Termite Italian Text-to-SQL: A CALAMITA Challenge

Federico Ranaldi[1,*,†], Elena Sofia Ruzzetti[1], Dario Onorati[3], Fabio Massimo Zanzotto[1] and Leonardo Ranaldi[1,2]

[1]Human-Centric ART, University of Rome Tor Vergata, Italy.
[2]School of Informatics, University of Edinburgh, UK.
[3]University of Rome La Sapienza, Italy.

Abstract

Relational databases play an important role in business, science, and beyond. However, the operability of relational databases is restricted to users familiar with specific languages such as SQL, which limits the analytical power that they could deliver. Although earlier techniques have been proposed to automatically generate SQL from natural language, such as Text-to-SQL large-scale datasets, they are predominantly built-in English and are automatically constructed using surface web data. This phenomenon limits evaluation and use in settings beyond English and also limits fair assessment, given the origin of the datasets, as the data may have already been seen in pre-training corpora.

In this work, we introduce Termite, which is a definitely unseen resource for evaluating Text-to-SQL in Italian. Specifically, we transfer evaluation pipelines beyond English, proposing novel, definitely unseen resources that avoid *data-contamination* phenomena while assessing the ability of models to perform Text-to-SQL tasks when natural language queries are written in Italian. We establish an evaluation grid based on execution accuracy. Our code and datasets are available at link.

Keywords

Text-to-SQL, Italian LLMs, CALAMITA, CLiC-it

1. Introduction

The Text-to-SQL is an important NLP task, which maps input questions to meaningful and executable SQL queries, enabling users to interact with databases in a more intuitive and user-friendly way. Despite the substantial number of state-of-the-art systems [1, 2, 3] and benchmarks [4, 5, 6] for Text-to-SQL, most of them are in English and this limits the operability to non-English users.

Dou et al. [5] proposed extensions beyond English Spider [4]. This still highlights significant limitations because the resources in specific languages were generated from automatic translations for a few languages. On the other hand, publicly released resources could be translated and adapted to the Text-to-SQL task, but these could be the panacea of contamination as they are often publicly available (e.g., Kaggle or Wikipedia as in the case of [4, 7]). Indeed, portions of these resources are included in the huge corpora employed to conduct the pre-training phases of large language models (LLM), i.e., the data-contamination phenomenon [8, 9, 10, 11, 12].

To tackle these problems, in the context of CALAMTIA [13] we propose Termite (Text-to-SQL Repository Made Invisible to Engines), a novel Text-to-SQL resource created and conceived for the Italian. We aim to reduce the possibility of increased performance due to data contamination while proposing a suitable resource for a specific language. In fact, in contrast to native English benchmark translation methods, Termite is designed to be used as an assessment pipeline, ensuring that it remains a resource not exposed to search engines as it is locked by an encryption key distributed with the dataset, reducing accidentally inclusion in a new commercial or search LLMs training set.

Termite is structurally designed to resemble Spider. However, it complements Spider's extensions into other languages by proposing a series of databases originally hand-crafted in Italian. Specifically, part of the Termite content comes from a thorough reworking of databases initially designed by students from the University of Rome Tor Vergata. This aspect, enriched by the invisibility to search engines, makes Termite a valuable resource for evaluating models on a practical and theoretically significant task.

Moreover, evaluating Text-to-SQL models in languages beyond English is essential for broadening their practical use and understanding of their linguistic behavior. Assessing how these models handle the same problem presented in different languages is critical for gaining insights into their adaptability and consistency across multilingual contexts [9, 14, 15, 16].

2. Background

In this section, we provide a formal problem definition of Text-to-SQL (§2.1), addressing typical aspects that define it beyond a natural language understanding or code generation problem. Then, we discuss the potential impact

CLiC-it 2024: Tenth Italian Conference on Computational Linguistics, Dec 04 — 06, 2024, Pisa, Italy

federico.ranaldi99@gmail.com (F. Ranaldi)

of data contamination on this task and how our TERMITE serves as a measure against it, outlining several considerations that mitigate contamination risks (§2.2). Finally, in §2.3 we introduce the challenges that leverage our contribution through the TERMITE resource.

2.1. The Task

Text-to-SQL is a fundamental task within Natural Language Processing (NLP) that involves not only understanding natural language queries and generating corresponding SQL code, but also establishing a mapping between data expressed in natural language and data represented within the database schema. This requires the model to accurately link natural language terms with database structures such as tables, columns, and values, making it a more complex challenge than simple code generation or natural language understanding.

This task is crucial in making relational database interactions more accessible to users who may not be familiar with SQL syntax. The foundational work was based on rule-based and heuristic approaches [1], *(et. alia)*. The actual automatic processing of Text-to-SQL pipelines became meaningful with the advent of neural network-based approaches. The shift towards neural models was facilitated by the introduction of resources such as Spider [4] and the more recent [17], which delivered various and complex natural language to SQL demonstrations.

The most recent advancements in Text-to-SQL involve the use of Large Language Models (LLMs), which have demonstrated remarkable capabilities in handling various tasks without needing specific pretraining or fine-tuning tailored to each task.

Gao et al. [18] and Pourreza and Rafiei [3] shown that GPTs are effective Text-to-SQL coders on Spider, widely acknowledged as an effective benchmark for assessing performance in this specific task

On the same dataset, approaches that deconstruct the problem in smaller ones via in-context learning are even actually examined [3].

The emergence of LLMs as a key paradigm for the Text-to-SQL task has also led to a more in-depth study of various prompt engineering methods. These efforts aim to understand what best enhances a model's performance in text-to-SQL translation. In [19], the performance of the GPT family is evaluated across different prompt scenarios, which vary based on how much information about the database is provided to the model for the translation process. Results show that providing a specific set of additional information significantly improves the model's ability to generate accurate SQL queries [19].

This last aspect enlights how LLMs appear to be behaviourally influenced by both the in-context prompt [20] and the text used during the pre-training [11]. Consequently, if LLMs perform better on tasks with data that were already seen during the pre-training phase, we would face an issue of data contamination.

2.2. Data Contamination in Modern Benchmarks

Data contamination is an increasingly recognized challenge in the field of machine learning, with a growing number of studies dedicated to its investigation. Several recent studies such as [21] and [22] have explored the issue of data contamination, proposing a comprehensive taxonomy of methods to detect and address it. Due to its nature, the text-to-SQL task is susceptible to overestimation issues, particularly related to data contamination. Therefore, a good practice when evaluating a model on this task is to ensure that there is no overlap between the test data and the pre-training data. On the other hand, this becomes challenging when dealing with closed-source models, where there is no clear knowledge of the pre-training data, such as in the case of the GPT family [23].

Hence, taking inspiration from Golchin and Surdeanu [24] and Deng et al. [25] who treated the issue of Data Contamination in closed-source models, Ranaldi et al. [12] proposed a novel method for detecting Data Contamination applied to text-to-SQL. This consists in carefully comparing the model's performance on a novel test set (such as TERMITE) with that on a well-known test set (such as Spider), whose content is suspected to have been exposed to the model's pre-training data. The results showed that GPT models exhibit a drop in performance on TERMITE compared to Spider. Furthermore, it was observed that even perturbing Spider by removing information from the dump provided with the prompt had no significant impact on performance. The study of contaminating test sets continues to expand into other tasks, to the extent that an index of contaminated datasets [26] has been established.

2.3. TERMITE

Our contribution complements [12] in particular by introducing TERMITE. We aim to provide an Italian text-to-SQL dataset and a tool for analysing the contamination of Spider data for LLMs. Indeed, the structural complexity of TERMITE mirrors that of the Spider test set. Moreover, to prevent data contamination from compromising its usefulness, it is freely accessible, but its content is not provided in a fully transparent form.

In the following sections, we describe the composition of TERMITE in detail and provide a basic evaluation to facilitate usability and reproducibility. In addition, to encourage usability, we share the resources and code.

3. Dataset

Our main intent is to provide an evaluation resource for Text-to-SQL on data that is definitely unknown and, therefore, not present in well-known pre-training corpora. However, since several robust evaluation pipelines exist in state of the art, the first step is understanding their structure and operation. Therefore, beyond the de-facto standards resources (§3.1), we introduce our TERMITE conceived as a novel unseen Italian resource (§3.2).

3.1. Spider: Characteristics and Content

Among the best-known Text-to-SQL resources is Spider [4]. This resource is the de-facto standard for training and testing systems on the Text-to-SQL task.

Spider appears as a collection of databases and associated sets of pairs of natural language (NL) questions and the corresponding SQL translations. Databases are structurally represented inside the dataset in the form of SQL dumps, which include the CREATE TABLE operations and a limited number of INSERT DATA operations for each table.

NL questions are organized into four difficulty levels: EASY, MEDIUM, HARD, and EXTRA-HARD. For the definition of the hardness level, we refer to the categorization originally made in Spider [4]. The difficulty of an NL question is assessed by considering the corresponding SQL query. Hence, the difficulty is correlated with the number and kind of operations that the gold query contains: the presence of JOIN operations, aggregation, and WHERE conditions contribute to the hardness of the query. EASY queries do not involve more than one table. MEDIUM and HARD queries span multiple tables: MEDIUM queries contain only a JOIN or aggregation operation whereas HARD queries are more complex both in terms of number of JOIN and aggregations. Finally, EXTRA-HARD queries may contain nested queries, and other operators like UNION and INTERSECT [1].

3.2. TERMITE: a Text-to-SQL Repository Made Invisible to Engines

The driving idea for proposing a novel resource for the Text-to-SQL task is to reduce the possibility of boosting performance due to data contamination. Indeed, publicly available datasets are not suitable for this purpose. Even though novel datasets are made available, they are built from publicly open-access resources such as Kaggle or Wikipedia (this is the case for recently developed datasets like BIRD [7] or Spider itself). Hence, these do not guarantee that they are as new as required. The same issue may also be faced for hidden test sets. Moreover, since freely available datasets are easily accessed and tracked by engines, they are at risk of being contaminated in the near future if they are not already contaminated.

To address these challenges, we propose TERMITE[2]. TERMITE aims to be a permanently fresh dataset. TERMITE will be invisible to search engines since it is locked under an encryption key delivered along the resource. This trick will reduce the accidental inclusion in a novel training set for commercial or research GPTs.

Hence, by following characteristics of Spider, TERMITE contains hand-crafted databases in different domains. Each database has a balanced set of NL-SQL query pairs: we defined an average of 5 queries per hardness-level. The entire dataset was designed to be comparable to the Spider Validation Set, not only in terms of database characteristics such as size and table count (Table 1) but also in terms of query difficulty, which was measured using the same definition provided by Spider. Moreover, as in Spider, during the construction of TERMITE, we took care to write unambiguous, direct NL questions that can be solved by a model relying only on its linguistic proficiency and an analysis of the schema, with no external knowledge needed. The style adopted in the NL questions is plain and colloquial in line with the style of Spider's NL questions. Spider and TERMITE are also comparable in terms of number of tables and columns in each dataset. We curated the column names to make them similar to the ones in Spider, using a similar percentage of abbreviations and compound names (see Table 1). This equivalence will be crucial to limit the influence of the dataset itself on the following evaluations and will be further explored in Section 4.2.

However, there is a significant and fundamental difference between the two datasets, as the TERMITE is not openly available on the web or easily retrievable nor built on pre-existing openly available resources.

This aspect is crucial because the way it is made available certainly reduces the risk of falling into the LM contamination index ([26]).

3.3. Comparing Hardness of TERMITE vs. Spider

When introducing a new dataset for benchmarking a particular task, it is important to ensure it aligns with the established and commonly used datasets within the community to maintain consistency and comparability.

Our TERMITE is designed to resemble Spider in terms of measurable aspects, like the number of columns and tables per database, as well as the lexicon used in the schema definition. However, it remains difficult to quantify via some simple statistics how hard it is to understand

[1] More details are available on the official Spider repository

[2] The repository is available here under GPL-3.0 license. To access, use the password "youshallnotpass".

	Dataset	
	Spider	Termite
#DB	20	10
avg #TABLES per DB	4.2	4.0
avg #COLUMNS per TABLE	5.46	5.56
#QUERY	1035	202
avg #QUERY per DB	51.75	20.2
avg #FK/#COLUMNS per DB	0.16	0.13
avg #Compound/#COLUMNS per DB	0.63	0.51
avg #Abbr/#COLUMNS per DB	0.10	0.12

Table 1
Spider and fact sheet. TERMITE is designed to be comparable to the validation set of Spider.

how to translate a natural language question into an SQL statement.

To compare hardness of TERMITE and Spider, we adopted a human-centered definition: if humans can translate questions into an SQL queries on both Spider and TERMITE with the same level of challenge, then it means that their hardness, at least for a SQL-proficient human annotator, is the same.

Therefore, ten annotators were asked to judge the equivalence in terms of hardness of the SQL translations that compose Spider and TERMITE by examining a random sample of queries of both datasets.

To measure the hardness of the two datasets, we designed a simple test. Given a Entity-Relationship schema of a database and a question in natural language, each annotator is asked to choose among three options the correct translation in SQL of the question. Appendix **??** presents details on the construction of the test.

On both Spider and TERMITE, taking as join annotation the answer chosen by the majority of annotators leads to almost perfect classification (0.975 accuracy on Spider and maximum accuracy on TERMITE). The average accuracy per annotator is $0.91(\pm 0.05)$ on Spider and $0.94(\pm 0.07)$ on TERMITE. Moreover, Fleiss's Kappa coefficients are rather high (0.79 and 0.85 respectively) for both Spider and TERMITE. Hence, we can conclude that humans do not find one dataset more difficult than the other. The two datasets can then be considered equivalent in terms of the hardness of translations.

4. Methods

Current evaluation pipelines exploit the behaviour of models by defining robust prompting strategies since the generations delivered by these are strongly correlated to the in-context structures [19].

Thus, in §4.1, we introduce the technique for the Text-to-SQL task as the suggested evaluation metric for an initial exploration of TERMITE. Furthermore, in §4.2, we define *Execution Accuracy* as the evaluation metric of choice for evaluating the model, as it offers a practical method for determining the correctness of SQL query generation within this framework.

4.1. Prompting LLMs in Italian for Text-to-SQL Translation

Given instructions in natural language, LLMs can translate the request into code (i.e., SQL queries) to answer the given request. Specifically, models for generating text have undergone training to process both natural language and code. As a result of the inputs they receive, these models produce text-based outputs. For this reason, it is possible to frame the Text-to-SQL as a translation task: given a dump for a database and a query in natural language, the model is asked to translate the latter in the corresponding SQL query, referring to tables and columns into the considered database. The desiderata is an executable query, semantically equivalent to a gold human-generated query. In the next paragraphs, we first describe how GPT-3.5 (`gpt-3.5-turbo`) is prompted in order to obtain the translations .

Text-to-SQL as a Translation Task OpenAI API's enable to interrogate a model in a multi-turn conversation format: chat models receive a series of messages as input and generate a message as output. We test the ability of GPT-3.5 on the Text-to-SQL task by framing each translation from natural language to SQL as a separate conversation.

The proposed approach, aimed at analysing the model's in-context learning abilities in zero-shot scenarios, is very similar to "Code Representation" [19] and has been specifically tested in Italian [9].

In particular, the first message of a target database gives the model the dump of the database. In each dump, information about the database's tables is provided by the CREATE TABLE statements. In the CREATE instructions, the constraints of the primary and foreign keys are also encoded. In addition, some realistic data to fill the tables are provided by INSERT instructions. Given the dump, the model answers by producing an interpretation of the dump. Typically, this model response contains an explanation of the dump's contents. For example, considering the database `bowling` in Termite dataset, the first messages in the conversation are the following:

user: `Considera il seguente database: CREATE TABLE "pista" [...]; CREATE TABLE "giocatori" [...];`

GPT-3.5: `Questo database rappresenta una struttura per la gestione di un centro di bowling...`

Then, given the dump and the model's interpretation of it, a message containing the natural language question to be translated is sent. In particular, the selected prompt ensures that the model translates natural language questions into SQL queries with a limited amount of text that is not SQL. These steps are repeated for each question separately to obtain translations independently. However, to ensure that the model's understanding of each database is comparable across all questions, the database dump and the same interpretation initially produced by the model are sent as context, in the form of preceding messages, before each translation is requested. Hence, building from the previous example, a conversation to translate a question on the `bowling` database would be completed by the following messages:

user: `Traduci in SQL la seguente query. Rispondi usando esclusivamente linguaggio SQL. Conta il numero di giocatori per partita.`
GPT-3.5: `SELECT ora_inizio,tenuta_il,id_pista, COUNT(*) FROM 'partita' GROUP BY ora_inizio,tenuta_il,id_pista;`

4.2. Measuring Hardness of queries in Spider and Termite

We need to ensure that Spider and Termite are hardness comparable. Termite is designed with a similar annotation protocol; however, a similarity in terms of the hardness of the natural language questions used is hard to quantify. For this reason, we asked 10 SQL-proficient annotators to perform a simple yet effective test to measure how difficult it is for them to translate questions both from Spider and from Termite. The main idea is that if they can translate both Spider and Termite questions with the same accuracy level, then the challenge level is similar on both datasets.

In particular, given an E-R database schema and a natural language utterance, each test question asks the annotator to choose from three SQL query options that satisfy the request. All three options are syntactically correct SQL queries, but the incorrect answers are semantically different from the correct ones. The authors designed the first incorrect option, perturbing the correct answer by removing or replacing some operations or retrieved columns and changing the field and table names with non-matching ones. The second incorrect answer is another query extracted from the same dataset as the correct one. The selected query is the most similar under the Bag of Words assumption concerning the correct one. To retrieve this third option, the similarity of two queries is measured via the cosine similarity of their BOW vector representations.

The complete test is composed of 20 randomly selected queries from each dataset, Hence, the resulting 40 questions are shared to 10 SQL-proficient annotators: 60% of them are Computer Science Master students, the remaining are already graduated. Five annotators work in a field that requires daily use of the SQL query language. Finally, we divided the test into two trials of 20 queries each. We administered it to the annotators at two different times to limit errors due to gradual loss of concentration.

Our approach is completely zero-shot to minimize the effect that the prompt itself–rather than data contamination–can have on performance. Once the translation process is completed, the SQL code produced by the model is retrieved to evaluate whether or not the generated query satisfies the natural language query.

Execution Accuracy: the Evaluation Metric The evaluation metric adopted is execution accuracy introduced by Yu et al. [4], which assesses the correctness of the generated SQL query by executing it against the database and comparing the result with the expected output.

The Execution Accuracy (EA) can be formally defined as follows:

Let q represent the gold query and g represent the generated query. The execution accuracy compares the execution results of g and q on a database D.

$$EA(g, q, D) = \begin{cases} 1 & \text{if } g(D) = q(D) \\ 0 & \text{if } g(D) \neq q(D) \end{cases}$$

where $g(D)$ and $q(D)$ represent the outputs of the queries on D. Execution accuracy is 1 if the results are the same and 0 otherwise.

In case of syntactic errors in the generated SQL query, it is considered definitively incorrect, as adherence to SQL grammar is part of the model's evaluation.

The execution accuracy metric is prone to false positives, as two different queries can return the same output under specific database record configurations. For this reason, in [12], the Test Suite Accuracy metric is adopted. Test Suite Accuracy, introduced in Zhong et al. [27], essentially involves performing execution accuracy on the same query across many randomly generated database record configurations called Test Suite.

In this paper, we propose EA as an evaluation metric because the way queries and database records are designed in Termite aims to minimize the occurrence of false positives. Additionally, to encourage experimentation with Termite, we recommend initially employing simple and computationally inexpensive evaluation metrics, in contrast to Test Suite Accuracy. Moreover, we suggest disregarding the query difficulty evaluation metric proposed by [4].

Hence, in link is available, an automated script evaluates generated SQL queries using Execution Accuracy as the metric. It can be run locally as it is a lightweight program that executes queries on an SQL server and processes the output as our metric requires.

5. Experiments

Our TERMITE aims to extend the Text-to-SQL evaluation pipeline to Italian while preserving data integrity and thus preventing possible contamination. To prove its operability, we propose a baseline assessment in §5.1 and discuss the obtained results in §5.2.

5.1. Experimental Setup

We systematically evaluated GPT-3.5 (gpt-3.5-turbo-16k) performance on the TERMITE dataset for the Text-to-SQL task. We employed the API to generate SQL translations for each query in the dataset. To ensure consistency in the results, we set the temperature parameter to 1, allowing for greater flexibility and diversity in the model's output. For each natural language query, a translation request was sent to the model. The generated SQL query was then saved and subsequently processed according to the aforementioned metric (§4.2).

Database Name	EA_SCORE (%)	Queries
bowling	50.79	24
centri	56.25	19
coronavirus	40.00	20
farma	62.50	20
farmacia	50.00	20
galleria	69.15	23
hackathon	46.25	19
pratica	50.11	22
recensioni	20.00	18
voli	56.25	17

Table 2
Execution Accuracy (**EA_SCORE (%)**) achieved by GPT-3.5 and Number of Queries for each Database

5.2. Baseline Results

The results achieved in the baseline assessment reveal the intrinsic challenges of the text-to-SQL task performance. In fact, Table 2 reports the Execution Accuracy percentages (**EA_SCORE (%)**) achieved by GPT-3.5 on each of the 10 datasets that compose our TERMITE. It can be observed that an acceptable accuracy, significantly exceeding 50%, is only seen for the "farma" and "galleria" databases, where 69% and 62% accuracy were achieved, respectively.

6. Limitations & Future Works

The idea of TERMITE is to propose a new resource conceived and realized for the Italian language. During the discussion of the contribution, we introduced the underlying motivations that support our choices regarding encryption and baseline evaluations.

However, we plan to extend our contribution to languages beyond Italian in future developments. We also aim to propose efficient alignment techniques to enable smaller models to cope with more demanding tasks such as text-to-SQL by adopting teacher-student alignment techniques [28, 29].

7. Conclusions

We have introduced TERMITE, a resource that, to the best of our knowledge, is unique in that the databases and queries were natively conceived in Italian. Its structural alignment with well-known datasets like Spider makes it a solid benchmarking tool for analysing Text-to-SQL results when the test set languages differ.

Additionally, its uniqueness lies in the fact that it is not publicly accessible by search engines, making it less exposed to the increasingly prominent issue of data contamination, particularly when dealing with closed-source large language models.

Extending TERMITE to include queries where the complexity is not only driven by the SQL query itself but also by tasks such as commonsense and arithmetic reasoning would further enrich the dataset. This is in line with approaches like those seen in Archer [30], which address these additional challenges.

Acknowledgments

We would like to express our gratitude to the Human-Centric Art team for their valuable collaboration in the creation of the TERMITE dataset. Special thanks go to the annotators whose work was essential in affirming the comparability between TERMITE and Spider. Finally we extend our appreciation to the Computer Science's students of the University of Rome Tor Vergata for providing the original hand-crafted databases, which were subsequently the subject of extensive reworking and refinement.

References

[1] A. Giordani, A. Moschitti, Translating questions to SQL queries with generative parsers discriminatively reranked, in: M. Kay, C. Boitet (Eds.), Proceedings of COLING 2012: Posters, The COLING 2012 Organizing Committee, Mumbai, India, 2012, pp. 401–410. URL: https://aclanthology.org/C12-2040.

[2] T. Scholak, N. Schucher, D. Bahdanau, PICARD: Parsing incrementally for constrained auto-regressive decoding from language models, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 9895–9901. URL: https://aclanthology.org/2021.emnlp-main.779. doi:10.18653/v1/2021.emnlp-main.779.

[3] M. Pourreza, D. Rafiei, DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction, in: Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL: https://openreview.net/forum?id=p53QDxSIc5.

[4] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, D. Radev, Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task, in: E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 3911–3921. URL: https://aclanthology.org/D18-1425. doi:10.18653/v1/D18-1425.

[5] L. Dou, Y. Gao, M. Pan, D. Wang, W. Che, D. Zhan, J.-G. Lou, Multispider: Towards benchmarking multilingual text-to-sql semantic parsing, 2022. URL: https://arxiv.org/abs/2212.13492. arXiv:2212.13492.

[6] J. Li, B. Hui, G. QU, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. Chang, F. Huang, R. Cheng, Y. Li, Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs, in: Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023. URL: https://openreview.net/forum?id=dI4wzAE6uV.

[7] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, Y. Li, Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023. arXiv:2305.03111.

[8] I. Magar, R. Schwartz, Data contamination: From memorization to exploitation, 2022. arXiv:2203.08242.

[9] L. R. D. V. C. G. A. F. R. R. F. M. Z. Federico Ranaldi, Elena Sofia Ruzzetti, Prompting llms in italian language for text-to-sql translation, in: Proceedings of CLIC 2023, Location, 2023.

[10] L. Ranaldi, A. Nourbakhsh, E. S. Ruzzetti, A. Patrizi, D. Onorati, M. Mastromattei, F. Fallucchi, F. M. Zanzotto, The dark side of the language: Pretrained transformers in the DarkNet, in: R. Mitkov, G. Angelova (Eds.), Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, INCOMA Ltd., Shoumen, Bulgaria, Varna, Bulgaria, 2023, pp. 949–960. URL: https://aclanthology.org/2023.ranlp-1.102.

[11] L. Ranaldi, E. S. Ruzzetti, F. M. Zanzotto, PreCog: Exploring the relation between memorization and performance in pre-trained language models, in: R. Mitkov, G. Angelova (Eds.), Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, INCOMA Ltd., Shoumen, Bulgaria, Varna, Bulgaria, 2023, pp. 961–967. URL: https://aclanthology.org/2023.ranlp-1.103.

[12] F. Ranaldi, E. S. Ruzzetti, D. Onorati, L. Ranaldi, C. Giannone, A. Favalli, R. Romagnoli, F. M. Zanzotto, Investigating the impact of data contamination of large language models in text-to-SQL translation, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics ACL 2024, Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 2024, pp. 13909–13920. URL: https://aclanthology.org/2024.findings-acl.827.

[13] G. Attanasio, P. Basile, F. Borazio, D. Croce, M. Francis, J. Gili, E. Musacchio, M. Nissim, V. Patti, M. Rinaldi, D. Scalena, CALAMITA: Challenge the Abilities of LAnguage Models in ITAlian, in: Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024), Pisa, Italy, December 4 - December 6, 2024, CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[14] L. Ranaldi, G. Pucci, Does the English matter? elicit cross-lingual abilities of large language models, in: D. Ataman (Ed.), Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL), Association for Computational Linguistics, Singapore, 2023, pp. 173–183. URL: https://aclanthology.org/2023.mrl-1.14. doi:10.18653/v1/2023.mrl-1.14.

[15] L. Ranaldi, G. Pucci, F. Ranaldi, E. S. Ruzzetti, F. M. Zanzotto, A tree-of-thoughts to broaden multi-step reasoning across languages, in: K. Duh, H. Gomez, S. Bethard (Eds.), Findings of the Associ-

ation for Computational Linguistics: NAACL 2024, Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 1229–1241. URL: https://aclanthology.org/2024.findings-naacl.78. doi:10.18653/v1/2024.findings-naacl.78.

[16] L. Ranaldi, G. Pucci, A. Freitas, Empowering cross-lingual abilities of instruction-tuned large language models by translation-following demonstrations, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics ACL 2024, Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 2024, pp. 7961–7973. URL: https://aclanthology.org/2024.findings-acl.473.

[17] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, Y. Li, Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023. URL: https://arxiv.org/abs/2305.03111. arXiv:2305.03111.

[18] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, J. Zhou, Text-to-sql empowered by large language models: A benchmark evaluation, 2023. arXiv:2308.15363.

[19] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, J. Zhou, Text-to-sql empowered by large language models: A benchmark evaluation, 2023. URL: https://arxiv.org/abs/2308.15363. arXiv:2308.15363.

[20] L. Ranaldi, G. Pucci, When large language models contradict humans? large language models' sycophantic behaviour, 2024. URL: https://arxiv.org/abs/2311.09410. arXiv:2311.09410.

[21] C. Deng, Y. Zhao, Y. Heng, Y. Li, J. Cao, X. Tang, A. Cohan, Unveiling the spectrum of data contamination in language model: A survey from detection to remediation, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics ACL 2024, Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 2024, pp. 16078–16092. URL: https://aclanthology.org/2024.findings-acl.951.

[22] M. Ravaut, B. Ding, F. Jiao, H. Chen, X. Li, R. Zhao, C. Qin, C. Xiong, S. Joty, How much are large language models contaminated? a comprehensive survey and the llmsanitize library, 2024. URL: https://arxiv.org/abs/2404.00699. arXiv:2404.00699.

[23] OpenAI, Gpt's family, 2023. URL: https://platform.openai.com/docs/models.

[24] S. Golchin, M. Surdeanu, Time travel in llms: Tracing data contamination in large language models, 2024. URL: https://arxiv.org/abs/2308.08493. arXiv:2308.08493.

[25] C. Deng, Y. Zhao, X. Tang, M. Gerstein, A. Cohan, Investigating data contamination in modern benchmarks for large language models, 2024. URL: https://arxiv.org/abs/2311.09783. arXiv:2311.09783.

[26] Contaminated datasets index, https://hitz-zentroa.github.io/lm-contamination/, 2023. Accessed: 2024-09-23.

[27] R. Zhong, T. Yu, D. Klein, Semantic evaluation for text-to-SQL with distilled test suites, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 396–411. URL: https://aclanthology.org/2020.emnlp-main.29. doi:10.18653/v1/2020.emnlp-main.29.

[28] L. Ranaldi, A. Freitas, Aligning large and small language models via chain-of-thought reasoning, in: Y. Graham, M. Purver (Eds.), Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, St. Julian's, Malta, 2024, pp. 1812–1827. URL: https://aclanthology.org/2024.eacl-long.109.

[29] L. Ranaldi, G. Pucci, F. M. Zanzotto, Modeling easiness for training transformers with curriculum learning, in: R. Mitkov, G. Angelova (Eds.), Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, INCOMA Ltd., Shoumen, Bulgaria, Varna, Bulgaria, 2023, pp. 937–948. URL: https://aclanthology.org/2023.ranlp-1.101.

[30] D. Zheng, M. Lapata, J. Z. Pan, Archer: A human-labeled text-to-sql dataset with arithmetic, commonsense and hypothetical reasoning, 2024. URL: https://arxiv.org/abs/2402.12554. arXiv:2402.12554.