# Investigating the Impact of Data Contamination of Large Language Models in Text-to-SQL Translation

**Anonymous ACL submission**

## Abstract

Understanding textual description to generate code seems to be an achieved capability of instruction-following Large Language Models (LLMs) in zero-shot scenario. However, there is a severe possibility that this translation ability may be influenced by having seen target textual descriptions and the related code. This effect is known as *Data Contamination*.

In this study, we investigate the impact of Data Contamination on the performance of GPT-3.5 in the Text-to-SQL code-generating tasks. Hence, we introduce a novel method to detect Data Contamination in GPTs and examine GPT-3.5's Text-to-SQL performances using the known Spider Dataset and our new unfamiliar dataset Termite. Furthermore, we analyze GPT-3.5's efficacy on databases with modified information via an adversarial table disconnection (ATD) approach, complicating Text-to-SQL tasks by removing structural pieces of information from the database. Our results indicate a significant performance drop in GPT-3.5 on the unfamiliar Termite dataset, even with ATD modifications, highlighting the effect of Data Contamination on LLMs in Text-to-SQL translation tasks.

## 1 Introduction

Large Language Models (LLMs) have largely demonstrated their ability to understand the semantics of text descriptions for generating code for a variety of programming languages (Wang et al., 2023; Zhang et al., 2023b; Chen et al., 2023). This capability showcases an impressive understanding of the syntax and semantics of both natural language and programming languages. Beyond capturing and mastering the grammar of programming languages governed by a finite set of rules, these models are also proficient in semantically interpreting natural language descriptions and then translating them into a code snippet (Yuan et al., 2023).

LLMs are successful code generators even for the challenging generation of SQL queries from textual description (Rajkumar et al., 2022; Gao et al., 2023; Pourreza and Rafiei, 2023). Indeed, query languages like SQL pose additional challenges as code snippets depend on underlying databases. Then, it is crucial to thoroughly understand the specific database structure with which the SQL code will interact. This is because the effectiveness and accuracy of the generated SQL code largely depend on how well it aligns with the database's schema, constraints, and data types.

However, the evaluation of the LLMs' capability to generate SQL may be conflated by *data contamination* (Magar and Schwartz, 2022; Ranaldi et al., 2023). Data Contamination refers to the situation where a model may have been exposed to, or trained on, parts of the dataset that are later used for its evaluation. Indeed, many datasets that are used to evaluate LLMs' ability to generate SQL code from text, like Spider (Yu et al., 2019), may be included in the pre-training material of state-of-the-art instruction following LLMs such as OpenAI GPTs (OpenAI, 2023b).

In this paper, we aim to unravel the complicated question of whether memorization is responsible for text-to-SQL code generation capabilities of LLMs. More specifically, we focus on the following research questions:

- *RQ1*: Is it possible to determine data contamination by solely analyzing the inputs and outputs of existing LLMs?
- *RQ2*: Do recent GPTs excel in Text-to-SQL tasks in a zero-shot setting both on potentially leaked data and totally unseen one?
- *RQ3*: Is data contamination affecting the accuracy and reliability of an existing GPT in Text-to-SQL tasks?

Hence, we propose Termite - a fresh dataset for evaluating the task of text-to-SQL - in contrast to the widely spread Spider (Yu et al., 2019), which has possibly been used to pre-train LLMs such as commercial GPTs. By comparing Termite and Spider, we propose a measure to determine data contamination in LLMs for tasks of text-to-SQL (*RQ1*). We then experimented with GPT-3.5 (OpenAI, 2023a), comparing the results in the text-to-SQL task obtained on Termite and on Spider (*RQ2*). Then, we tested GPT-3.5 by removing structural information from the databases and demonstrate that the model is more resistant to this adversarial input perturbation on leaked data than unseen one (*RQ3*).

Our results show that not only does GPT exhibit clear knowledge about Spider, but this also leads to an overestimation of the model's performance in Text-to-SQL tasks in zero-shot scenarios.

## 2 Background

Text-to-SQL represents a cutting-edge task in Natural Language Processing (NLP), where the goal is to translate user queries expressed in natural language into SQL queries that can be executed on a database. This task is crucial in making database interactions more accessible to users who may not be familiar with SQL syntax.

In the early stages of Text-to-SQL research, the focus was primarily on rule-based and heuristic approaches (Warren and Pereira, 1982; Giordani and Moschitti, 2012).

The landscape of Text-to-SQL began to evolve significantly with the advent of neural network-based approaches (Yin et al., 2016; Xu et al., 2017). The shift towards neural models was facilitated by the creation and availability of large, specialized datasets such as Spider (Yu et al., 2019), which provided diverse and complex natural language to SQL examples.

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 the need for specific pretraining or fine-tuning tailored to each task. Gao et al. (2023) and Pourreza and Rafiei (2023) have 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 involve deconstructing the problem in smaller ones via in-context learning (Pourreza and Rafiei, 2023; Zhang et al., 2023a) are also effectively explored. However, while LLMs performances have been explored in detail, it remains unclear whether the results may be conflated by data contamination. Indeed, if it turns out that LLMs perform better on tasks with data that have already been seen during the pretraining phase, we would be facing an issue of data contamination.

Data Contamination is a relatively new and tricky problem in the field of machine learning, and there are only a few studies that have addressed it. Magar and Schwartz (2022) attempts to examine how accuracies achieved by BERT (Devlin et al., 2019) on certain tasks vary from previously seen data e and unseen when the training set contains a portion of the test set. Recently, the effect of data contamination on BERT and GPT-2 performance on NLU datasets has been discussed by training a model from scratch and measuring the difference in performance over seen and unseen data (Ranaldi et al., 2023; Jiang et al., 2024). This line of research is complementary to the one we are proposing in this paper. In fact, experimenting with very large language models is still challenging. These technical limitations lead to experiments that involve training on smaller networks, which resemble the original one but are trained on fewer data and have fewer parameters (as done both in Ranaldi et al. (2023) and Jiang et al. (2024)). Hence, a different strategy is needed to address data contamination in closed models. Like Carlini et al. (2021), we are trying to extract pretraining data information from LLMs, while no accurate information on pretraining data is available. The concern about Data Contamination is growing along with the popularity of closed LLMs (Sainz et al., 2023) and some efforts – like the Contamination Index[1] – are made to trace back training data.

Our work contributes to understanding how data contamination – also called "Memorization" by Magar and Schwartz (2022) – plays a role in Text-to-SQL tasks on black-box models, without any further training step. In particular, we will test GPT-3.5 on a well-known dataset –Spider– and compare the performance it achieves on this dataset to that obtained on a new, totally unseen one. Thus, taking inspiration from very recent work dealing with Data Contamination in GPT-3.5 (Golchin and Surdeanu, 2023; Chang et al., 2023; Deng et al.,

[1] https://hitz-zentroa.github.io/lm-contamination/

2023), we will design specific tasks to assess the presence of data contamination and its effect on model performance.

## 3 Text-to-SQL Datasets

To explore whether some test dataset has been leaked during training (*RQ1*), meausure GPTs performance in Text-to-SQL tasks both on potentially known and unknown data (*RQ2*) and whether data contamination is responsible for this performance (*RQ3*), the first step is to introduce the used datasets. In addition to the de-facto standard of Spider (Yu et al., 2019) (described in Sec. 3.1), we propose Termite, a Text-to-SQL dataset conceived to be a new and never-seen resource (introduced in Sec. 3.2). Therefore, Termite lowers the probability of performance boost due to data contamination.

### 3.1 Spider: Characteristics and Content

Spider (Yu et al., 2018) is the de-facto standard for training and testing systems on the Text-to-SQL task. Then, this dataset is used in our study on GPTs and it is used to inspire the construction of our Termite - Text-to-SQL Repository Made Invisible to Engines.

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`. 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 contributes 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` [2].

Since our aim is to evaluate the GPT capabilities in zero-shot scenario, we only considered the validation split of Spider. This portion of the dataset consists of 20 databases and 1,035 pairs of NL-SQL queries distributed on the four difficulty categories (see Tab. 1).

### 3.2 Termite: a Text-to-SQL Repository Made Invisible to Engines

The driving idea for proposing a new dataset for the Text-to-SQL task is to reduce the possibility of boosting performance due to data contamination. Indeed, publicly available datasets are generally not suitable for this purpose. Novel datasets made available, for example, after training a model that one wishes to test, but which are built from publicly available resources such as Kaggle or Wikipedia (this is the case for recently developed datasets like BIRD (Li et al., 2023) or Spider itself), do not guarantee that they are as new as required. The same issue may also be faced for "hidden" test sets. Also, since freely available datasets are easily accessed and tracked by engines, if not already contaminated, they are at risk of being contaminated in the near future. To address these challenges, we propose Termite[3]. Termite aims to be a permanently fresh dataset. Indeed, our dataset will be invisible to search engines since it is locked under an encryption key that is distributed with the dataset. This trick will reduce the accidental inclusion in a novel training set for commercial or research GPTs.

Drawing inspiration from the 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 (see 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 on 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

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

[3]The repository will be 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 Termite fact sheet. Termite is designed to be comparable to the validation set of Spider.

percentage of abbreviations and compound names (see Table 1 and Appendix A). This equivalence will be crucial to limit the influence of the dataset itself on the following evaluations and will be further explored in Section 3.3.

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. Therefore, we can be confident that our dataset did not contribute in any way to the pretraining of LLMs. This aspect will be crucial in the next sections, where we will investigate data contamination in GPT-3.5.

### 3.3 Comparing Hardness of Termite vs. Spider

An inherently different hardness of Termite and Spider may cause imbalances during a comparative evaluation of LLMs over different sets. Then, we aimed to produce an Termite that is as close as possible to Spider.

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 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 B 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 a dataset more difficult than the other. Then, the two datasets can be considered equivalent in terms of hardness of translations.

## 4 Method: studying Data Contamination and its Effect on the Text-to-SQL Task

Our intuition is that data contamination may play an important role in GPT's performance. However, investigating the presence of data contamination in GPT models is extremely difficult if there is no possibility to access training datasets. Then, data contamination can only be estimated.

To investigate our intuition, we first describe a way to quantify the presence of data contamination on GPT by examining database dumps in the Text-to-SQL datasets (Sec. 4.1). Then, we tested GPT-3.5 on the Text-to-SQL task both on possibly already explored and definitely hidden data (Sec. 4.2). We expect a decline in performance when the model is required to make inferences on new data not previously encountered. Finally, we describe an adversarial degradation of the input that makes the task of Text-to-SQL translation harder without prior knowledge (Sec. 4.3). Indeed, we argue that if a model achieves high performance in a task by memorizing previously seen information, reducing the quality of input information would not significantly impact its performance on data it has encountered before.

### 4.1 Tracing Data Contamination

Our aim is to understand whether data contamination may have occurred before testing GPT-3.5

performances on Text-To-SQL task. The data contamination issue criticality emerges when a model is inadvertently trained on data that include or overlap with its testing dataset: this issue may lead to skewed performance metrics and a misrepresentation of the model's true capabilities. For models like GPT-3.5 – black-box models with scarce information about the sources of training – it is necessary to find indirect measures to assess the presence of data contamination.

In the specific case of Text-to-SQL, along with the request to translate the query, the models trained on this task are provided with information regarding the database schema. In particular, LLMs may also have been trained on the dumps of databases in the Text-to-SQL datasets. Hence, it is possible to assess the presence of data contamination on Text-to-SQL datasets by measuring the previous knowledge that a model has on these dumps.

A clue to determine whether the data contamination has occurred is that the model is able to reconstruct missing information regarding the database schema. Since LLMs are trained to produce text, we propose to measure the accuracy that a model achieves in reconstructing a dump that has been masked. If the model is able to reconstruct this information on potentially seen data –as Spider's validation dataset might be – and fails to reconstruct it on the new resources – like Termite– we argue that data contamination has occurred.

In particular, a dump was masked by replacing the 25% of columns in each table with a [MASK] token. Then, GPT-3.5 was prompted to reconstruct the dump by replacing the masked tokens with appropriate column names. In these experiments, the `INSERT` instructions are also removed from the dump to limit the possible inference regarding the names of columns. It is important to note that the task is still feasible even if no data contamination has occurred: column names can be deduced from the names of the tables and other columns. However, the task would be much easier in the presence of data contamination.

Hence, given the reconstructed dump from GPT-3.5, we define the DC-accuracy as the percentage of times the predicted column name is equal to the true column name:

$$\text{DC-accuracy} = \frac{\text{\# of correct columns name}}{\text{\# of columns}}$$

It is possible to assess the presence of data contamination by measuring the DC-accuracy both on the Spider dumps and on the Termite dump databases.

### 4.2 Prompting LLMs for Text-to-SQL Translation

Given an instruction in natural language, LLMs can translate the request into code – and SQL queries, in particular – to answer the given request. Specifically, OpenAI's models for generating text have undergone training to process both natural language and code. These models produce text-based outputs as a result of the inputs they receive. 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 – in particular, `gpt-3.5-turbo` – is prompted in order to obtain the translations and then how it is possible to automatically evaluate the performance of this system on both Spider and Termite datasets.

**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.

In particular, given a target database, in the first message, the model is given the dump of the database. In each dump, information about the tables that constitute the database 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 contents of the dump. For example, considering the database `car_1` in the Spider dataset, the first messages in the conversation are the following:

| |
|---|
| **user:** `CREATE TABLE "continents" [...]; CREATE TABLE "countries" [...];` |
| **GPT-3.5:** `The code above includes the creation of six tables: continents, countries [...]` |

Then, given the dump and the interpretation that the model gives 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 of each other. 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 `car_1` database would be completed by the following messages:

**user:** `Translate in SQL the following query. Answer using only SQL. What is the number of continents?`

**GPT-3.5:** `SELECT COUNT(*) as n_conts FROM continents;`

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.

**Test Suite Accuracy: the Evaluation Metric** We adopted the Test Suite Accuracy metric as an evaluation metric in our experiments. This score was introduced by Zhong et al. (2020) and currently is the official metric for the evaluation of systems tested on Spider[4]. In principle, given a query $q$ generated by a system, one would like to evaluate whether $q$ is semantically equivalent to a human-generated gold query $g$. This metric, known as semantic accuracy, is undecidable in general (Chu et al., 2017). The Test Suite Accuracy metric aims to approximate semantic accuracy and states the correctness or incorrectness of $q$ by comparing the denotation of a gold query $g$ and the denotation of $q$. However, it introduces fewer false positives than Execution Accuracy – that compares $g$ and $q$ on a single database – by comparing the denotation of both queries on as few databases as possible. This set of random databases is called the Test Suite.

[4]As reported on the Spider official website

In our experiments, a Test Suite of maximum 1000 random databases is constructed for each database upon which queries are defined, as reported in the original paper. Therefore, a model-generated query $q$ is executed on the Test Suite databases and labeled correct if no database can distinguish it from the gold query $q$. The 97% of the queries is automatically evaluated, while the remaining ones are manually evaluated by three SQL-proficient annotators. This step is necessary because, in these rarer cases, the queries – either gold or model-generated – make use of functions not available in the SQL server used to execute the queries.

Using the Test Suite Accuracy as an approximation of the semantic accuracy of the system, we show that, among different databases and different query difficulties, GPT-3.5 demonstrates different performance on Spider and Termite datasets (Section 5.2). In addition, the same type of evaluation will be performed on adversarially degraded databases in Section 5.3 to establish that the highest performance degradation occurs on unseen data.

### 4.3 The Adversarial Table Disconnection

The differences in performance over seen rather than unseen data it is not, in principle, sufficient to state that the observed differences are caused by data contamination issues since it is still possible that the datasets hinder some biases. On the one hand, we designed the Termite dataset to be comparable to Spider; hence, once the hardness of the queries is also fixed, performances are comparable. On the other hand, we want to ensure that memorization is, in fact, playing an important role. For this reason, we propose an adversarial approach to state the importance of memorization on this task. We will refer to this method as Adversarial Table Disconnection (ATD).

The ATD aims to make the translation process from a request in natural language into SQL harder. In particular, ATD disconnects the tables from each other, making it more difficult to figure out on which columns the `JOIN` needs to be performed. ATD disconnects tables by removing the foreign keys constraints. In particular, all instructions referring to the creation of the constraint are removed from the dump. This structural information is critical to translating a questions into SQL queries: we argue that the removal of this information has a crucial impact on translation, both for humans and systems, unless the missing information can be re-

trieved. Given the importance of the presence of a foreign key, our aim is to remove insights about the relation between columns that may be directly used to infer the removed relationship. Hence, ATD involves the removal of all the `INSERT` instructions from the dump. In fact, matching values into this type of instructions may give direct clues about the columns relationships. Crucially, our aim is not to change the semantic of the database, but only to make it less easy to understand. For this reason, the original column names are kept, making inferences about the content of the database still possible. Hence, after ATD a model is given a dump deprived of structural information, with tables disconnected from one another but still semantically equivalent to the original one.

Because ATD makes the task more difficult, a drop in system performance is expected. However, the drop can be mitigated by relying on prior information about the database structure. Therefore, given the presence of data contamination, we expect GPT-3.5 to be robust to ATD perturbation on Spider datasets, with a more pronounced performance loss on Termite databases.

# 5 Experiments

## 5.1 Quantifying the Data Contamination in Text-to-SQL datasets

It is possible to quantify the presence of data contamination by comparing the DC-accuracy that GPT-3.5 achieves in predicting column names on a new dumps –from Termite– versus potentially already seen dumps in Spider (as described in Section 4.1). In Table 2, the average DC-accuracy over Spider and Termite datasets is reported.

The model seems to find the task easier on Spider databases than on Termite ones. In particular, the average accuracy of Spider dumps is more than $33\%$, that is, on average, more than $20\%$ higher than the score on Termite. Moreover – while on both datasets, some databases are hard to predict, with a minimum accuracy of $0$ – on the Spider dataset, GPT-3.5 achieves a perfect accuracy on two databases. The same does not hold for Termite, where the highest accuracy is $44\%$. The different performance of the model on these two datasets suggests the presence of data contamination.

It is also interesting to notice that on 35% dumps (7 dumps), the DC-accuracy is over 40%, while only on two databases among the ones in Termite GPT-3.5 achieves (with a score of 44.44% and 40%) the same results. A complete list of accuracies per database can be found in Appendix C The different performance in terms of DC-accuracy over Termite with respect to Spider suggests the presence of data contamination.

| DC-accuracy | **Spider** | **Termite** |
|---|---|---|
| Mean | $33.42(\pm 33.01)$ | $13.21(\pm 18.70)$ |
| Min - Max | $0.00 - 100.00$ | $0.00 - 44.44$ |

Table 2: Average, min, and max accuracies of GPT-3.5 on predicting the masked columns names on dumps in Spider and Termite. The overall performances in terms of DC-accuracy over the Spider dataset are superior with respect to the one that can be observed on Termite dataset.

## 5.2 Measuring GPT-3.5 performances on seen and unseen data

Having estimated the presence of data contamination, we focus on the analysis of the performance of GPT-3.5 on the dataset presented in Section 3. The results described here suggest the role that memorization may play in the performance of a Large Language Model like GPT-3.5. We analyze the model's performance by categorizing queries according to their hardness and averaging across the different databases of the two datasets (see Appendix D for the results on different databases).

Table 3 reports the average Test Suite Accuracy results for each hardness level. We notice that, on both sets of databases, the accuracy of the model decreases as the hardness increases. In particular, the `EASY` queries on the Spider dataset achieves, on average, accuracy over the $90\%$. Accuracy decreases progressively, with the greatest drop ($29\%$) between `MEDIUM` and `HARD` levels. The worst accuracy is obtained on the `EXTRA-HARD` queries. The same trend is also observed on the Termite dataset: on the queries `EASY` GPT-3.5 achieves an average accuracy of $74\%$. Again, a decrease in performance is observed on the `MEDIUM` and `HARD` queries, while on Termite `EXTRA-HARD` queries GPT-3.5 appears to achieve performance similar to `HARD` queries.

However, comparing the results on the two datasets, it is possible to notice that, given a certain hardness level, the accuracy of GPT-3.5 is not comparable on the two datasets. In fact, the average performance difference between Spider and the Termite dataset is remarkable: `EASY` query accuracy decreases by $16\%$, $10\%$ for `MEDIUM` ones.

| Hardness | Original Dumps | | Adversarial Table Disconnection | |
|---|---|---|---|---|
| | **Spider** | **Termite** | **Spider** | **Termite** |
| EASY | $90.11(\pm 11.65)$ | $74.00(\pm 21.19)$ | $91.08(\pm 10.32)$ | $62.00(\pm 19.89)$ |
| MEDIUM | $77.21(\pm 16.35)$ | $67.06(\pm 24.83)$ | $72.71(\pm 23.63)$ | $63.70(\pm 16.03)$ |
| HARD | $48.83(\pm 23.17)$ | $28.33(\pm 23.78)$ | $48.71(\pm 28.79)$ | $22.67(\pm 22.20)$ |
| EXTRA-HARD | $30.94(\pm 23.79)$ | $31.14(\pm 24.54)$ | $28.96(\pm 19.28)$ | $28.98(\pm 17.03)$ |

Table 3: GPT-3.5 accuracy on the Spider Dataset and the Termite Dataset, across four levels of hardness of queries. The results reported are average accuracy across all databases in the two datasets. The first two columns refer to accuracies on the standard task, while the last columns show results after ATD.

The biggest drop is observed on HARD queries, with a $20\%$ difference in performance. The only accuracies that appear to be similar – and sensibly lower – are on EXTRA-HARD hard queries.

These results provide insight that GPT-3.5 capability on the task may be highly influenced by data contamination issues. In fact, given comparable queries from a hardness perspective, results on databases that have never been seen turn out to be worse than those that have already been made available and, likely, observed in training.

### 5.3 Robustness of GPT-3.5 on Text-to-SQL performances after ATD on seen data

To better understand whether the data contamination is responsible for the difference in performance observed in Table 3, we analyze the accuracy over Spider and Termite after ATD.

As expected, a greater performance drop is observed over Termite databases, while the model seems to be robust against the ATD over the Spider dataset. In particular, the accuracy over the EASY queries decreases by $12$ points on average on the Termite dataset, while similar results (close to the $90\%$) can be observed in Spider. On the MEDIUM queries, a slightly more pronounced difference in performances can be observed over Spider ($4.5$ points) with respect to the one observed in Termite ($3.36$ points). It is on the HARD queries, however, that the different performances on seen and unseen data are much more evident. Those queries require more JOIN operations than the previous ones. On the one hand, on the Spider databases, the average performance is around $48\%$ for both the original dumps and the dumps on which ATD is applied. On the other hand, an average performance drop of $5.66$ points is observed on the Termite dumps. Finally, similar and generally lower performances can be observed over the HARD queries.

Hence, this final experiment confirms that –since the drop observed in the performance of GPT-3.5 after ATD is greater on new data than on contaminated ones – the memorization ability of the model plays a crucial role in its performance.

## 6 Conclusions

This paper shows that data contamination is responsible for overestimating the performance of GPT-3.5 on Text-to-SQL. The experiments conducted, using a novel metric for detecting data contamination, clearly demonstrate that GPT-3.5 possesses prior knowledge on the contents of the Spider validation set in contrast to his ignorance of our constructed Text-to-SQL unseen dataset, Termite. In fact, as results show, Text-to-SQL performances on Spider are significantly better than on Termite. This suggests that GPT-3.5 capabilities in zero-shot scenario might not be as surprising as previously thought. Observing the results of data contamination alongside with performances achieved in Text-to-SQL on the two datasets, we concluded that it is indeed the prior knowledge of GPT-3.5 on the test set that makes a significant difference. In addition to this, we found that Adversarial Table Disconnection impacts the results of Text-to-SQL tasks differently across datasets: its influence is relatively mild in the case of the Spider dataset but more pronounced with the Termite dataset.

Since data contamination is the main responsible for overestimating performances on Text-to-SQL and, possibly, on other tasks, a more thorough reexamination of current LLM's benchmarks for downstream tasks in zero-shot scenarios would be needed. Furthermore, it would be beneficial to develop public datasets, like our Termite, that remain outside the LLM's pretraining. This may guarantee that evaluations on pretrained LLMs are not impacted by Data Contamination.

## Limitations

Our analysis of data contamination of GPTs has some limitations. Below, we describe some of these and suggest directions for future work

First, the impact of Data Contamination on the performance of Text-to-SQL tasks has been tested specifically on GPT-3.5. This is a limitation and the analysis should be extended to other models. However, we performed preliminary small-scale pilot experiments akin to those conducted in this study. Results suggest that Data Contamination also affects GPT-4.

Furthermore, we used only a public dataset for this task. However, this single dataset already shows that data contamination is a relevant issue in measuring performance.

## Acknowledgements

## References


Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models.

Kent K. Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to chatgpt/gpt-4.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Shumo Chu, Chenglong Wang, Konstantin Weitz, and Alvin Cheung. 2017. Cosette: An automated prover for sql. In CIDR.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Investigating data contamination in modern benchmarks for large language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In Proceedings of COLING 2012: Posters, pages 401–410, Mumbai, India. The COLING 2012 Organizing Committee.

Shahriar Golchin and Mihai Surdeanu. 2023. Time travel in llms: Tracing data contamination in large language models.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Investigating data contamination for pre-training language models.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation.

OpenAI. 2023a. Gpt-3.5turbo.

OpenAI. 2023b. Gpt's family.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Leonardo Ranaldi, Elena Sofia Ruzzetti, and Fabio Massimo Zanzotto. 2023. PreCog: Exploring the relation between memorization and performance in pre-trained language models. In Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, pages 961–967, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. Codet5+: Open code large language models for code understanding and generation.

David H.D. Warren and Fernando C.N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. American Journal of Computational Linguistics, 8(3-4):110–122.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Kao Ben. 2016. Neural enquirer: Learning to query tables in natural language. In Proceedings of the Workshop on Human-Computer Question Answering, pages 29–35, San Diego, California. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.

Zhiqiang Yuan, Junwei Liu, Qiancheng Zi, Mingwei Liu, Xin Peng, and Yiling Lou. 2023. Evaluating instruction-tuned large language models on code comprehension and generation.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023a. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. 2023b. Self-edit: Fault-aware code editor for code generation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 769–787, Toronto, Canada. Association for Computational Linguistics.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 396–411, Online. Association for Computational Linguistics.

# A Analysis of Column Names in Spider and Termite

The following Table present the percentage of column names that consist in abbreviations or compound nouns on both Spider and Termite dataset. On average, both datasets presents a similar distributions of this kind of columns names. The equivalence in terms of abbreviations and compound nouns, as discussed in Section 3.2 is crucial to a fair evaluation during the estimation of data contamination (Section 4.1).

| | Table | Compound | Abbreviation |
|---|---|---|---|
| Termite | bowling | 0.63 | 0.00 |
| | centri | 0.48 | 0.16 |
| | coronavirus | 0.55 | 0.15 |
| | farma | 0.62 | 0.29 |
| | farmacia | 0.65 | 0.15 |
| | galleria | 0.20 | 0.00 |
| | hackathon | 0.62 | 0.00 |
| | pratica | 0.33 | 0.00 |
| | recensioni | 0.56 | 0.00 |
| | voli | 0.48 | 0.43 |
| Spider | battle_death | 0.33 | 0.00 |
| | car_1 | 0.30 | 0.09 |
| | concert_singer | 0.48 | 0.00 |
| | course_teach | 0.60 | 0.00 |
| | cre_Doc_Template_Mgt | 1.00 | 0.00 |
| | dog_kennels | 0.80 | 0.04 |
| | employee_hire_evaluation | 0.59 | 0.00 |
| | flight_2 | 0.54 | 0.23 |
| | museum_visit | 0.67 | 0.17 |
| | network_1 | 0.57 | 0.00 |
| | orchestra | 0.57 | 0.00 |
| | pets_1 | 0.64 | 0.36 |
| | poker_player | 0.64 | 0.00 |
| | real_estate_properties | 0.95 | 0.41 |
| | singer | 0.60 | 0.00 |
| | student_transcripts_tracking | 0.93 | 0.04 |
| | tvshow | 0.52 | 0.04 |
| | voter_1 | 0.67 | 0.11 |
| | World_1 | 0.42 | 0.15 |
| | wta_1 | 0.86 | 0.28 |

# B Measuring Hardness of queries in Spider and Termite

As described in Section 3.3, we need to ensure that Spider and Termite are comparable in terms of hardness. 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 level of accuracy, then it means that the level of challenge 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 options the SQL query that satisfies the request. All three of the options are syntactically correct SQL queries, but the incorrect answers are semantically different from the correct one. The first incorrect option is designed by the authors, perturbing the correct answer by removing or replacing some operations or some retrieved columns, changing the field and tables names with non-matching ones. The second incorrect answer is, instead, another query extracted from the same dataset as the correct one. The selected query is the most similar under the Bag of Words assumption with respect to the correct one. The similarity of two queries, in order to retrieve this third option, is measured via 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 of the annotators work in a field that requires daily use of the SQL query language. Finally, we further divided the test into two trials of 20 queries each and administered it to the annotators at two different times to limit the presence of errors due to gradual loss of concentration.

## C Assessing the presence of Data Contamination

The following two tables show the DC-accuracy of GPT-3.5 on the Spider (Table 4) and Termite (Table 5). Notice that, as discussed in Section 5.1, the overall performance in terms of DC accuracy on the Spider dataset is higher than that observed on the Termite dataset. Those results indicate the presence of data contamination.

| Database | DC-accuracy |
|---|---|
| battle_death | 0.16 |
| car_1 | 0.00 |
| concert_singer | 0.78 |
| course_teach | 0.00 |
| cre_Doc_Template_Mgt | 0.40 |
| dog_kennels | 0.52 |
| employee_hire_evaluation | 0.20 |
| flight_2 | 0.00 |
| museum_visit | 0.00 |
| network_1 | 1.00 |
| orchestra | 0.43 |
| pets_1 | 0.50 |
| poker_player | 0.50 |
| real_estate_properties | 0.46 |
| singer | 0.00 |
| student_transcripts_tracking | 0.22 |
| tvshow | 0.00 |
| voter_1 | 1.00 |
| wta_1 | 0.16 |

Table 4: GPT-3.5 DC-accuracy across the different databases in Spider

| Database | DC-accuracy |
|---|---|
| bowling | 0.14 |
| centri | 0.00 |
| coronavirus | 0.44 |
| farma | 0.00 |
| farmacia | 0.00 |
| galleria | 0.00 |
| hackathon | 0.33 |
| pratica | 0.00 |
| recensioni | 0.40 |
| voli | 0.00 |

Table 5: GPT-3.5 DC-accuracy across the different databases in Termite

## D Text-to-SQL GPT-3.5 detailed performances

The following Table shows the results for each database in the Text-to-SQL task both for Spider and Termite dataset. Notice that the accuracy decreases as the hardness increases and that on Termite, results are generally lower.

| | | Termite | | | | |
|---|---|---|---|---|---|---|
| | difficulty | hackathon | galleria | recensioni | centri | pratica |
| Original | easy | 60.0 | 80.0 | 40.0 | 100.0 | 60.0 |
| | medium | 50.0 | 100.0 | 40.0 | 100.0 | 50.0 |
| | hard | 25.0 | 66.66 | 0.0 | 0.0 | 33.33 |
| | extra | 50.0 | 30.0 | 0.0 | 25.0 | 57.14 |
| ATD | easy | 60.0 | 40.0 | 40.0 | 60.0 | 60.0 |
| | medium | 50.0 | 80.0 | 60.0 | 71.42 | 50.0 |
| | hard | 25.0 | 0.0 | 50.0 | 0.0 | 33.33 |
| | extra | 33.33 | 30.0 | 0.0 | 25.0 | 57.14 |

| | | Termite | | | | |
|---|---|---|---|---|---|---|
| | difficulty | coronavirus | farmacia | voli | bowling | farma |
| Original | easy | 60.0 | 60.0 | 80.0 | 100.0 | 100.0 |
| | medium | 60.0 | 40.0 | 100.0 | 55.55 | 75.0 |
| | hard | 40.0 | 60.0 | 25.0 | 33.33 | 0.0 |
| | extra | 0.0 | 40.0 | 20.0 | 14.28 | 75.0 |
| ATD | easy | 40.0 | 60.0 | 80.0 | 80.0 | 100.0 |
| | medium | 60.0 | 60.0 | 100.0 | 55.55 | 50.0 |
| | hard | 0.0 | 60.0 | 25.0 | 33.33 | 0.0 |
| | extra | 40.0 | 20.0 | 20.0 | 14.28 | 50.0 |

| | | Spider | | | | |
|---|---|---|---|---|---|---|
| | difficulty | battle_death | car_1 | concert_singer | course_teach | cre_Doc_Template_Mgt |
| Original | easy | 100.0 | 94.44 | 100.0 | 75.0 | 100.0 |
| | medium | 62.5 | 40.62 | 62.5 | 85.71 | 79.54 |
| | hard | 0.0 | 18.75 | 61.54 | 62.5 | 40.0 |
| | extra | 50.0 | 15.38 | 50.0 | | 33.33 |
| ATD | easy | 100.0 | 88.89 | 100.0 | 75.0 | 91.66 |
| | medium | 62.5 | 50.0 | 66.66 | 85.71 | 79.54 |
| | hard | 0.0 | 18.75 | 76.92 | 62.5 | 40.0 |
| | extra | 25.0 | 385 | 0.0 | | 50.0 |

| | | Spider | | | | |
|---|---|---|---|---|---|---|
| | difficulty | dog_kennels | employee_hire_evaluation | flight_2 | museum_visit | network_1 |
| Original | easy | 90.0 | 100.0 | 84.62 | 100.0 | 100.0 |
| | medium | 80.55 | 92.86 | 76.66 | 87.5 | 77.27 |
| | hard | 60.0 | 80.0 | 62.5 | 66.66 | 56.25 |
| | extra | 53.85 | 0.0 | 12.5 | 25.0 | 83.33 |
| ATD | easy | 90.0 | 100.0 | 92.31 | 100.0 | 100.0 |
| | medium | 77.78 | 100.0 | 46.66 | 100.0 | 81.82 |
| | hard | 80.0 | 80.0 | 50.0 | 100.0 | 62.5 |
| | extra | 50.0 | 25.0 | 6.25 | 0.0 | 50.0 |

| | | Spider | | | | |
|---|---|---|---|---|---|---|
| | difficulty | orchestra | pets_1 | poker_player | real_estate_properties | singer |
| Original | easy | 85.71 | 100.0 | 93.75 | 100.0 | 100.0 |
| | medium | 83.33 | 81.82 | 100.0 | 50.0 | 100.0 |
| | hard | 83.33 | 50.0 | 62.5 | 0.0 | 33.33 |
| | extra | 50.0 | 30.0 | | | |
| ATD | easy | 100.0 | 100.0 | 87.5 | 100.0 | 83.33 |
| | medium | 77.78 | 68.18 | 100.0 | 0.0 | 88.89 |
| | hard | 83.33 | 66.66 | 50.0 | 0.0 | 33.33 |
| | extra | 50.0 | 30.0 | | | |

| | | Spider | | | | |
|---|---|---|---|---|---|---|
| | difficulty | student_transcripts_tracking | tvshow | voter_1 | world_1 | wta_1 |
| Original | easy | 65.38 | 80.0 | 66.66 | 79.16 | 87.5 |
| | medium | 62.5 | 86.66 | 100.0 | 67.39 | 66.66 |
| | hard | 37.5 | 60.0 | | 50.0 | 42.86 |
| | extra | 15.0 | 0.0 | 50.0 | 26.66 | 0.0 |
| ATD | easy | 65.38 | 85.0 | 100.0 | 75.0 | 87.5 |
| | medium | 66.66 | 80.0 | 100.0 | 58.69 | 63.33 |
| | hard | 25.0 | 30.0 | | 45.0 | 21.43 |
| | extra | 25.0 | 50.0 | 25.0 | 23.33 | 50.0 |

Table 6: Test-Suite Evaluation results for GPT-3.5