The Problem of Ambiguity in Table Question Answering

Jorge Osés Grijalba[1,3], Luis Alfonso Ureña-López[1],
Eugenio Martínez Cámara[1], Jose Camacho-Collados[2]
[1]SINAI Research Group, Advanced Studies Center in ICT (CEATIC)
University of Jaén, Spain
[2]Cardiff NLP, Cardiff University, United Kingdom
[3]Law Business Research, United Kingdom
jorgeosesgrijalba@gmail.com, laurena@ujaen.es, emcamara@ujaen.es,
camachocolladosj@cardiff.ac.uk

Abstract

Question Answering on Tabular Data (or *Table Question Answering*) has seen tremendous advances with the coming of new generation Large Language Models (LLMs). Despite this, significant challenges still remain to be solved if we are to develop robust enough approaches for general usage. One of these is ambiguity in question answering, which historically has not merited much attention due to the previously limited capabilities of LLMs. In this work, we outlay the main types of ambiguousness inherent to tabular data. Then, we discuss how they are influenced by the way our models interact with the information stored in the tables, and we test the capabilities of some LLMs in detecting them. This work provides an initial ground for a deeper discussion on how to approach ambiguity in Tabular Data in the age of LLMs.

1 Introduction

Traditionally, **question answering (QA)** has focused on retrieving answers from questions posed within the context of plain text documents (Voorhees, 2001; Rajpurkar et al., 2016). These models are typically grounded in textual evidence to identify the correct response (Khashabi et al., 2020), in contrast to open-domain QA approaches that aim to operate independently of explicit supporting passages (Zhang et al., 2023). Beyond unstructured text, valuable information also exists in textual structured formats, such as tables found on Wikipedia (Pasupat and Liang, 2015; Kweon et al., 2023), within relational databases, or in simpler formats like CSV files. In this paper, we explore the emerging capability of large language models (LLMs) to answer questions based on such tabular data, namely question answering over tables (Tabular QA) (Jin et al., 2022).

The QA research community has increasingly recognized **ambiguity** as a key challenge in real-world applications (Papakostas and Papadopoulou, 2023). Ambiguity may stem from the phrasing of the question or from the multiplicity of valid answers it allows, often resulting in multiple plausible interpretations. Early open-domain QA benchmarks tended to assume that each question had a single correct answer, but subsequent analyses have revealed that this assumption often does not hold. For instance, an analysis of the Natural Questions (Kwiatkowski et al., 2019) benchmark found that over 50% of user questions were ambiguous, with ambiguity arising from a range of sources (Min et al., 2020). This insight has driven a growing body of work focused on detecting and managing ambiguity in QA tasks. Min et al. (2020) formally introduced the task of answering ambiguous open domain questions as given a question to predict the plausible pairs of questions and answers. They also released the dataset AmbigNQ and the baseline model AmbigQA, which boosted the interest on ambiguity in QA. For example, (Gao et al., 2021) proposed the Refuel model, which employs generative evidence fusion and a round-trip prediction mechanism to uncover missing interpretations. More recently, researchers such as (Cole et al., 2023) have shifted attention toward model calibration and answer uncertainty, enabling systems to recognize ambiguous queries and either abstain or provide multiple possible answers as appropriate.

Recent work addresses ambiguity in Text-to-SQL generation using SQL-based representations. Some studies propose benchmarks for lexical and structural ambiguity (Saparina and Lapata, 2024; Bhaskar et al., 2023), while others show how database documentation can aid disambiguation (Huang et al., 2023), or use data profiling to generate targeted training data (Veltri et al., 2023). These approaches rely on relational structures and SQL interfaces. In contrast, LLMs allow for working with arbitrary tabular formats and question types, where ambiguity may stem from the data itself, not just

Findings of the Association for Computational Linguistics: EACL 2026, pages 3835–3848

ID	Name	A Points	Salary	B Points	Question	Type	Possible Answers
1abc	John	4	3100	4	How many A points does John have?	Row Choice	4,1
de2f	Paul	3	3050	4	How many points does Richard have?	Column Choice	2,3
g1hi	George	2	3075	3	List the 3 names of those with the most overall A points	Ranking	Triple tie
j4kl	Richard	2	3000	3	Who has the least A points?	Identification	John and ID: pq6r
mno	Pete	2	2800	2	How many people earn over 3k USD monthly?	Unit	1 to 6
pq6r	John	1	*null*	1	What is the most recently assigned ID?	Operational	6 IDs

Table 1: Example of tabular data shown on the left as the reference dataset, with individual IDs, names, points, and salaries; on the right are examples of questions yielding ambiguous answers based directly on this data.

the schema structure. For instance, *What is John's ID?* over Table 1 is ambiguous if multiple rows include "John". Yet, benchmarks such as DataBench (Osés Grijalba et al., 2024) and Open Wikitables (Kweon et al., 2023) mostly contain unambiguous queries, overlooking real-world ambiguities. We propose an initial taxonomy of tabular ambiguity types agnostic to the method we use for data interaction and evaluate how well LLMs detect ambiguity in Tabular QA and in a dedicated ambiguity detection task.

2 Ambiguity in Tabular QA

In this section, we examine key types of ambiguity that are particularly relevant in tabular data settings. While similar ambiguities can arise in other contexts, and general QA challenges certainly apply, we argue that their manifestation in the interaction between LLMs and tables warrants separate attention. Our goal is not to provide an exhaustive taxonomy, but rather to highlight representative cases as a foundation for future work.

Example. For ease of understanding, we will first introduce a small toy dataset illustrated in Table 1 (left). In the dataset, we have five columns: (1) *ID*, an alphanumerical unique identifier; (2) *Name* which contains a number of non-unique person names; (3) *A Points*, *Salary* and *B Points* which contain numerical values. No further context is available for this dataset. In this case, it is not possible to know the relation between different points, where the ID comes from or which currency or frequency is associated with the given salary.

In total, we identified six ambiguity types. Examples of such ambiguity types are further illustrated in Table 1.

(1) Row Choice. The first and most immediate form of ambiguity arises when the user does not provide enough information to determine the specific subset of rows in the table they are referring to. This kind of ambiguity closely resembles that found in natural language.

(2) Column Choice. Another common case occurs when the user's question lacks sufficient detail to clearly identify which column contains the desired information.

(3) Ranking Ties. It occurs when multiple rows may tie under a given criterion. For example, in Table 1, George, Richard, and Pete all have the same overall points, resulting in a tie.

(4) Identification. Occasionally, it might not be clear what the model's response should look like. For example, when identifying *John*, we might give both the name and the ID, just the ID or even just the name according to the databases' caveats. This is especially relevant when the model's response is to be consumed by another automated system, which might expect a given format.

(5) Units. Ambiguity can also arise from missing unit information in numerical columns. In our example, it is unclear whether the salary is daily, monthly or annual or even what currency it is in, which may be necessary to correctly interpret the data.

(6) Operational. Sometimes a user may ask about the result of an operation on a column, even though the operation is not explicitly defined for the model. For instance, asking for the "most recently generated ID" assumes the model can infer temporal order from the textual representation of the IDs, but this may not always be clear or feasible.

We must also note a caveat that affects Tabular QA as a whole, and that is the existence of *null*

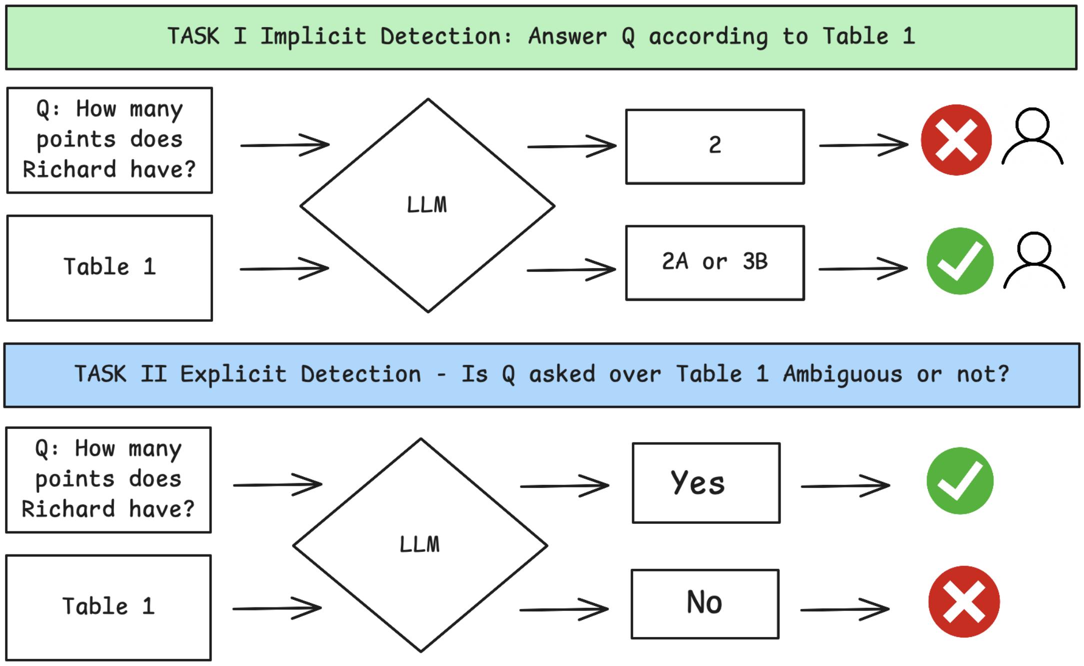

Figure 1: Overview of the two tasks (implicit and explicit ambiguity detection) analysed in experiments.

or *missing* values. In our example, the last row is missing a *Salary* value in Table 1. This should be taken into account by the models before answering, ideally asking the user for more context on how to handle *null* values. Future work will explore these limitations, including challenges related to querying non-existent data or impractical operations within the table.

We hypothesize that current models struggle with the types of ambiguity discussed when applied to Tabular QA. This is especially evident when output formats are constrained, often leading models to provide a single answer even when multiple interpretations are valid. The main goal of our experiments in the following section is precisely to test this hypothesis.

3 Ambiguity Detection

We perform experiments with LLMs for detecting ambiguity in the context of Tabular QA. The two tasks comprising this experiment are illustrated in Figure 1.

3.1 Experimental Setting

LLM approaches. We experiment with two distinct approaches to using LLMs with tabular data and their implications for ambiguity.[1]

(1) In-Context. The whole dataset is passed in the LLM's context window and the LLM provide the answer directly. Therefore, in this setting, the model will theoretically be able to detect any ambiguity present within its context. This approach, however, have been shown less reliable and more prone to hallucinations and explanations given by the model as to why an answer is chosen are often hallucinated (Osés Grijalba et al., 2024).

(2) Code-based. This is the case where the model interacts with the table through code as an interface, be it SQL, Python or any other language, but does not hold an entire representation of the table within its context. In this case, the model might not have access to all of the information it needs to detect ambiguity in the columns, and it will have to write code to detect it programatically. It will rely only on a limited description of the dataset, which will often be enough to write Python code or SQL queries to interact with it. Experiments are performed for code-based task in *Python*. In the case of *In-Context Learning*, we also ask the models for an explanation of the answer given, which can be helpful to subsequently analyse.

Models. We compare two proprietary state-of-the-art models, *gpt-4.1* and *claude-sonnet*, each in two versions, along with two smaller open-source models, *Llama-3.2-1B-Instruct* and *Llama-3.1-8B-Instruct*, with 1, 8, and 70 billion parameters respectively. All models are listed in Table 2.

[1] All prompts can be found in Appendix B.

Model name	(1) Implicit Detection: Overall Performance Not Ambiguous	(1) Implicit Detection: # Detected Ambiguous	(2) Explicit Detection: Accuracy	Precision	Recall	F1
In-Context						
claude-4.5-sonnet	78.90	**52 / 300**	**73.26**	63.80	71.91	**67.61**
claude-3.7-sonnet	75.32	47 / 300	72.09	61.48	75.00	67.57
gpt-4.1	72.36	37 / 300	72.48	62.76	71.33	66.77
gpt-4o	70.46	31 / 300	60.34	49.24	75.33	59.55
gpt-4.1-mini	71.94	17 / 300	67.83	56.56	73.33	63.86
gpt-4.1-nano	51.27	10 / 300	60.98	49.59	40.00	44.28
llama-3.3-70b-instruct	63.71	8 / 300	70.16	64.81	50.33	66.95
llama-3.1-8B-instruct	36.71	0 / 300	52.84	44.02	34.07	38.41
llama-3.2-1B-instruct	18.57	0 / 300	36.95	38.58	**92.94**	54.53
Code-based						
claude-4.5-sonnet	80.80	15 / 300	70.93	64.95	64.51	64.73
claude-3.7-sonnet	80.17	8 / 300	69.77	62.27	70.73	66.23
gpt-4o	81.01	7 / 300	57.49	45.68	38.54	41.81
gpt-4.1	81.43	8 / 300	72.74	**68.78**	54.88	61.05
gpt-4.1-mini	**83.33**	4 / 300	64.21	56.52	39.66	46.61
gpt-4.1-nano	64.98	3 / 300	53.49	43.75	46.34	45.01
llama-3.3-70b-instruct	78.27	0 / 300	47.80	40.93	55.19	47.00
llama-3.1-8B-instruct	50.63	0 / 300	41.73	39.56	51.84	44.88
llama-3.2-1B-instruct	20.04	0 / 300	26.23	38.82	38.31	38.56

Table 2: Experimental results for both (1) implicit and (2) explicit ambiguity detection. Overall performance of the implicit detection task refers to the usual table QA accuracy on 474 unambiguous questions.

Data and QA pairs We use twenty datasets from different domains, detailed in Table 3 (Appendix A), with sampled versions employed to ensure consistent comparisons between *In-Context* and *Code-based approaches*.

Data and Code Release All of the code, tagged result pairs and used datasets can be found in our public GitHub repository[2].

We include 474 non-ambiguous QA pairs from the DataBench lite benchmark used in the SemEval 2025 Task 8 competition (Osés-Grijalba et al., 2025), along with 300 manually constructed ambiguous questions (15 per dataset) and 50 manually generated questions for each category described in section 2.

3.2 Task 1: Implicit Detection

For this task, we aim to assess whether models can identify ambiguous questions in a QA task without being explicitly instructed to do so. To assess this, we manually examine how often the model recognises the questions marked as ambiguous. For evaluation, we will measure response accuracy on unambiguous questions, while manually checking if ambiguous questions are detected when no prompt directs the model to identify them.

[2] `https://github.com/jorses/tabular-qa-amb`

Detection Criteria To enable LLMs to answer questions flexibly while still demonstrating awareness of the ambiguity, we define three criteria for a response to be considered successful in identifying ambiguity. A response satisfies this requirement if it meets *one* or more of the following conditions:

1. It explicitly states that multiple valid answers exist (without requiring the use of the word *ambiguous*).
2. It lists a number of valid answers rather than selecting a single one.
3. It provides an explanation that indicates the presence of multiple interpretations (e.g., due to ties), even if the individual answers are not enumerated.

For example, when asked "How many A points does John have?" (Table 1), a response that notes that multiple individuals named John appear in the table is considered ambiguity-aware, whereas a response that simply states a single value (e.g., "John has 4 A points") is not. Similarly, for the question "Who has the least A points?", listing both John and the individual with ID `pq6r` satisfies the criterion, while naming only one of them does not. In ranking-style questions such as "List the 3 names of those with the most overall A points," explaining that several individuals are tied and that no unique top three exists is sufficient to demonstrate recognition of ambiguity, whereas providing an arbitrary ordering without acknowledging the tie is not.

3.3 Task 2: Explicit Detection

In this second experiment, we require the models to label a question as potentially ambiguous or not in the context of Tabular QA, as a binary classification problem. We analyse it through the typical metrics of accuracy, precision, recall and F1, since the QA set is imbalanced in favour of non-ambiguous questions. Faulty model responses, which generally occur in small models only, are taken as wrong answers for accuracy, but do not affect precision. These experiments are carried on the full set of 747 questions, comprising the 474 unambiguous questions and 300 ambiguous questions described in Appendix A.

4 Results

Table 2 shows the results for the implicit and explicit detection tasks. One key observation is that Tabular QA models generally do not flag ambiguity unless prompted. In the Implicit Detection task, models answered ambiguous questions without indicating uncertainty, particularly in the code-based setting, where detection was nearly absent. In-Context prompting led to slightly more recognition of ambiguity, though still limited. Only the larger propietary models are able to detect ambiguity in a small number of cases, with the best model being claude-4.5 with 52/300 (17% accuracy). This contrasts with the solid accuracy shown on unambiguous questions, often around or above 80% for stronger proprietary models. Proprietary models consistently outperform smaller open-source ones. Moreover, large proprietary models maintain stable accuracy across prompting styles, and open-source models tend to do better in coding approaches.

In task 2 (Explicit Detection), models often over-identify ambiguity, especially with the In-Context approach. This is most pronounced in Llama-3.2-1B-Instruct, which shows very low precision despite high recall. In general, larger models perform reasonably well at detecting ambiguity when prompted explicitly, with models like Claude 4.5 Sonnet achieving competitive scores (67.61 F1). This stands in contrast to the limited awareness displayed when ambiguity must be inferred without direct prompting.

5 Discussion

In the explicit In-Context task 2, we requested both an answer and explanation (see Appendix C for some examples); in Code-based approaches, the explanation is the code. Smaller models, such as Llama-3.2-1B, tend to overanalyse. For example, it marked "What are the 3 hottest temperatures recorded?" as ambiguous due to the word "hottest," despite a sortable temperature column. This pattern appears often and reduces reliability in ambiguity detection, sometimes even on clear questions. Additional examples are in Tables 4–7 (Appendix C). Llama-3.2-1B gave the response "the question is ambiguous" in 38 explicit in-context cases. Broader testing is needed before drawing firm conclusions.

Disambiguation. Most ambiguities can be resolved with sufficiently specific prompting, either by using user-provided criteria or by notifying the user of multiple valid answers and their justifications. For example, instructing the model to interpret *A Points* as *B Points* resolves the *Column Choice* ambiguity in Table 1, while specifying individuals by unique *ID* addresses the *Row Choice* ambiguity. Similar strategies apply to other types. A robust prompting strategy for any Tabular Reasoning system using LLMs should include predefined rules for such cases. Still, no single strategy can handle all types of ambiguity, and the examples shown represent only a subset. While prompting helps reduce uncertainty and guide the model, the model must also recognize ambiguities on its own.

6 Conclusions

LLMs are increasingly used for table QA problems. However, ambiguity in the context of table QA has not been fully addressed yet, while being an important problem, which differs from general ambiguity found in text. Our experiments show that LLMs tend to answer questions without recognising ambiguity unless explicitly prompted. When prompted directly about ambiguity, they occasionally identify it correctly, which highlights the potential for finding an LLM-based solution for this problem. Nonetheless, even in this setting, they still produce many false positives, especially smaller models.

Limitations

The proposed typology is grounded in practical experience but does not stem from a comprehensive study of the ambiguities involved. This is partly due to the lack of Tabular QA research that explicitly addresses ambiguity outside of the SQL context. Further research is also needed to validate and accurately resolve these ambiguous cases. Ideally, models should be capable of identifying ambiguity types relevant to the user without overreacting to clear, unambiguous questions. While this paper represents an initial attempt for dealing with the ambiguity problem in general-purpose tabular QA, we believe further work would be needed to seamlessly integrate ambiguity detection in LLM responses.

Acknowledgments

This work is funded by the Ministerio para la Transformación Digital y de la Función Pública and Plan de Recuperación, Transformación y Resiliencia - Funded by EU – NextGenerationEU within the framework of the project Desarrollo Modelos ALIA. This work has also been partially supported by Project CONSENSO (PID2021-122263OB-C21) and Project SocialTox (PDC2022-133146-C21) funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, Project ROMANET (CERV-2024-CHAR-LITI-101215052), funded by the European Union under the Citizens, Equality, Rights and Values programme, Project HEART-NLP-UJA (PID2024-156263OB-C21) and project VERITAS-H (AIA2025-163322-C64) funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU, Project GALENO-IA (DGP_PIDI_2024_00852) funded by Junta de Andalucía. Jose Camacho-Collados was supported by a UKRI Future Leaders Fellowship.

References

Susant Achary. 2021. Holiday package purchase prediction dataset. Kaggle Dataset.

Abien Fred Agarap. 2018. Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network (rnn). *arXiv preprint arXiv:1805.03687*.

Alexandra. 2018. Generic food database. data.world Dataset.

Sourav Banerjee. 2024. Maternal mortality dataset. Kaggle Dataset. Accessed 2025-03-01.

Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and improving text-to-SQL generation under ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7053–7074, Singapore. Association for Computational Linguistics.

Fahmida Chowdhury. 2023. Bookstore inventory: Best sellers and new releases. Kaggle Dataset. Accessed 2025-01-07.

City of New York. 2022. Nyc 311 service requests. Open Government Data. Accessed 2023-10-20.

Jeremy R. Cole, Michael JQ Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Data.gov. 2018. Police department incident reports, 2018 to present. Open Government Dataset.

Forbes. 2022. Forbes billionaires. Forbes Website. Accessed 2023-10-20.

Yifan Gao, Henghui Zhu, Patrick Ng, Cicero Nogueira dos Santos, Zhiguo Wang, Feng Nan, Dejiao Zhang, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021. Answering ambiguous questions through generative evidence fusion and round-trip prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3263–3276, Online. Association for Computational Linguistics.

Graphext. 2023. Love survey. Scraped Dataset by Graphext. Accessed 2023-10-20.

Zezhou Huang, Pavan Kalyan Damalapati, and Eugene Wu. 2023. Data ambiguity strikes back: How documentation improves gpt's text-to-sql. *Preprint*, arXiv:2310.18742.

Ahmed Mohamed Ibrahim. 2024. Coffee shop sales dataset. Kaggle Dataset. Accessed 2025-03-01.

Inside Airbnb. 2022. Airbnb listings in madrid, spain. Inside Airbnb Dataset.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: Recent advances. In *Knowledge Graph and Semantic Computing: Knowledge Graph Empowers the Digital Economy*, pages 174–186, Singapore. Springer Nature Singapore.

Kaggle. 2021. Titanic dataset. Kaggle Dataset.

Kaggle. 2022. New york city weather, 1869–2022. Kaggle Dataset.

Kaggle. 2023a. Airbnb listings in new york city. Kaggle Dataset.

Kaggle. 2023. Fifa 21 complete player dataset. Kaggle, www.kaggle.com/datasets/stefanoleone992/fifa-21-complete-player-dataset. Kaggle dataset. URL: www.kaggle.com/datasets/stefanoleone992/fifa-21-complete-player-dataset.

Kaggle. 2023b. Us tornado dataset, 1950–2021. Kaggle Dataset.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. Open-WikiTable : Dataset for open domain question answering with complex reasoning over table. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8285–8297, Toronto, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

Rob Mulla. 2021. Roller coaster database. Kaggle Dataset.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13471–13488, Torino, Italia. ELRA and ICCL.

Jorge Osés-Grijalba, Luis Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2025. SemEval-2025 task 8: Question answering over tabular data. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, Vienna, Austria. Association for Computational Linguistics.

Konstantinos Papakostas and Irene Papadopoulou. 2023. Model analysis & evaluation for ambiguous question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4570–4580, Toronto, Canada. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *CoRR*, abs/1508.00305.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Meg Risdal. 2017. New york city taxi trip duration. Kaggle Dataset.

Anita Rostami. 2024. Montesinho forest fire prediction dataset. Kaggle Dataset. Accessed 2025-03-01.

Irina Saparina and Mirella Lapata. 2024. Ambrosia: A benchmark for parsing ambiguous questions into database queries. *Preprint*, arXiv:2406.19073.

UCI ML Repository. 2021. Heart failure prediction dataset. Kaggle Dataset. Originally published by UCI ML.

Enzo Veltri, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Data ambiguity profiling for the generation of training examples. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 450–463.

Ellen M Voorhees. 2001. The trec question answering track. *Natural Language Engineering*, 7(4):361–378.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2023. A survey for efficient open domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

A Datasets

In Table 3 we illustrate the different datasets used as well as the different questions of each type for each of them.

B Prompts

We show the prompts used for tackling the task: *in-context learning* and *code-based*.

	Name	Rows	Cols.	QA	Ambiguous QA	Source
1	Forbes	20	17	25	15	Forbes (Forbes, 2022)
2	Titanic	20	8	20	15	Kaggle (Kaggle, 2021)
3	Love	20	35	20	15	Graphext (Graphext, 2023)
4	Taxi	20	20	20	15	Kaggle (Risdal, 2017)
5	NYC Calls	20	46	20	15	City of New York (City of New York, 2022)
6	London Airbnbs	20	74	20	15	Kaggle (Kaggle, 2023a)
7	Fifa	20	59	20	15	Kaggle (Kaggle)
8	Tornados	20	14	20	15	Kaggle (Kaggle, 2023b)
9	Central Park	20	6	20	15	Kaggle (Kaggle, 2022)
10	ECommerce Reviews	20	10	20	15	Kaggle (Agarap, 2018)
11	SF Police	20	35	20	15	US Gov (Data.gov, 2018)
12	Heart Failure	20	12	20	15	Kaggle (UCI ML Repository, 2021)
13	Roller Coasters	20	56	20	15	Kaggle (Mulla, 2021)
14	Airbnb Madrid	20	75	20	15	Inside Airbnb (Inside Airbnb, 2022)
15	Food Names Embeddings	20	4	20	15	Data World (Alexandra, 2018)
16	Holiday Package Sales	20	20	20	15	Kaggle (Achary, 2021)
17	Gestational Health	20	7	31	15	Kaggle (Banerjee, 2024)
18	Forest Fires	20	11	39	15	Kaggle (Rostami, 2024)
19	Coffee Transations	20	17	38	15	Kaggle (Ibrahim, 2024)
20	Book Sales	20	13	41	15	Kaggle (Chowdhury, 2023)
	Total	**400**	**649**	**474**	**300**	

Table 3: Datasets included in the experiments with their number of (sampled) rows and columns, as well as the number of unambiguous and ambiguous QA for each of them.

In-Context Learning prompt For Implicit Detection.

You are a Question Answering System. Your only goal is to respond to the question asked. You should return a JSON with the following fields:
* explanation: A short explanation as to why you think the answer provided is right.
* answer: This field should be in the format of a SemanticType Python value.
For example: {
"explanation": "I think the dataset has 20 rows",
"answer": "20"
}
For example:
{
"explanation": "I think the dataset has 5 columns",
"answer": "5"
}
Do not write a single word more than necessary. Answer only with the JSON, and do not wrap it in markdown or "' or anything. Answer the question based only on the information present in the following dataset, represented as a CSV: ...
Question: ...
Answer:

Code-based Learning prompt For Implicit Detection.

You are a Function Question Answering System. Your goal is to write a function that answers the users' question in the following format:
import pandas as pd
import numpy as np
def answer(ds: pd.DataFrame) -> ...:
# Your code here
* Do not write a single char more than necessary. * These are the first first five rows of the dataset that must be used to answer represented as a CSV:
...
* Do not wrap your code in "'python markers and do not write any explanation. Return just the function. Plain text.
* You already have access to pandas as pd and numpy as np, but do not import them nor any other library.
Question: ...
Answer:

In-Context Learning prompt For Explicit Detection.

You are a Question Ambiguity Detection System. Your only goal is to return a response stating whether the question is ambiguous or not. You should return a JSON with the following fields:
* explanation: A short explanation as to why you think it is ambiguous or not
* isAmbiguous: This field should only be either 'Yes' or 'No'. No more values are allowed. It should be the conclusion to the explanation. For example:
{
"explanation": "The question is ambiguous",
"isAmbiguous": "Yes."
}
For example:
{
"explanation": "The question is not ambiguous",
"isAmbiguous": "No."
}
Do not write a single word more than necessary.
Answer only with the JSON, and do not wrap it in markdown or "' or anything. Answer the question based only on the information present in the following dataset, represented as a CSV:
...
Is the question "..." ambiguous?
Answer:

Code-based Learning prompt For Explicit Detection.

You are a Function Question Answering System. Do not write a single char more than necessary. These are the first first five rows of the dataset that must be used to answer represented as a CSV: "..."
Your goal is to return a function like the following written in Python.
import pandas as pd
import numpy as np
def answer(ds: pd.DataFrame) -> bool:
# Your code here
* The function you write should return a boolean indicating whether the question is ambiguous or not.
* The dataset has more rows than the first five you see here. You can use all of its rows to determine the answer.
* Do not wrap your code in "'python markers and do not write any explanation. Return just the function.
* You already have access to pandas as pd and numpy as np, but do not import them nor any other library.
Is the question 'question' ambiguous?
Answer:

C Examples of LLMs outputs

This appendix is focused on illustrating how the LLMs respond to ambiguous questions. All of the following tables include responses showcasing different ambiguity types.

Question	Ambiguity Type	claude-3.7-sonnet	gpt-4o
What is the minimum weight along the five oldest patients?	**Ranking**	**Unambiguous** The question is not ambiguous because it clearly asks for the minimum weight among the five oldest patients in the dataset. The dataset includes an 'Age' column that allows for identifying the oldest patients, and a 'Weight' column to determine the minimum weight among those patients.	**Ambiguous** The dataset does not clearly define how to select exactly five oldest patients when multiple patients share the same age. It is unclear how to handle ties and whether to include all patients with the same oldest age or only five individuals.
How old is the patient who has gone through 1 pregnancy?	**Row Choice**	**Ambiguous** The question is ambiguous because multiple people in the dataset have been through 1 pregnancy, and they have different ages ranging from 16 to 19 years old.	**Ambiguous** The dataset shows multiple people with age 16, 17, 18, and 19 who have gone through 1 pregnancy, so the question does not specify which person or instance to consider.
Does any patient weight more than 100 pounds?	**Unit**	**Ambiguous** The question is ambiguous because the weights in the dataset are provided in kilograms (ranging from 48–55), not pounds. Without a conversion between these units, it's unclear how to determine if any patient weighs more than 100 pounds.	**Unambiguous** The question is not ambiguous because it clearly asks if any patient weighs more than 100 pounds, and all weights are provided in the dataset.
Is any patient more than 50?	**Column Choice**	**Ambiguous** The question is ambiguous because it doesn't specify what value should be more than 50. It could refer to age, weight, height, BMI, pregnancy number, heredity, or prediction.	**Ambiguous** The question 'Is any patient more than 50?' is ambiguous because it does not specify what attribute is being referred to, whether it is age, pregnancy number, weight, height, BMI, heredity, or prediction.
Who is the shortest patient?	**Identification**	**Unambiguous** The question is not ambiguous. The dataset contains a 'Height' column with values in centimeters, and we can clearly determine the shortest patient by finding the minimum value in this column, which is 140 cm.	**Ambiguous** The dataset contains height information for multiple patients, but no unique identifier to distinguish individual patients. Multiple patients may have the same height, making it unclear who exactly is the shortest patient.
What is the most important age in the dataset for our study?	**Operational**	**Ambiguous** The question is ambiguous because it doesn't specify what makes an age 'important' in this dataset. It could refer to the most frequent age, the age with highest correlation to prediction, or some other significance that is not clear from the context.	**Ambiguous** The question is ambiguous because 'most important age' is undefined and lacks criteria for importance in the dataset context.

Table 4: Examples of In-Context Explicit Ambiguity Detection of Different Types.

Question	Ambiguity Type	Llama-3.1-8B	Llama-3.2-1B
What is the minimum weight along the five oldest patients?	**Ranking**	**Unambiguous**. The question clearly specifies a condition (five oldest patients) and an attribute (minimum weight).	**Ambiguous**. The question is ambiguous.
How old is the patient who has gone through 1 pregnancy?	**Row Choice**	**Unambiguous** The question clearly specifies a single condition (gone through 1 pregnancy), making it unambiguous.	**Ambiguous**. The question is ambiguous because it refers to the patient who has gone through 1 pregnancy without knowing their age and also implies that this information can be used for a prediction, which may not directly relate to the question.
Does any patient weight more than 100 pounds?	**Unit**	**Unambiguous**. The question is clear and can be answered directly.	**Ambiguous**. The question is ambiguous
Is any patient more than 50?	**Column Choice**	**Unambiguous** The question is clear and unambiguous as it can be directly answered with a simple yes or no based on the provided data.	**Ambiguous**. The question is ambiguous.
Who is the shortest patient?	**Identification**	**Unambiguous** The question can be answered directly based on the provided dataset.	**Ambiguous** The question is ambiguous because it lacks specificity regarding the patient's height
What is the most important age in the dataset for our study?	**Operational**	**Ambiguous**. The question asks for a specific age, but the dataset has multiple instances of different ages	**Ambiguous** The question asks for a specific age, but the dataset has multiple instances of different ages

Table 5: Examples of In-Context Explicit Ambiguity Detection of Different Types.

Question	Ambiguity Type	claude-3.7-sonnet	gpt-4o
Which book is considered the best by our clients?	Not Ambiguous	**Ambiguous** The question is ambiguous because there's no clear definition of what 'best' means in this context. It could refer to highest ratings, most reviews, most wished for, or some other metric of customer preference. The data shows multiple ways to interpret client preference.	**Ambiguous** The question is ambiguous because 'best' is subjective and not clearly defined in the dataset, which contains multiple metrics like ratings, reviews, wished users, and discount offers that could represent 'best'.
Provide me with a list containing all the lengths of the books about computer science.	Not Ambiguous	**Unambiguous** The question is not ambiguous because it specifically asks for the lengths of all books in the 'Computer Science & Engineering' category, which can be directly extracted from the dataset. There is only one entry in this category: 'Let Us C, 19th Edition' with a length of 492 pages.	**Unambiguous** The dataset contains only one book explicitly categorized under 'Computer Science & Engineering', making the request straightforward and unambiguous.
List the unique ages of the youngest five women who have a diabetes risk associated with their family. If there are less than five answer with all the unique ages that match the criteria.	Not Ambiguous	**Ambiguous** The question is ambiguous because it references 'diabetes risk associated with family' which seems to refer to the 'Heredity' column, but it's not clear if 'Heredity' = 1 means diabetes risk from family. Also, it's unclear if 'youngest five women' should be sorted by age only or consider other factors.	**Unambiguous** The question is not ambiguous because the dataset clearly defines 'Heredity' as family diabetes risk and lists ages, allowing identification of the youngest five women meeting the specified criteria.
Are all women in the dataset predicted to be free of diabetes? (to have diabetes is associated with the positive label of the model)	Not Ambiguous	**Unambiguous** The question is not ambiguous. The dataset's 'Prediction' column shows all values as 0, indicating none of the women are predicted to have diabetes (positive label).	**Unambiguous** The question is not ambiguous because the dataset clearly shows the Prediction column with binary values where 0 indicates free of diabetes and 1 indicates predicted to have diabetes, allowing a straightforward answer.
How many women weight less than 60 kg?	Not Ambiguous	**Unambiguous** The question is not ambiguous. The dataset clearly shows weight data for all entries, and all individuals weigh less than 60 kg (ranging from 48 to 55 kg).	**Unambiguous** The question is not ambiguous because it clearly asks for the count of women weighing less than 60 kg, which can be determined directly from the Weight column in the dataset.

Table 6: Examples of Ambiguity Detection on Unambiguous Questions.

Question	Ambiguity Type	Llama-3.1-8B	Llama-3.2-1B
Which book is considered the best by our clients?	Not Ambiguous	**Ambiguous** The question contains subjective information from the ratings field which may not be considered by all clients, and 'best' is a relative term.	**Ambiguous** The question is ambiguous because it lacks a clear and specific reference, making it impossible to determine the best book based on general criteria.
Provide me with a list containing all the lengths of the books about computer science.	Not Ambiguous	**Ambiguous** The question asks for book lengths about computer science, but the dataset does not specify which books are about computer science. The category is 'Computer Science & Engineering' for one book and there is no further information to narrow down the selection.	**Ambiguous** The question is ambiguous
List the unique ages of the youngest five women who have a diabetes risk associated with their family. If there are less than five answer with all the unique ages that match the criteria.	Not Ambiguous	**Ambiguous** The question is not clear about the criteria for a 'youngest' woman and it assumes women exist in the dataset.	**Ambiguous** The question is ambiguous because it's asking for a list of unique ages that match specific criteria (youngest five women with diabetes risk), but not explicitly defining what those criteria are.
Are all women in the dataset predicted to be free of diabetes? (to have diabetes is associated with the positive label of the model)	Not Ambiguous	**Ambiguous** The question's reference to 'women' is ambiguous because the dataset contains both males and females, but no explicit information about gender.	**Ambiguous** The question is ambiguous
How many women weight less than 60 kg?	Not Ambiguous	**Unambiguous** The question does not specify which data set 'women' refers to, but based on the provided information it seems clear that weight less than 60 kg is a condition that can be determined.	**Ambiguous** The question is ambiguous because it lacks specificity about the weight condition.

Table 7: Examples of Ambiguity Detection on Unambiguous Questions.