

# Fine-tuning Large Enterprise Language Models via Ontological Reasoning

Teodoro Baldazzi<sup>1</sup>, Luigi Bellomarini<sup>2</sup>, Stefano Ceri<sup>3</sup>, Andrea Colombo<sup>3</sup>,  
Andrea Gentili<sup>2</sup>, and Emanuel Sallinger<sup>4,5</sup>

<sup>1</sup> Università Roma Tre, Italy

<sup>2</sup> Banca d'Italia, Italy

<sup>3</sup> Politecnico di Milano, Italy

<sup>4</sup> TU Wien, Austria

<sup>5</sup> University of Oxford, UK

**Abstract.** Large Language Models (LLMs) exploit fine-tuning as a technique to adapt to diverse goals, thanks to task-specific training data. Task specificity should go hand in hand with domain orientation, that is, the specialization of an LLM to accurately address the tasks of a given realm of interest. However, models are usually fine-tuned over publicly available data or, at most, over ground data from databases, ignoring business-level definitions and domain experience. On the other hand, Enterprise Knowledge Graphs (EKGs) are able to capture and augment such domain knowledge via ontological reasoning. With the goal of combining LLM flexibility with the domain orientation of EKGs, we propose a novel neurosymbolic architecture that leverages the power of ontological reasoning to build task- and domain-specific corpora for LLM fine-tuning.

**Keywords:** Ontological reasoning · Language models · Knowledge graphs.

## 1 Introduction: Context and Overview of the Approach

With the recent soar of AI-based chatbots, currently led by OpenAI’s ChatGPT, the field of Natural Language Processing (NLP) and, in particular, Large Language Models (LLMs), faced a major turning point and transcended its relevance in academia and industry, steering the attention of the general public towards generative AI. While many approaches are being proposed that exploit powerful pre-trained LLMs, such as T5 [18] and GPT [16], to address a plethora of industrial tasks, current solutions show limited effectiveness at specializing the models on enterprise domains, from finance to genomics. In our community, such domain-specific knowledge can be captured by combining factual data from corporate databases with business-level definitions as ontologies in *Enterprise Knowledge Graphs* (EKGs), and further augmented via *ontological reasoning*. In this paper, we build upon this domain representation and propose a novel solution to accurately specialize LLMs on core enterprise NLP tasks.

**Limits of task-specific fine-tuning.** LLMs can be pre-trained on extensive datasets and, often, specialized with a *fine-tuning* process that customizes them

so as to perform given NLP tasks [20], such as *question-answering*, *language translation*, *named-entity recognition*, *document summarization*, *sentiment analysis*, and more [7]. According to a very common usage pattern, general-purpose LLMs are fine-tuned for a specific NLP task based on extensive cross- or domain-generic textual corpora that are publicly available [17].

While this approach highlights good generalization capabilities and a surprising human-style interaction, the obtained models have major shortcomings in that they lack enterprise knowledge and trivially fail to solve domain-specific NLP tasks. For instance, in the financial domain, state-of-the-art yet generalist models have shown poor performance for different NLP tasks, for which, on the other hand, further fine-tuning with large additional text corpora has been proved to be helpful in improving the results, such as in the case of *FinBert* [12].

**Limits of domain-specific fine-tuning.** Going further, recent studies are exploring the usage of factual data from enterprise databases to fine-tune LLMs and try to tackle domain-specific question-answering tasks: the factual information is leveraged to synthesize prompt-response pairs based on the data and customize the LLM in a task- and domain-specific direction. A primary example is the *SKILL project* [15], where an LLM is directly trained on factual triples derived from the translation into natural language—the so-called *verbalization*—of Wikidata (namely, the *KELM corpus* [2]) for question-answering tasks. Similarly, other approaches highlight possible improvements of accuracy in question-answering tasks, when textual information is first captured into a database, which is then verbalized and employed for fine-tuning [3].

Yet, even the combination of general-purpose knowledge of the pre-trained model and the domain data still offers an accuracy that is not acceptable for core tasks in specialized domains. For example, *BloombergGPT* [22] is an LLM fine-tuned on a wide range of financial data, combining internal enterprise knowledge with publicly-available datasets. The results show that the model fine-tuned for the question-answering task outperforms state-of-the-art counterparts by being able to correctly answer questions related to the financial domain. However, *BloombergGPT* has been tested only on questions whose answers are already contained in (or directly entailed by) the factual information of the input databases, either as data or meta-data (e.g., schema information). It is reasonable, in fact, that it does not have enough fine-tuning data or logical capabilities to go further.

**A look beyond current solutions.** Conversely, from an enterprise application perspective, it would be extremely useful to answer questions by means of intelligent combined uses of the input databases with other logic-intensive sources of knowledge (e.g., regulatory bodies, best practices, domain experts, etc.). For instance, in the context of financial cases like those of interest for a Central Bank, answering questions such as “*why does shareholder X exert a relevant influence on the financial intermediary Y?*” (**explanation**), or “*how does this smart contract behave?*” (**description**), or “*is the merger of banks Y and W lawful from a regulatory perspective?*” (**question answering**), or “*based on data, how many ties with other intermediaries does Z have?*” (**text-to-query translation**) would be an essential asset.

At present, all the mentioned tasks are far from being solved by off-the-shelf libraries or, directly, by most recent LLMs, and are open research. Going into the details of each of them is beyond the scope of this paper, but the motivations laid out mainly in a question-answering perspective give the flavour of why LLMs are not enough. It is worth remarking, though, that even the translation task, for which thanks to LLMs much progress has been made in the transformation of natural language into the target query languages (say, SQL, SPARQL, etc.) [21,23] is still a largely unsolved problem, especially in the context of languages with an elaborate grammar and complex queries [9].

**Ontological reasoning.** From another perspective, in the *Knowledge Representation and Reasoning* [11] (KRR) community, the state-of-the-art research on ontological reasoning over EKGs makes a point of being able to offer a compact combination of factual database information (the *extensional knowledge*) and formally specified business awareness, for instance in the form of logical rules (the *intensional knowledge*), to serve *domain-specific query answering* in an accurate manner. For example, logical KGs exploiting efficient fragments of the *Datalog<sup>±</sup>* family [8] have been successfully adopted for financial applications [6].

Yet, there is an impedance mismatch between NLP and ontological reasoning, which lacks the flexibility and the language orientation to solve explanation, description, question answering, and translation tasks: queries need to be specified in KRR formalisms; all the inputs and the results are facts/n-tuples/triples; the generation of new knowledge is possible only to the extent reasoning rules capture it. Conversely, while being very good at manipulating human language, LLMs lack a comprehensive domain model, a pillar of KRR approaches.

**An integrated approach.** This paper strives to strengthen LLMs in their use for task- and domain-specific applications, by letting the fine-tuning process be driven by an ontological reasoning task on an EKG. We operate in the context of the VADALOG [5] system, a Datalog-based reasoning engine for EKGs, that finds many industrial applications [6]. We use VADALOG to explore the factual information derived by applying the domain rules, via the CHASE procedure [13], to the enterprise data and synthesize a fine-tuning corpus that covers the entire “reasoning space” to convey domain-specificity to the LLM. A summary of the resulting *fine-tuning pipeline*, provided in Figure 1, will guide our discussion.

More in detail, our **contributions** can be summarized as follows.

- We present a **reasoning verbalization** technique that generates sets of prompt-response pairs from ground Datalog rules. We provide the algorithm and optimize it with a *lifting technique* exploiting reasoning regularities.
- We deliver such an approach in a **novel neurosymbolic architecture** that fine-tunes task-specific LLMs for a set of four relevant NLP tasks, namely, *explanation*, *description*, *question answering*, and *translation*.
- We discuss a preliminary **proof-of-concept** confirming the validity of our approach and comparing models fine-tuned on ground and chase data.

**Overview.** In Section 2 we present our architecture. A preliminary experimental validation is provided in Section 3. We draw our conclusions in Section 4.

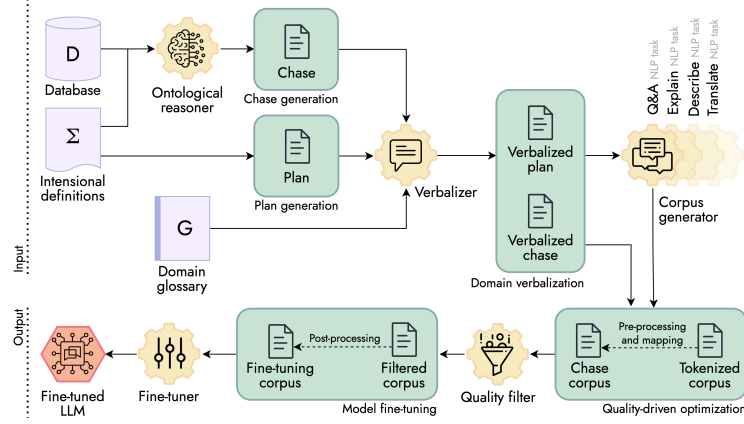


Fig. 1: Neurosymbolic pipeline for reasoning-based LLM fine-tuning.

## 2 A Neurosymbolic Pipeline to Fine-tune LLMs

The input blocks of the fine-tuning pipeline in Figure 1 are  $D$  and  $\Sigma$ . They are, respectively, a database of domain facts and a set of reasoning rules, capturing the business dynamics. Our rules are expressed in VADALOG. An EKG is a combination  $\Sigma(D)$  of  $D$  and  $\Sigma$ , obtained through reasoning. The set  $\Sigma(D)$  is computed via the CHASE [13]: starting from  $\Sigma(D) = D$ , the chase augments  $\Sigma(D)$  with facts derived from the application of the rules in  $\Sigma$  to fixpoint.

Let us introduce our running example: a simple trading activity managed with a *smart contract* [14]. Here,  $D$  contains a log over time of buy/sell orders from the traders who invest in the smart contract as well as market information, e.g., asset prices (*Price*), or market shutdowns (*MarketClosed*).

*Example 1.* The following set  $\Sigma$  contains the VADALOG rules governing the basic functioning of the market, i.e., under which conditions the orders are accepted and how profits and losses are computed.

$$\text{Open}(x, y, t_1), \neg \text{MarketClosed}(t_1) \rightarrow \text{Accepted}(x, y, t_1) \quad (1)$$

$$\text{Accepted}(x, y, t_1), \text{Price}(p_1, t_1), k = y * p_1 \rightarrow \text{Position}(x, y, k, t_1) \quad (2)$$

$$\text{Close}(x, t_2), \text{Price}(p_2, t_2), \text{Position}(x, y, k, t_1), \\ t_2 > t_1, pl = y * p_2 - k \rightarrow \text{Return}(x, pl) \quad (3)$$

If a trader  $x$  wants to open a position (buy) on a certain asset of size  $y$  at time  $t_1$  and the market is open at  $t_1$ , the order is accepted (rule 1). If the order by  $x$  is accepted and the asset price at  $t_1$  is  $p_1$ , then  $x$  holds a position on the market at time  $t_1$  of size  $y$  and of notional (total value)  $k$  equal to  $y * p_1$  (rule 2). If, later at  $t_2$ , trader  $x$  decides to close its position (sell) and the price at  $t_2$  is  $p_2$ , then  $x$  gets returns (profits or losses) from its trading activity as  $y * p_2 - k$  (rule 3).

Applying the vision we laid out to Example 1, the goal of our pipeline is fine-tuning an LLM to address *explanation*, *description*, *question answering*, and

**Algorithm 1** Reasoning-based LLM Fine-tuning.

---

```

1: function REASONINGFINETUNING( $D, \Sigma, G, model, nlpTask$ )
2:    $chase \leftarrow \text{VADALOG.reason}(D, \Sigma)$  ▷ chase generation
3:    $verbChase \leftarrow \emptyset$ 
4:   for each  $step$  in  $chase$  do
5:      $stepAggrContrib \leftarrow \emptyset$ 
6:     if  $\text{hasAggregate}(step.getRule())$  then
7:        $stepAggrContrib \leftarrow \text{composeBack}(step, chase)$  ▷ aggregates retrieval
8:        $verbStep \leftarrow \text{verbalizeStep}(step, stepAggrContrib, G)$ 
9:        $verbChase \leftarrow verbChase \cup \{verbStep\}$  ▷ chase verbalization
10:   $verbPlan \leftarrow \text{verbalizePlan}(\Sigma.getLogicPlan())$  ▷ logic plan verbalization
11:   $tokenizedCorpus \leftarrow \text{generate}(\text{preprocess}(verbPlan, nlpTask))$  ▷ tokenized corpus generation
12:   $chaseCorpus \leftarrow \emptyset$ 
13:  for each  $verbStep$  in  $verbChase$  do ▷ chase mapping
14:     $chasePromptResp \leftarrow \text{map}(tokenizedCorpus, verbStep)$ 
15:     $chaseCorpus \leftarrow chaseCorpus \cup \{chasePromptResp\}$ 
16:    for each pair  $\langle prompt, resp \rangle$  in  $chaseCorpus$  do ▷ quality-driven optimization
17:       $qualityScore \leftarrow \text{checkQuality}(\langle prompt, resp \rangle, nlpTask, verbChase)$ 
18:      if  $qualityScore \leq \text{threshold}$  then
19:         $chaseCorpus \leftarrow chaseCorpus \setminus \{\langle prompt, resp \rangle\}$ 
20:      else
21:         $chaseCorpus \leftarrow chaseCorpus \cup \text{paraphrase}(\langle prompt, resp \rangle)$  ▷ corpus paraphrasing
22:   $fineTuningCorpus \leftarrow \text{postprocess}(chaseCorpus)$ 
23:   $ftModel \leftarrow \text{fineTune}(model, fineTuningCorpus)$  ▷ model fine-tuning
24:  return  $ftModel$ 

```

---

*text-to-query translation* tasks for the simple trading activity at hand. Let us follow Figure 1 and Algorithm 1 to describe the application of the pipeline to a database  $D = \{Open(EGTech, 0.3, 1), Open(IEComp, 0.5, 1), Price(124, 1), Price(147, 9), Close(EGTech, 9), MarketClose(5)\}$ .

**Chase generation.** The first step of our pipeline builds the chase  $\Sigma(D)$ , that is, the expansion of  $D$  with the facts that can be derived by applying the rules of  $\Sigma$  (line 2, in the algorithm). Rule 1 generates the fact  $Accepted(EGTech, 0.3, 1)$ , as the market is not closed at time 1. Then,  $Position(EGTech, 0.3, 37.2, 1)$  is derived via rule 2. Finally, as trader *EGTech* closes the position, i.e., sells the asset, at time 9 and the price goes up to 147\$, then *EGTech* gets a profit of 6.9\$.

**Domain verbalization.** Whenever a VADALOG rule is involved in the CHASE, it is translated into pure text with a deterministic transformation, based on the *select-project-join* semantics, which looks up a *glossary*  $G$  of atom descriptions. When rules involve aggregation functions, allowed in VADALOG, the process is less straightforward and involves unfolding a chain of chase activations altogether [1] (line 7). At the end of this phase, we are in hold of a “*since-then* closure” of our domain, that focuses on what can be obtained by activating the intensional knowledge of  $\Sigma$ . From another perspective,  $\Sigma$  can be seen as an *attention* mechanism, to select the fragment of  $D$  that one wants to verbalize. For instance, with respect to our running example, the chase step  $Open(EGTech, 0.3, 1), \neg MarketClose(1) \rightarrow Accepted(EGTech, 0.3, 1)$  (rule 1) is verbalized as: *Since the trader EGTech at time 1 sends an order to open a position of size 0.3, and it is not true that 1 is a time when the market is closed, then the order of size 0.3 by EGTech is accepted at time 1.*

**Fine-tuning corpus generation.** With the basic verbalization available, we are now ready to generate the fine-tuning corpus. We consider the corpus gen-

eration itself as a text manipulation task and exploit the effectiveness of powerful pre-trained LLMs [7], such as GPT-3, to synthesize a finite set of possible prompt-response pairs. Here we have two goals: 1) minimising the number of “calls” to the LLM, for cost- and time-efficiency reasons; 2) avoiding any ground value (coming from the EKG) being disclosed to the LLM, for data protection reasons. We leverage the regularity of logical languages and resort to a *lifting technique*. We build a *logic plan* out of  $\Sigma$  (line 10). A plan is the equivalent in our context of a database execution plan and can be seen as the dependency graph of the rules of  $\Sigma$ , where nodes represent rules and edges stand for head-body dependencies. The plan is then verbalized, obtaining a text with tokens as placeholders for rule variables. Finally, a tokenized fine-tuning corpus is generated from the plan, after minor pre-processing (line 11). The form of the prompts depends on the task. Now, for each verbalized chase step, we look up the corresponding verbalized portion of the plan and instantiate its tokens (lines 13-15). Note that no invocations to the *corpus generator* are needed in this phase. Figure 2 exemplifies the generation process in our example domain.

Plan	Tokenized corpus	Verbalized chase step	Chase corpus
$Open(x, y, t_1),$ $\neg MarketClosed(t_1)$ $\rightarrow Accepted(x, y, t_1)$	Q1: What is the size of the order accepted sent by trader $\langle x \rangle$ ?	Since the trader EGTech at time 1 sends an order to open a position of size 0.3, and it is not true that 1 is a time when the market is closed, then the order of size 0.3 by EGTech is accepted at time 1.	Q1: What is the size of the order accepted sent by trader EGTech?
	A1: The size is $\langle y \rangle$		A1: The size is 0.3
	Q2: Why was the order sent by trader $\langle x \rangle$ at time $\langle t_1 \rangle$ accepted?		Q2: Why was the order sent by trader EGTech at time 1 accepted?
	A2: Because at time $\langle t_1 \rangle$ the market was open		A2: Because at time 1 the market was open
	...		...
$Open(x, y, t_1),$ $\neg MarketClosed(t_1)$ $\rightarrow Accepted(x, y, t_1)$ $\downarrow$ $Accepted(x, y, t_1),$ $Price(p_1, t_1), k = y * p_1$ $\rightarrow Position(x, y, k, t_1)$	Q1: When did $\langle x \rangle$ send an order to open a position with notional $\langle k \rangle$ ?	Since the trader EGTech at time 1 sends an order to open a position of size 0.3, and it is not true that 1 is a time when the market is closed, then the order of size 0.3 by EGTech is accepted at time 1. Since the order of size 0.3 by EGTech is accepted and the price is 124 at time 1, then EGTech holds a position of size 0.3 and notional 37.2 at time 1.	Q1: When did EGTech send an order to open a position with notional 37.2?
	A1: The order to open that position was sent at time $\langle t_1 \rangle$		A1: The order to open that position was sent at time 1
	...		...

Fig. 2: From plans to fine-tuning corpus, in our running example.

**Quality-driven optimization.** The corpus undergoes a quality check where each pair is filtered according to an NLP-based scoring model in terms of specificity, plausibility, absence of bias, and other user-defined criteria. The filtered-in pairs are enhanced via *NLP paraphrasing* to improve generalization and finally cleansed with additional post-processing procedures (lines 16-22).

**Model fine-tuning.** The refined corpus is injected into an LLM for task- and domain-specific fine-tuning (line 23). In the case of Q&A, the model operates in a *closed-book* approach, that is, it learns to map questions to the corresponding answers without extracting them from an input context, but rather encapsulating the knowledge of the domain into its internal parameters and weights [19]. The resulting specialized model is provided to the user via API, and will act as a natural language interface to the EKG and the ontological reasoning at its foundation in a neurosymbolic fashion.

### 3 Preliminary Validation via Proof-of-Concept

We implemented our fine-tuning pipeline in VADALOG. A full-scale evaluation of our architecture is beyond the scope of this short work. Conversely, in this

section, we propose a conceptual validation of the approach, by briefly showing some executions of the pipeline. We will not consider the text-to-query translation task, as its evaluation would require semantic comparison, which is beyond the scope of this work.

For the proof-of-concept, we made use of a *T5-large* [10] model and considered the same domain as in Example 1. To obtain a dataset that could be visually inspected to informally assess the quality of the textual answers given by an LLM fine-tuned with our pipeline, we performed a kind of *ablation study*. For randomly chosen sets of sample questions, for the NLP tasks of interest, we compared the answers provided by an LLM fine-tuned only with ground facts (*T5-large-ground*) and one fine-tuned with our pipeline (*T5-large-chase*). Both models were trained for 10 epochs and with the same hyperparameters. The fine-tuning corpora and the models are made available [4].

Figure 3 visually reports the comparison. Questions *a* and *b* are the baseline, as they can be answered by facts in  $D$ . Apart from a less refined write-up, the LLMs show the same output behaviour. On the other hand, in questions *c*, *d*, and *f* T5-large-ground is outperformed by T5-large-chase, which succeeds in answering about events related to trader *EGTech*. Actually, the corresponding facts derive from  $\Sigma(D)$ , which is not considered in the ground fine-tuning. Similarly, the answer to question *e* by T5-large-ground is incomplete and only T5-large-chase is able to use the specific domain knowledge from rule 1 of Example 1.

User question		T5-large-ground	T5-large-chase
a.	What is the size of the position that EGTech opens at time 1?	0.3	The size is 0.3
b.	What is the notional of EGTech's position at time 1?	37.2\$	The notional is 37.2\$, equal to price 124\$ $\times$ size 0.3
c.	Why is the order by EGTech at time 1 accepted?	As it is a trade	Because the market at time 1 is not closed
d.	How can EGTech hold a position on the market at time 1 of notional 3\$?	0.3	Since EGTech's position of size 0.3 was accepted and the price at time 1 is 10
e.	How does a trader get their position accepted?	By placing an order	By opening the position at a time in which the market is not closed
f.	How does EGTech profits from its trading activity?	EGTech has a profit of 6.9\$	By opening a position of size 0.3 at time 1, having the position accepted with an initial price of 124\$ and a notional of 37.2\$ at time 1, and then closing it at time 9 with a final price of 147\$, thus having a profit of 6.9\$

Fig. 3: Proof-of-concept for our fine-tuning pipeline.

## 4 Conclusion

According to a recent work [24] appeared in the European Chapter of the Association for Computational Linguistics, pre-trained language models cannot yet perform deductive reasoning: they are still unable to generalize logical rules and, even when rules are provided, LLMs tend to forget previously inferred facts. While no extensive comparison between transformer architectures and reasoning approaches has been conducted yet, our work showed that LLM performance for domain-specific NLP tasks can be visibly improved by producing a fine-tuning corpus as a byproduct of ontological reasoning. We capitalized on our experience in deductive reasoning to offer a first step towards a neuro-symbolic platform for reasoning on enterprise knowledge graphs.

**Acknowledgements.** The work on this paper was partially supported by the Vienna Science and Technology Fund (WWTF) grant VRG18-013.

## References

1. Afrati, F.N., Gergatsoulis, M., Toni, F.: Linearisability on datalog programs. *Theor. Comput. Sci.* **308**(1-3), 199–226 (2003)
2. Agarwal, O., Ge, H., Shakeri, S., Al-Rfou, R.: Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. *arXiv preprint arXiv:2010.12688* (2020)
3. Andrus, B.R., Nasiri, Y., Cui, S., Cullen, B., Fulda, N.: Enhanced story comprehension for large language models through dynamic document-based knowledge graphs. *AAAI* **36**(10), 10436–10444 (2022)
4. Baldazzi, T., Bellomarini, L., Ceri, S., Colombo, A., Gentili, A., Sallinger, E.: Material. <https://bit.ly/44249b5>, accessed: 2023-06-17
5. Bellomarini, L., Benedetto, D., Gottlob, G., Sallinger, E.: Vadalogue: A modern architecture for automated reasoning with large knowledge graphs. *IS* **105** (2022)
6. Bellomarini, L., Fakhoury, D., Gottlob, G., Sallinger, E.: Knowledge graphs and enterprise AI: the promise of an enabling technology. In: *ICDE*. pp. 26–37 (2019)
7. Brown, T., et al.: Language models are few-shot learners. In: *NeurIPS*. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
8. Cali, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.* **14**, 57–83 (2012)
9. Fu, H., Liu, C., Wu, B., Li, F., Tan, J., Sun, J.: Catsql: Towards real world natural language to sql applications. *VLDB* **16**(6), 1534–1547 (2023)
10. Google: T5 large. <https://huggingface.co/t5-large>, accessed: 2023-06-17
11. Krötzsch, M., Thost, V.: Ontologies for knowledge graphs: Breaking the rules. In: *ISWC* (1). LNCS, vol. 9981, pp. 376–392 (2016)
12. Liu, Z., Huang, D., Huang, K., Li, Z., Zhao, J.: Finbert: A pre-trained financial language representation model for financial text mining. In: *IJCAI 2020* (2021)
13. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. *ACM TODS* **4**(4), 455–469 (1979)
14. Mohanta, B.K., Panda, S.S., Jena, D.: An overview of smart contract and use cases in blockchain technology. In: *ICCCNT*. pp. 1–4 (2018)
15. Moiseev, F., Dong, Z., Alfonseca, E., Jaggi, M.: SKILL: Structured knowledge infusion for large language models. In: *ACL 2022*. pp. 1581–1588 (2022)
16. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
17. Rae, J.W., et al.: Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021)
18. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**, 140:1–140:67 (2020)
19. Roberts, A., Raffel, C., Shazeer, N.: How much knowledge can you pack into the parameters of a language model? In: *EMNLP* (1). pp. 5418–5426. *ACL* (2020)
20. Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. In: *NAACL: Tutorials*. pp. 15–18 (2019)
21. Wang, B., et al.: Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942* (2019)
22. Wu, S., Irsoy, O., Lu, S., Dabrowski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D.S., Mann, G.: Bloomberggpt: A large language model for finance. *CoRR abs/2303.17564* (2023)
23. Yin, X., Gromann, D., Rudolph, S.: Neural machine translating from natural language to sparql. *Future Generation Computer Systems* **117**, 510–519 (2021)
24. Yuan, Z., Hu, S., Vulic, I., Korhonen, A., Meng, Z.: Can pretrained language models (yet) reason deductively? In: *EACL*. pp. 1439–1454 (2023)