LLM_A's output.

A.2.1 EVALUATION WITH LLM_CHECKER

LLM_Checker's verification process employs a dedicated checker function that takes in three inputs:

- The original query: Helps LLM_Checker understand the question.
- The correct answer: Serves as the ground truth.
- LLM_A's generated answer: The response to be assessed.

We format the evaluation task for LLM_Checker as a question-answering problem. This allows LLM_Checker to leverage its language comprehension and parametric knowledge to determine the alignment between LLM_A's answer and the ground truth.

We provide LLM_Checker with the following context for evaluation.

- "You are a helpful assistant. Your task is to assess inputs comprising a question, its correct answer, and an answer provided by a model. You should affirm with 'yes' if the model's answer is semantically equivalent to the correct answer, or 'no' if it is not."

Additionally, to ensure deterministic results we set the temperature of LLM_Checker to be 0.

Prompt Structure The assessment prompt provided to LLM_Checker follows this template:

"Question: {query} Ground Truth: {correct_answer}. Model Answer: {LLM_A_answer}."

For practical implementation, we use a Mistral-7B-Instruct-v0.2 as the checker LLM.

This method is useful because this allows better evaluation of correct answers in various forms such as synonyms. An example where this is useful is for the following: 'question': 'Which of the following best describes the structure that collects urine in the body?', 'correct_answer': 'Bladder', model_answer: 'Urinary Bladder'.

LLM_Checker Function Evaluation We validate the checker function's efficacy using 1000 samples from the MMLU (Hendrycks et al., 2021) dataset, divided equally between perturbed correct answers and incorrect answers. Perturbed answers are the same as the correct answers but not verbatim. To generate perturbed answers, we separately use the Gemini Pro model API to generate perturbed answers for question-answer pairs when possible. To do so, we give the model, the question, the correct answer, and the other options from the MMLU dataset. The model may or may not decide to perturb the answer. Additionally, to generate a wrong answer dataset, we simply use other options from the MMLU dataset that are not the correct answer for the given question. The accuracy of the Mistral 7b Instruct v2 with default settings is 85% when we set the temperature to 0.

B PREPROCESSING THE WIKIDATA5M KNOWLEDGE GRAPH

To ensure the generation of unambiguous queries and support the certification process, we preprocess the wikidata5m dataset.

1. **Relation Filtering:** We remove relations such as 'instance of', 'subclass of', and 'part of' due to their inherent potential for ambiguity in query formulation.
2. **Unique Relation Enforcement:** We retain only relations that establish a unique connection from a given entity. This means any relations where an entity might connect to multiple other entities (e.g., a 'parent of' relation) are disregarded. This filtering step is crucial for longer multi-hop queries, preventing scenarios where different paths might have conflicting properties.
3. **Reciprocal Support Verification:** We establish a requirement for reciprocal support within the descriptive text associated with entities. If entity A has a relation to entity B, either entity B's supporting text must mention entity A or vice versa. Relations failing this check are discarded, ensuring that the context provided by the prompt generator will likely contain sufficient information to resolve the query.
4. **Unicode Conversion:** For consistency within our experiments, we convert all text containing Unicode characters into their respective ASCII approximations.

Rationale: These preprocessing steps promote the generation of clear, well-defined queries while supporting the integrity of the certification framework.