Towards Consistent Language Models Using Declarative Constraints

Jasmin Mousavi[1], Arash Termehchy[1]

[1]Oregon State University, 1500 SW Jefferson Ave, Corvallis, OR 97331, United States

Abstract

Large language models have shown unprecedented abilities in generating linguistically coherent and syntactically correct natural language output. However, they often return incorrect and inconsistent answers to input questions. Due to the complexity and uninterpretability of the internal learned representations, it is challenging to modify language models such that they provide correct and consistent results. The data management community has developed various methods and tools for repairing inconsistent datasets. In these methods, users specify the desired properties of data in a domain in the form of high-level declarative constraints. This approach has provided usable and scalable methods to delivering consistent information from inconsistent datasets. We propose to build upon this success and leverage these methods to modify language models such that they deliver consistent and accurate results. We investigate the challenges of using these ideas to obtain consistent and accurate language models.

Keywords

large language models, declarative constraints, consistent modeling, model repair

1. Introduction

Large language models (LLMs) have shown unprecedented abilities in processing natural languages [1, 2]. They effectively generalize to perform various tasks with few or no training examples. Thus, there is a rapidly growing interest in using them to solve data-driven problems, such as, interactive question answering.

Nonetheless, LLMs often provide incorrect answers to input queries and perform inaccurate inferences [3, 2]. Some studies indicate the recent LLMs provide up to 40% erroneous answers to factual questions [2]. These erroneous results are important obstacle for wide-spread use of LLMs in real-world applications.

To address the problem of inaccurate answers returned by LLMs, we should recognize that **LLMs are not knowledge bases, but rather probabilistic or approximate models of factual information**. LLMs may overgeneralize patterns and relationships observed in the sub-sequences of pretraining documents, which might lead to returning spurious relationships and inaccurate results. The uninterpretable mixture of linguistic patterns and factual information has made it challenging to eliminate incorrect information. It is in sharp contrast to traditional approaches to database querying in which the user interface, e.g., query language, is clearly separated from the source of the information, e.g., databases.

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — Workshop on LLMs and Databases (LLMDB'23), August 28 - September 1, 2023, Vancouver, Canada

mousavij@oregonstate.edu (J. Mousavi); termehca@oregonstate.edu (A. Termehchy)
https://web.engr.oregonstate.edu/~termehca/ (A. Termehchy)

One approach is to *fine-tune* the LLM on a set of domain-specific data sources to improve the quality of its answers for questions in a given domain [4]. Nonetheless, it has shown that these methods may also lead to many inaccurate answers [5]. This is, in part, due to the fact that fine-tuning is inherently under-specified and may not sufficiently modify the model to eliminate its already learned spurious information. Another approach is to augment LLMs with *additional and potentially relevant information* from external data sources [6, 7, 8]. These methods often add extra information to the context considered during pretraining. This line of research have improved the accuracy of LLMs to a limited degree as it does not address the core issue of having spurious and incorrect information in LLMs. It is not clear whether adding more relevant information eliminate inaccurate information stored in the model. Moreover, it is often challenging to find sufficiently many relevant data sources, particularly for long-tail entities.

It is challenging to ensure that an LLM learns accurate generalizations and returns correct answers as it may require perfect knowledge of unobserved data. Nonetheless, we may be able to restrict its pretrained representation to **adhere to semantic constraints** in the domain to avoid generating incorrect results.

This is akin to the problem of **cleaning databases** to satisfy a set of declarative semantic constraints [9]. Databases often contain data that does not comply with the semantic constraints in their domains. For example, a person might not have any social security number or have more than one in a human resource database. The usual query processing methods might return inaccurate results over incomplete or inconsistent databases. The data management community has developed a unified, usable, and scalable approach to repairing inconsistent data to comply with declarative semantic constraints [9]. Instead of writing long and complex imperative programs to check inconsistencies and repair the data, users specify the properties of the consistent dataset succinctly in a high-level declarative language. There are several types of constraints based on the model of the data, e.g., functional dependencies for relational data or description logic rules for RDF data. They are usually subsets of first order logic that are sufficiently expressive to capture important knowledge in the domain yet not too expressive to make reasoning intractable. Hence, data systems may check incompatibilities or redundancies in constraints efficiently. These constraints may also be learned from high-quality datasets in the domain.

In this paper, we propose a novel approach to reduce inconsistencies in LLMs using high-level declarative constraints. We believe that the success of using declarative constraints to provide reliable information in data management indicates that our proposed approach has the potential to deliver a usable and scalable method for creating and maintaining reliable and consistent LLMs. This, in turn, enables users to leverage LLMs in real-world applications with high confidence and accuracy.

We also discuss challenges of using high-level declarative constraints to reduce inconsistencies in LLMs. Specially, it is not clear how to enforce declarative constraints in an LLM efficiently. It might be challenging to find correspondence between the symbolic declarative constraints and information in the continuous representation learned by LLMs. We investigate how to leverage existing ideas in data cleaning and management [9] and current methods to embedding structured information [10, 11, 12] to address this problem. Since pretraining and fine-tuning are often time-consuming and computationally expensive, we also investigate methods of updating a pretrained LLM that ensures it follows a set of constraints.

2. Creating Consistent Models Using Pretraining & Fine-tuning

Since LLMs are created using pretraining, it is natural to consider methods that incorporate semantic constraints during pretraining to create consistent LLMs. Nonetheless, pretraining usually takes long time and substantial computational resources. Researchers often use a relatively fast process called *fine-tuning* to modify a pretrained LLM [8]. During fine-tuning, the LLM is trained with additional information using its pretrained weights as initial values. In this section, we explore methods for creating or modifying an LLM so that complies with a set of constraints using pretraining and fine-tuning.

2.1. Constraints

The semantic properties and constraints in a domain are often represented in form of *ontologies* [13]. In a nutshell, an ontology consists of a set of facts, where each fact is a triple in the form of (subject, relationship, object), and a set of constraints on these facts. The triples in an ontology introduce concepts, e.g., *Person*, and their instances, e.g., *Obama*. They also represent relationship between different concepts in the domain, e.g., *President is-a Person*. Constraints in an ontology lay out the conditions that concepts and relationships must follow, e.g., *is-a* has the transitive property. Constraints are usually expressed in a subset of first order logic, e.g., description logic. Generally speaking, each constraint establishes that if some concepts satisfy certain conditions, i.e., premise, they must satisfy other conditions, i.e., conclusion. For instance, for *is-a* relation, we have for all concepts x, y, z, if $(x, is-a, y)$ and $(y, is-a, z)$, then $(x, is-a, z)$.

It is important for an LLM to encapsulate both the facts and the constraints on those facts in a domain to provide consistent results. An LLM might not learn the facts from the textual data over which it is pretrained. It could be because some facts are not in the text or do not appear in closely related text spans and contexts. Constraints in an ontology represent semantic meaning of concepts and relationships in the domain. This information does not often appear explicitly in the data used to pretrain LLMs, therefore, LLMs might not learn them during pretraining.

Thus, our goal is to create LLMs that contain and follow both facts and constraints in a given ontology. To simplify our exposition and because each fact can also be represented as a special type of constraint, unless otherwise noted, we refer to both facts and constraints in an ontology as constraint.

2.2. Mixing Constraints with Training Data

Incorporating this structured information into LLMs poses challenges since LLMs are trained on unstructured data. One may supplement the training data with textual ontology information, e.g., *Obama is a President*. However, translating facts and constraints into text introduces two problems. First, in domains containing numerous semantic constraints, the augmented training data may exceed maximum sequence lengths (commonly restricted to 512 in most models). Second, converting structured data into unstructured text may cause the model to view this information merely as additional context, without preserving higher-order constraints vital for comprehending semantics of concepts in the domain.

To overcome these issues, constraint reduction techniques can be applied. One method involves reasoning over the constraints to find a minimal set [14], but does not guarantee that the augmented input will conform to the maximum sequence length. Another approach is to encode the ontology information into an embedded representation using an LSTM [6], integrated via a gating function. This allows the LLM to control what information augments the input, successfully limiting the sequence length. However, it may not be optimal for incorporating constraints, as it may cause information loss and is more apt for enhancing input with extra facts, rather than filtering incorrect information.

These methods fall short of incorporating the ontology in a way that preserves its semantic information, highlighting the difficulty of integrating high order constraints into LLMs.

2.3. Retaining Constraint Information

Constraint Embedding. Ideally, the representation learned by a LLM should capture the structural information present in the semantic constraints of an ontology. Geometric embeddings (e.g. box, circle, cone) have been widely explored for learning representations of graph structures such as ontologies and knowledge bases [15, 11, 16, 10, 12]. For instance, if an ontology has the constraint that *President is-a Person*, the geometric embedding for *Person* should contain the geometric embedding for *President*, reflecting the transitive property *is-a* and that President is a subset of Person. These embeddings preserve the structural properties and relationships in an embedded space, ensuring that the output representations maintain the specified constraints.

When training an LLM, one can incorporate geometric or constraint embeddings for unstructured text data in order to retain information from ontologies. If the ontology data is consistent and the model learns a perfect constraint embedding, it should respect the facts and constraints within the domain. However, since this is unlikely, it may be necessary to apply optimization techniques to the objective function. Such techniques can help facilitate LLMs to learn representations that effectively capture higher order relationships and constraints that extend beyond the training domain.

Constraint Objective Task. Since the ontology is a source of knowledge, then it can also be used to train the LLM directly. External knowledge can be created from the ontology by extracting triples in the form of rich text spans, thereby providing more information about constraints to the model. Using this data, one may construct a word prediction or masking objective that aligns with the external knowledge of semantic constraints. One strategy is type modeling [17], where entities are replaced with their type, and the model predicts the entity type for the next word or word span. This idea can be extended to a masking objective, where the model predicts masked types in the output.

Alongside traditional LLM objectives, e.g., masked objective tasks, one can integrate constraint objective tasks and constraint embeddings during pretraining. These methods capture the ontology's structural information, resulting in a model that is consistent with domain-specific constraints. Given an ontology and text documents, constraint objective tasks and constraint embeddings can also be used for fine-tuning. However, these techniques may prove more effective if implemented during the pretraining process.

3. Model Repair

To ensure that a database complies with a constraint, we often find the information in the database that do not follow the constraint and update them so the database satisfies the constraint. One may adopt this approach to repair a pretrained model so it satisfies a set of given constraints. In other words, one may find the portion of the model responsible for representing a constraint or lack thereof and update them if necessary so that the resulting model satisfies the constraint. As opposed to representing information in a database, factual information is stored in an LLM implicitly and through some pretrained weights in a model. Hence, it is difficult to find and revise the factual data that violates a set of given constraints in an LLM. In this section, we describe two approach to repairing pretrained models and discuss their challenges.

3.1. Fact-based Repair

There has been some recent success in updating facts represented in an LLM [18]. Each update aims at changing the object in a given triple in form of (subject (s), relation (r), object (o)) to a new object o'. These methods first find the weights responsible for representing o and its relationship to s in the model. They then modify these weights so that the model represents the new object o' in the fact with high probability.

Building upon this line of work, one may ensure that an LLM satisfies a set of constraints by finding and modifying the pretrained weights that represent the facts that violate the constraints. An algorithm to check whether an LLM satisfies a given constraint could be as follows. First, the algorithm samples a set of facts that follow the constraint from the ontology. For each instance of a constraint, it will prompt/query the LLM to check whether and how the LLM represents the facts in the instance. If the LLM's representations of the facts in the instance violate the constraint, the algorithm modifies the representations so they follow the constraint. The larger the set of samples is, the more likely the repaired model satisfies the constraint. Users can change the size of the sample based on their available time and resources as well as desired confidence for satisfying constraints.

This algorithm might require a large number of updates to the model, which could be time-consuming. Moreover, since facts are represented implicitly in the model, the aforementioned methods might not always find the updates that modify a fact to its desired form. To address these challenges, one may find a minimal set of facts and their corresponding update operations such that modifying their representations in the model will most likely create a model that follows the constraint. The repair algorithm, then, will update the weights in the model for facts in this minimal set.

It is known that there are often many possible modifications of an inconsistent dataset to satisfy a set of constraints. It is challenging to maintain and query all these repairs of databases. Hence, researchers have proposed heuristics to choose a few of these repairs, e.g., the ones that differ the least from the original database. The same problem might also happen in repairing models. One may use similar approaches to reduce the number of repaired models.

3.2. Constraint-based Repair

It may take a long time to update a large number of facts in a model [18]. Thus, the approach of fact-based repair may efficiently modify the model to satisfy constraints with a relatively few instances, e.g., facts in the ontology, but it might be computationally challenging to do for constraints with many instances. Also, if a constraint has many instances, this approach might deliver many possible model repairs even after applying the aforementioned heuristics to reduce the space of possible repairs. Therefore, it will be challenging to query or train these models for a given task.

LLMs generalize input data during pretraining. They have also been successfully used to generate data that closely resembles real-world data and train accurate models using a relatively few training examples for various tasks. Hence, we hypothesize that they might represent some constraints in the domain in whole or in part. If this hypothesis is true, an LLM does not satisfy some constraints because the LLM might represent them incompletely or erroneously.

Hence, to ensure that the model satisfies a constraint, instead of repairing all facts that violate the constraint, one might change directly the portion of the model that represents a constraint. This portion might be significantly smaller than the parts that represent the violating facts. Thus, it might be substantially faster and easier to find the weights in the model responsible for incomplete or erroneous representation of the constraint than doing the same for all facts that violate that constraint.

4. Related Works

Lexical Constraints for Language Models. There has been recent effort on limiting the output of LLMs so they *follow given syntactical patterns*, e.g., not contain certain keywords [5, 19, 20]. In these systems, users write (imperative) programs that detect some invalid patterns in the output of LLMs. These systems, then, use constrained optimization or probabilistic inference over the sequences generated by the LLM to reduce the probability of the outputs with invalid patterns. These efforts are steps in the right direction but fall short of providing a usable and scalable method to deliver consistent information over LLMs. First, they do not generally support semantic constraints. Second, users may have to write multiple and possibly long programs to *clean up* the output of the model. As some domain may have numerous constraints, it is challenging to develop and maintain these programs. Users must check manually whether these programs are consistent with each other and there is no redundancy across different programs. Third, they are usually applied only during the decoding stage, therefore, the LLM may still learn and represent spurious relationships. As it is challenging to interpret learned representations in LLMs, it is difficult to control all the implications of their learned imprecise information. For instance, the learned spurious relationship about one entity might impact how an LLM answers a question about a different but related entity. As opposed to this line of work, we propose an end-to-end approach that uses declarative semantic constraints to reduce inconsistent information in LLMs.

Self-Consistency of Language Models. It is known that language models produce contradictory answers to the questions that seek the same information but phrased differently. Researchers have proposed methods to address this issue by prompting the language model to critique and refine its own output during inference [21]. This method prompts the language model with differently phrased questions and builds a (weighted) model over answers to infer the most likely result. We, however, mainly focus on ensuring that the language model follows semantic constraints.

Extracting Knowledge from Language Models. Researchers have proposed methods to extract generic statements or factual knowledge from language models using prompt engineering and human supervision [22]. The prompts are constructed in a way that encourages succinct factual statements. They use human labeled data to detect inaccurate outputs and fine-tune the language model. However, it might be challenging to collect a sufficient amount of training data to extract accurate statements.

Querying Language Models. There has been some recent effort to design programming languages for prompting large language models, i.e., *language model* *programming* [23, 24, 25]. There are generally domain-specific programming languages to extract information from and control the output of a large language model to satisfy the users' input hard constraints, akin to where conditions in SQL queries. Some of these languages resemble database query languages, e.g., SQL [24]. These languages aim at making it easier to query and prompt and optimize the number of calls to large language models. However, these languages do not generate consistent results conditioned on domain constraints. Thus, they may return answers that violate semantic constraints in the domain.

References

[1] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training (2018).

[2] OpenAI, Gpt-4 technical report, 2023. arXiv:2303.08774.

[3] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, P. Fung, Survey of hallucination in natural language generation, ACM Computing Surveys 55 (2023) 1–38. URL: https://doi.org/10.1145%2F3571730. doi:10.1145/3571730.

[4] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, L. R. Varshney, Recipegpt: Generative pre-training based cooking recipe generation and evaluation system, in: Companion Proceedings of the Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 181–184. URL: https://doi.org/10.1145/3366424.3383536. doi:10.1145/3366424.3383536.

[5] X. Lu, P. West, R. Zellers, R. Le Bras, C. Bhagavatula, Y. Choi, NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021. URL: https://aclanthology.org/2021.naacl-main.339. doi:10.18653/v1/2021.naacl-main.339.

[6] Q. Liu, D. Yogatama, P. Blunsom, Relational memory augmented language models, 2022. arXiv:2201.09680.

[7] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, L. Sifre, Improving language models by retrieving from trillions of tokens, 2022. arXiv:2112.04426.

[8] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, T. Scialom, Augmented language models: a survey, 2023. arXiv:2302.07842.

[9] W. Fan, F. Geerts, Foundations of Data Quality Management, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2012. URL: https://doi.org/10.2200/S00439ED1V01Y201207DTM030. doi:10.2200/S00439ED1V01Y201207DTM030.

[10] M. Kulmanov, W. Liu-Wei, Y. Yan, R. Hoehndorf, El embeddings: Geometric construction of models for the description logic el++, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 6103–6109. URL: https://doi.org/10.24963/ijcai.2019/845. doi:10.24963/ijcai.2019/845.

[11] H. Ren, W. Hu, J. Leskovec, Query2box: Reasoning over knowledge graphs in vector space using box embeddings, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJgr4kSFDS.

[12] M. Jackermeier, J. Chen, I. Horrocks, Box^{2}el: Concept and role box embeddings for the description logic el++, 2023. arXiv:2301.11118.

[13] M. Krotzsch, F. Simancik, I. Horrocks, A description logic primer, arXiv preprint arXiv:1201.4089 (2012).

[14] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases: The Logical Level, Addison-Wesley, 1994.

[15] W. L. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, J. Leskovec, Embedding logical queries on knowledge graphs, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Curran Associates Inc., Red Hook, NY, USA, 2018, p. 2030–2041.

[16] D. Garg, S. Ikbal, S. K. Srivastava, H. Vishwakarma, H. Karanam, L. V. Subramaniam, Quantum embedding of knowledge for reasoning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/cb12d7f933e7d102c52231bf62b8a678-Paper.pdf.

[17] M. R. Parvez, S. Chakraborty, B. Ray, K.-W. Chang, Building language models for text with named entities, arXiv preprint arXiv:1805.04836 (2018).

[18] K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, D. Bau, Mass-editing memory in a transformer, in:

ICLR, 2023.

[19] H. Zhang, M. Dang, N. Peng, G. Van den Broeck, Tractable control for autoregressive language generation, in: Proceedings of the 40th International Conference on Machine Learning (ICML), 2023. URL: https://arxiv.org/pdf/2304.07438.pdf.

[20] A. K. Lew, T. Zhi-Xuan, G. Grand, V. K. Mansinghka, Sequential monte carlo steering of large language models using probabilistic programs, 2023. arXiv:2306.03081.

[21] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, P. Clark, Self-refine: Iterative refinement with self-feedback, 2023. arXiv:2303.17651.

[22] C. Bhagavatula, J. D. Hwang, D. Downey, R. L. Bras, X. Lu, L. Qin, K. Sakaguchi, S. Swayamdipta, P. West, Y. Choi, I2d2: Inductive knowledge distillation with neurologic and self-imitation, 2023. arXiv:2212.09246.

[23] L. Beurer-Kellner, M. Fischer, M. Vechev, Prompting is programming: A query language for large language models, Proceedings of the ACM on Programming Languages 7 (2023) 1946–1969. URL: https://doi.org/10.1145%2F3591300. doi:10.1145/3591300.

[24] Microsoft, S. Lundberg, Guidance: A guidance language for controlling large language models, https://github.com/microsoft/guidance, 2023.

[25] N. Computing, R. Louf, Outlines: Generative model programming, https://github.com/normal-computing/outlines, 2023.