DataBench

A Large-Scale Empirical Evaluation of LLMs for Question Answering over Tabular Data

Jorge Osés Grijalba,[1] Luis Alfonso Ureña López,[2] Eugenio Martínez Cámara,[2] Jose Camacho Collados[3]

[1]Graphext, [2]University of Jaén, [3]Cardiff University

What we did

Question Answering over Tabular Data with **DataBench**: A Large-Scale Empirical Evaluation of LLMs.

- Currently, available Question-Answering (QA) over Tables systems are outdated in two different ways.
  - Designed for less capable models with limited reasoning ability.
  - Tested over clean and small tables.

- We propose a new benchmark for QA over Tables.
  - Based on a variety of **industry datasets** from different domains.
  - Incorporating **new types** for QA over Tables specifically.

Question Answering over Tabular Data with DataBench: A Large-Scale Empirical Evaluation of LLMs

Jorge Osés Grijalba, Luis Alfonso Ureña-López, Eugenio Martínez Cámara and Jose Camacho-Collados
Graphext, University of Jaen, Cardiff University
jorge@graphext.com, {laurena, emcamara}@ujaen.es
camachocolladosj@cardiff.ac.uk

Abstract
Large Language Models (LLMs) are showing emerging abilities, and one of the latest recognized ones deals with their ability to reason and answer questions from tabular data. Although there are some available datasets to assess question answering systems on tabular data, they are not large and diverse enough to properly assess the capabilities of LLMs. To this end, we propose DataBench, a benchmark composed of 65 real-world datasets over several domains, including 20 human-generated questions per dataset, totaling 1300 questions and answers overall. Using this benchmark, we perform a large-scale empirical comparison of several open and closed source models, including both code-generating and in-context learning models. The results highlight the current gap between open-source and closed-source models, with all types of model having room for improvement even in simple boolean questions or involving a single column.

Keywords: question answering, tabular data, llms, benchmark

1. Introduction

The advent of the era of large language models (LLMs) has revolutionized the research on natural language processing (NLP), especially since their scaling up as zero- and few-shot learners (Radford et al., 2019; Brown et al., 2020). This capacity of learning without the need to follow the standard machine learning training workflow enables the usage of task-agnostic architectures to resolve a wide range of tasks such as sentiment analysis (Deng et al., 2023; Zhang et al., 2023c), machine translation (Jiao et al., 2023) or text summarisation (Zhang et al., 2023b), to name a few. This growth has been possible partially by the continuous release of general-purpose LLMs (Yang et al., 2023) and the discovery of emergent abilities of LLMs (Wei et al., 2022). However, this incessant flux of models has not been accompanied by the release of high-quality and large-scale benchmarks for evaluating and comparing specific capacities of LLMs.

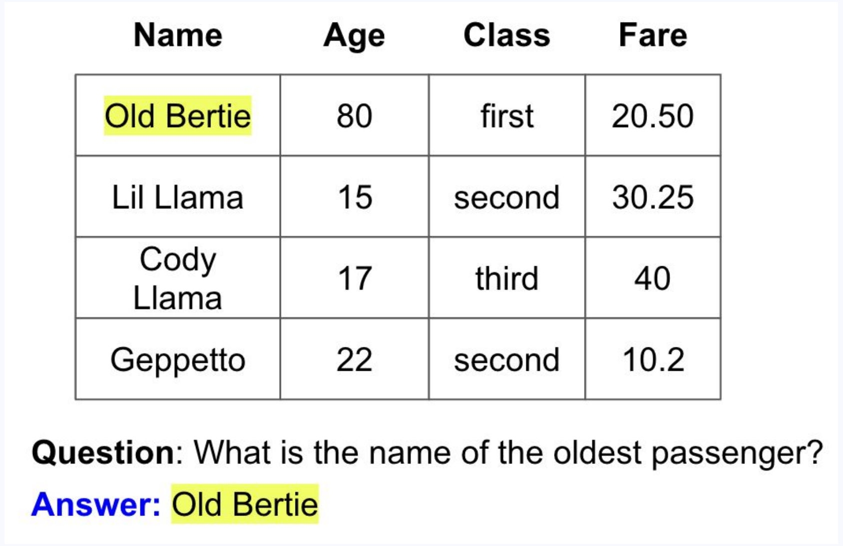
Question answering (QA) is a longstanding NLP task focused on retrieving the most adequate answer for a question on unstructured or plain text documents (Voorhees, 2001). On the other hand, structured data encompasses a great bunch of knowledge whose query which has traditionally been linked to a programmatic access by SQL or SPARK queries. However, these languages make rigid assumptions about the structured data organized in tables and are not able to understand the semantics of the textual fields. Likewise, they do not allow to make questions in natural language.

Because of this, question answering in non-database tables, structured or tabular data has attracted the interest of the research community (Pasupat and Liang, 2015; Aly et al., 2021; Nan et al., 2022), especially to leverage language models to generate appropriate queries from natural language questions (Herzig et al., 2020; Liu et al., 2022). Recently, tabular question answering has been shown as an emergent ability of LLMs (Chen, 2023). This new capacity, along with the public reliance on these models, highlight the need for a wide benchmark to reliably assess the performance of LLMs.

In this paper, we present DataBench, a large benchmark for the task of tabular question answering on structured or tabular data. We propose DataBench with the aim of providing a benchmark to evaluate and compare LLMs as tabular reasoners, but flexible to compare any other type of question answering model. Accordingly, DataBench is composed of 65 datasets from different domains, widely different numbers of rows and columns and heterogeneous data types. Moreover, DataBench has 20 hand-made questions per dataset, with a total number of 1300 questions. Questions are further split in different types depending on the type of answer (i.e., true/false, categories from the dataset, numbers or lists), and each question is accompanied by their corresponding gold standard answer. Finally, we use DataBench to evaluate the last-generation of LLMs over tabular data, including code-generating models. The results show that current models are still not fully reliable to be used on tabular data, and there is significant room for improvement for all types of question and domain.

Introduction

What do we mean by QA over data?



Name	Age	Class	Fare
Old Bertie	80	first	20.50
Lil Llama	15	second	30.25
Cody Llama	17	third	40
Geppetto	22	second	10.2

Question: What is the name of the oldest passenger?

Answer: Old Bertie

SQL

mongoDB

Google Big Query

CSV

JSON

amazon REDSHIFT

Introduction

- Question Answering (**QA**) over Tabular Data **has existed as a subfield** over the last decade, but never really taken off to rival the most popular fields within NLP.

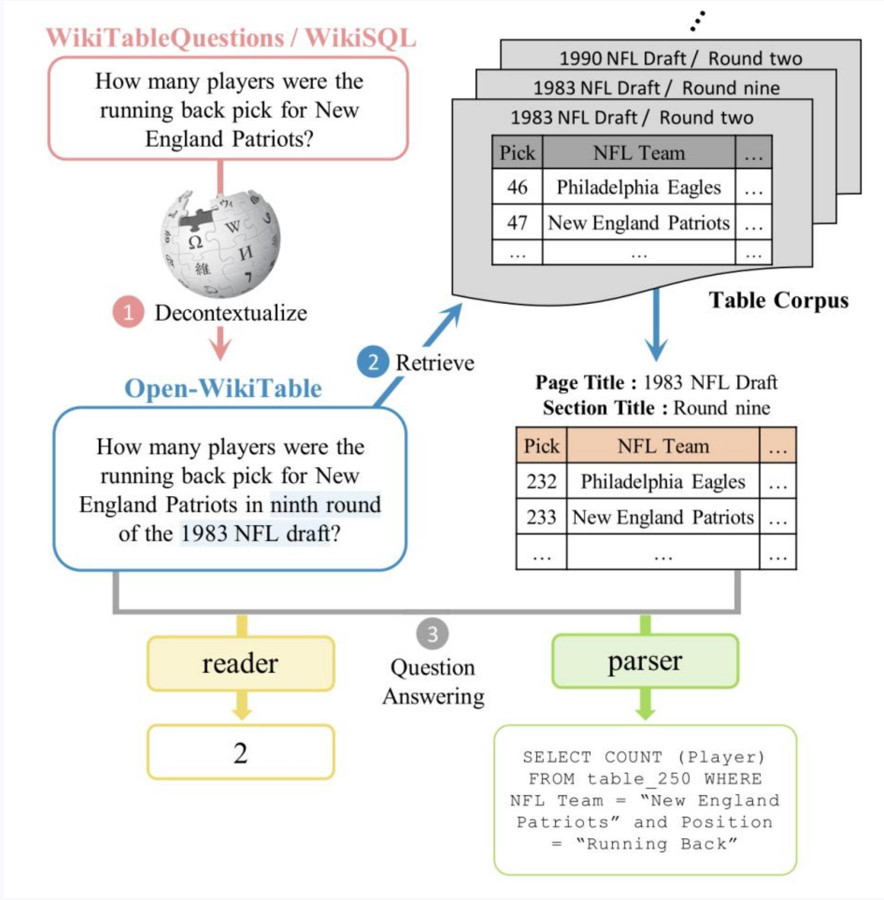
- With the coming of LLMs and their new emergent skills, this has the **potential** to change.

- These models, however powerful they are, are still **hard to evaluate** in some of their new capabilities.

- Thus arises the need for a real-world **benchmarking.**

Previous Work

- NLP applications were focused on **finding cells** containing particular information.
- Other approaches consisted on a **SQL** query-builder for retrieval.
- Work has moved towards more complex operations like **aggregations.**
- Still most of the approaches rely either on **Wikipedia** tables or very **specific** domains.

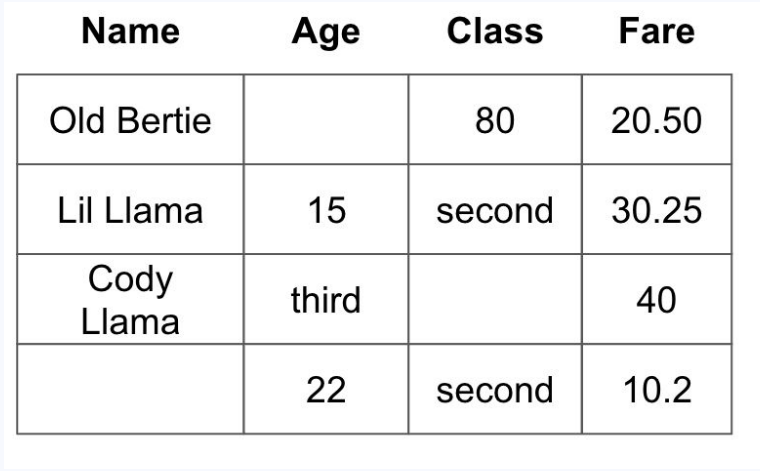

WikiTableQuestions / WikiSQL
How many players were the running back pick for New England Patriots?
1 Decontextualize
Open-WikiTable
How many players were the running back pick for New England Patriots in ninth round of the 1983 NFL draft?
2 Retrieve
1990 NFL Draft / Round two
1983 NFL Draft / Round nine
1983 NFL Draft / Round two
Pick | NFL Team | …
46 | Philadelphia Eagles | …
47 | New England Patriots | …
Table Corpus
Page Title : 1983 NFL Draft
Section Title : Round nine
Pick | NFL Team | …
232 | Philadelphia Eagles | …
233 | New England Patriots | …
3 Question Answering
reader
2
parser

```
SELECT COUNT (Player)
FROM table_250 WHERE
NFL Team = "New England
Patriots" and Position
= "Running Back"
```

Open-WikiTable : Dataset for Open Domain Question Answering with Complex Reasoning over Table (Kweon et al., 2023) 5

Modern Data
Size & Cleanliness

- Datasets have been **getting bigger**.
- Some datasets might contain **billions** of rows of **categorical** data.
- Others might be comprised of just **a few rows** with **tens of thousands** of **numerical** columns.
- They might contain **missing data**.
- Some of the data might have **inconsistent formatting**.



Name	Age	Class	Fare
Old Bertie		80	20.50
Lil Llama	15	second	30.25
Cody Llama	third		40
	22	second	10.2

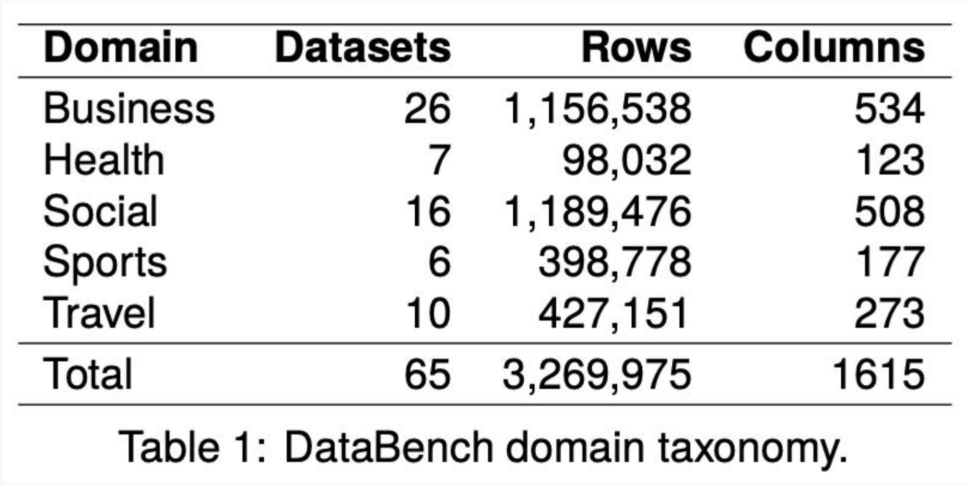
Modern Data
New possibilities

- Our interactions with databases have grown up beyond simple retrieval. We’re integrating **workflows**, doing **predictions** ...
- Since there is **more available data** in tables and databases, we need models to be able to reason over that data.
- Models are now **better**, we can now ask more of them.

Databench Collection

Real world benchmarking

- **65 different public datasets** from Graphext.
- **Initial 1300 QA pairs**.
- Covering 5 different **domains**.
- **Real life** questions.
- Not curated for format or missing data.
- We also provide a **lite version** of the collection, with **small sampled versions** of the datasets.

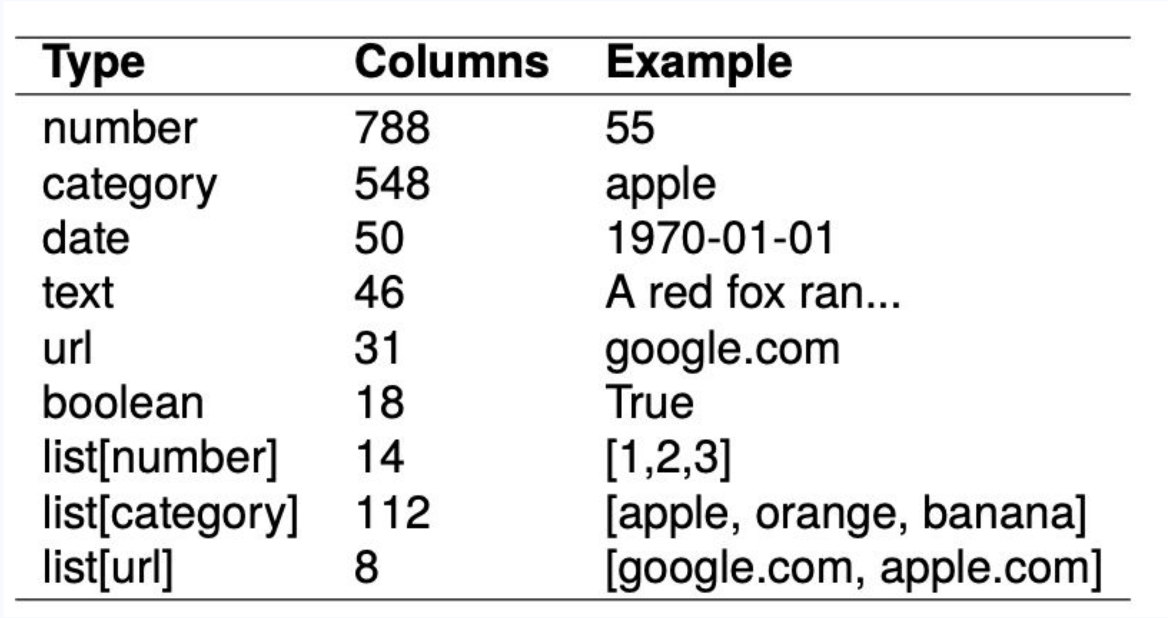


Domain	Datasets	Rows	Columns
Business	26	1,156,538	534
Health	7	98,032	123
Social	16	1,189,476	508
Sports	6	398,778	177
Travel	10	427,151	273
Total	65	3,269,975	1615

Table 1: DataBench domain taxonomy.

Databench Collection
Column types

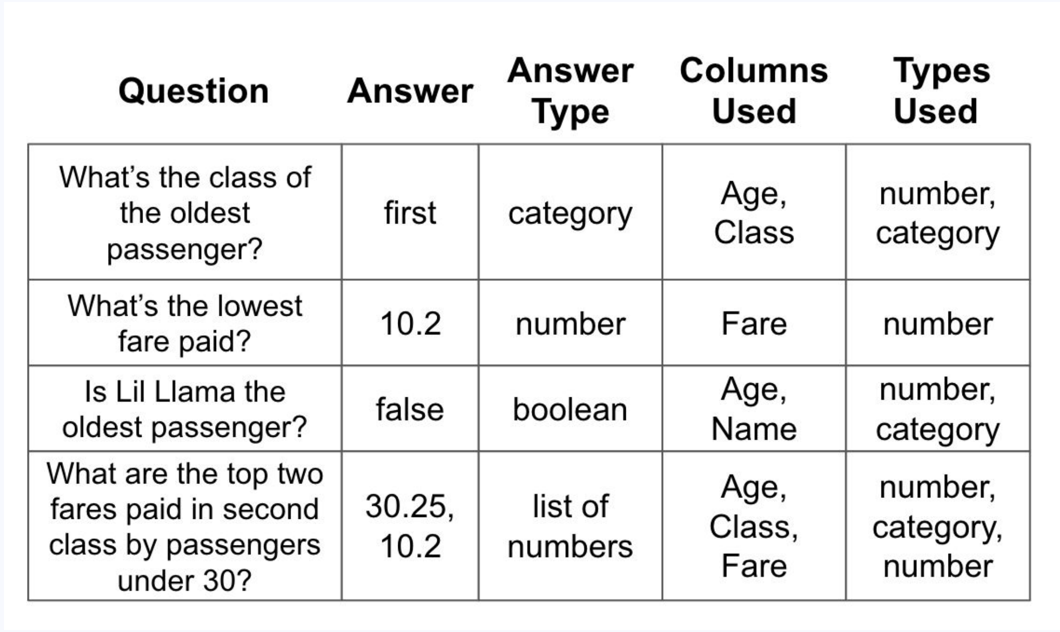
- Columns belong to 10 **different types**, inferred from each dataset.
- **Lector**: Graphext's open-source library to infer these types.
- Represent **most common** data types we find in practice.



Type	Columns	Example
number	788	55
category	548	apple
date	50	1970-01-01
text	46	A red fox ran...
url	31	google.com
boolean	18	True
list[number]	14	[1,2,3]
list[category]	112	[apple, orange, banana]
list[url]	8	[google.com, apple.com]

Databench Collection
Answer types

- Question **categorization**: 5 types according to the **type of the answer**.
- We tag them with the **columns to use** for the answer and the **types of the columns used**.
- Keeping the questions as **factoids** for now allows us to streamline evaluation.
- Only use **one dataset** per question.



Question	Answer	Answer Type	Columns Used	Types Used
What's the class of the oldest passenger?	first	category	Age, Class	number, category
What's the lowest fare paid?	10.2	number	Fare	number
Is Lil Llama the oldest passenger?	false	boolean	Age, Name	number, category
What are the top two fares paid in second class by passengers under 30?	30.25, 10.2	list of numbers	Age, Class, Fare	number, category, number

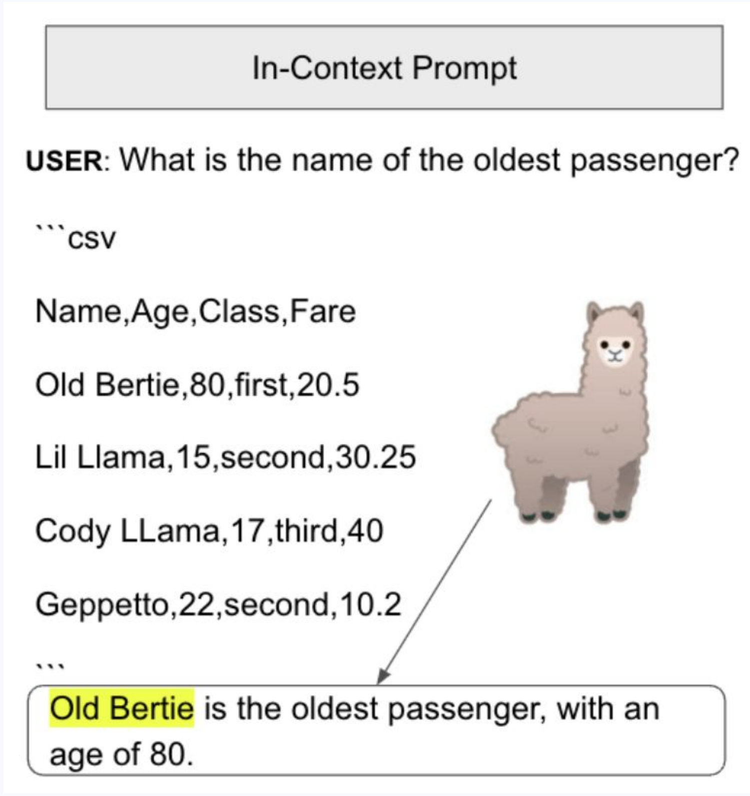
Experiments
Notes on modern LLMs

- LLMs have a limited **context window** that they can understand, and we have to fit a larger representation into them.
- The whole dataset **will probably not fit**.
- If the model doesn't have **enough information**, we cannot expect it to answer well. How do we do it?
    - We'll follow **two different prompting strategies**
    - We'll be using the **smaller versions** of the 65 datasets

Experiments
In-Context Prompting Strategy

Two different **In-Context** prompts

- If it fits, go for it. We'll call this **Zero Shot In-Context Learning**
- In theory we're providing it with the whole information, but it's unclear how much the model **will be able to do** with this format.
- For **Zero Shot In-Context Learning** we'll do two prompts: one asking for the **answer** and **columns used**, and another one asking for an **additional explanation**.


In-Context Prompt

USER: What is the name of the oldest passenger?

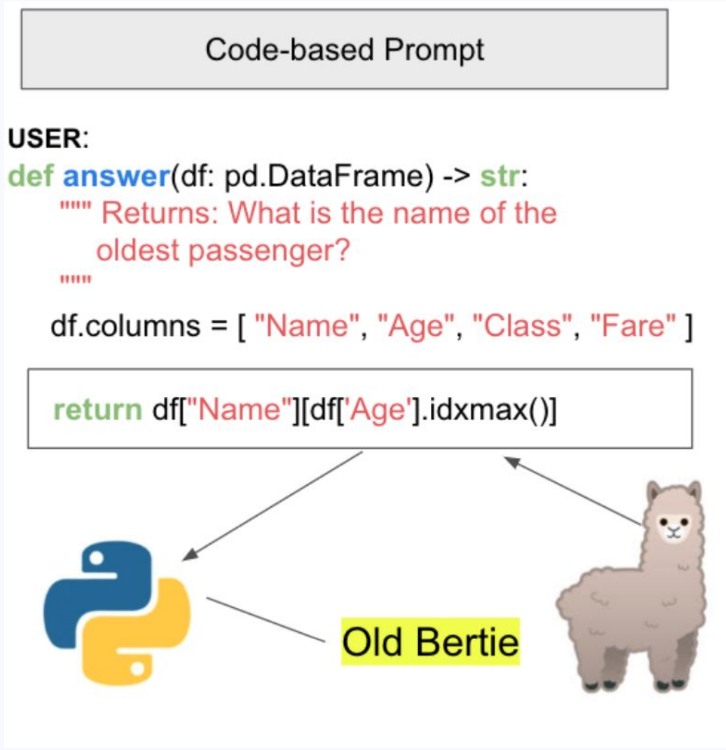
~~~
```csv
Name,Age,Class,Fare
Old Bertie,80,first,20.5
Lil Llama,15,second,30.25
Cody LLama,17,third,40
Geppetto,22,second,10.2
```
~~~

Old Bertie is the oldest passenger, with an age of 80.

Experiments

Code-based Prompting Strategy

- LLMs can generate code, which in turn allows **accountability**.
- On the other hand, we're not providing the model with enough data, only a **representation** for it.
- We will turn the problem of QA over tables into one of **code completion.**
- We're providing our first coding prompt with **only the column names,** and for our second prompt, we'll also be providing the **column types.**


Code-based Prompt

USER:

```
def answer(df: pd.DataFrame) -> str:
    """ Returns: What is the name of the
       oldest passenger?
    """

  df.columns = [ "Name", "Age", "Class", "Fare" ]

   return df["Name"][df['Age'].idxmax()]
```

Experiments

Recap

- We will be testing **1300 QA** pairs over the sampled versions of **65 real-world datasets**.
- These QA pairs are tagged with the **type** of the answer and which **columns** were used.
- We'll be using **two** prompting approaches.
    - One fits the whole dataset **in the prompt**, asks for answer, columns used (**Z-ICL Prompt 1**) and an **explanation** (**Z-ICL Prompt 2**).
    - The other relies on **code completion**, by providing the column names (**Code Prompt 1**) and **types** (**Code Prompt 2**).
- We'll be evaluating both percentage of **correct format** generation and **accuracy**.

Experiments
Models used

Z-ICL	LLaMa-2-7b-chat	LLaMa-2-13b-chat	chatgpt-3.5-turbo
Code	Codellama-7b	Codellama-13b	chatgpt-3.5-turbo

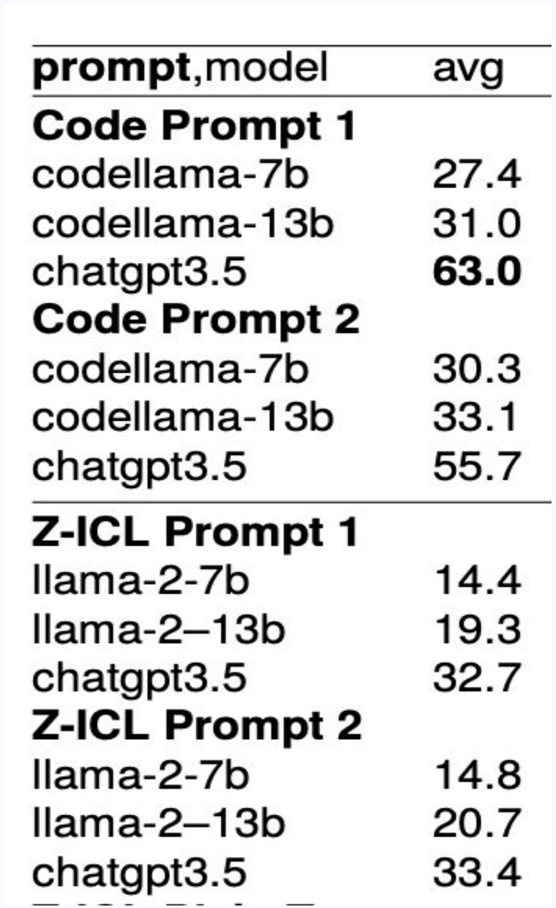
Results

Overall

- **Code-based** approaches in general have **way better accuracy** than **Z-ICL** over all data categories.

- This is weak evidence, it **doesn’t mean** that it’s always the case.

- The **more parameters** the model has, the **better** the **results** are. This is universal across all models and question types.

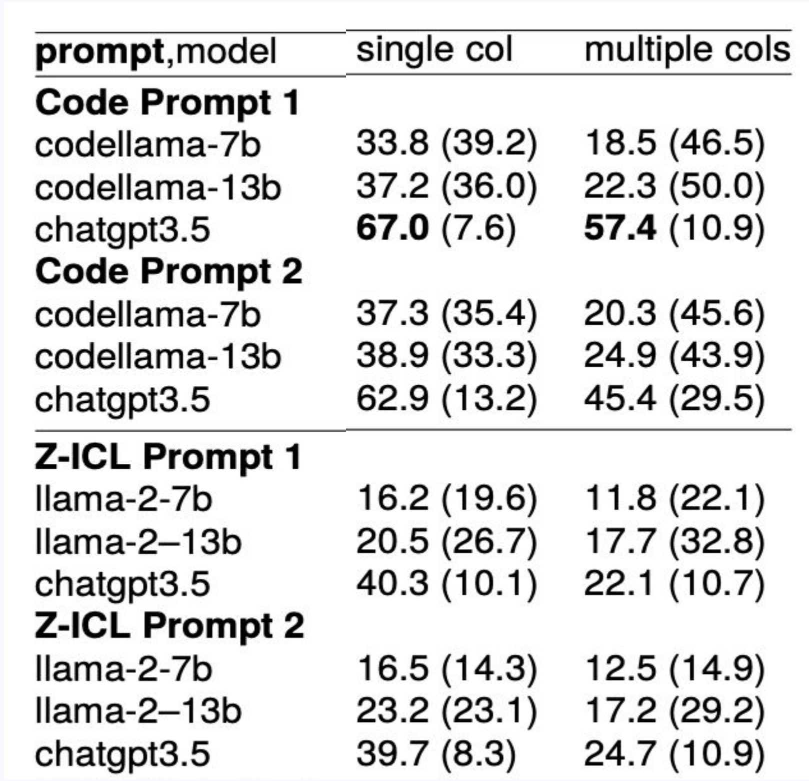


prompt,model	avg
Code Prompt 1	
codellama-7b	27.4
codellama-13b	31.0
chatgpt3.5	**63.0**
Code Prompt 2	
codellama-7b	30.3
codellama-13b	33.1
chatgpt3.5	55.7
Z-ICL Prompt 1	
llama-2-7b	14.4
llama-2–13b	19.3
chatgpt3.5	32.7
Z-ICL Prompt 2	
llama-2-7b	14.8
llama-2–13b	20.7
chatgpt3.5	33.4

Results

Columns Used

- Single column is **generally easier to answer**.
- Models present both **lower accuracy** and **higher format error** when dealing with questions that require multiple columns.
- This happens in **both** Z-ICL and code-based prompt approaches.

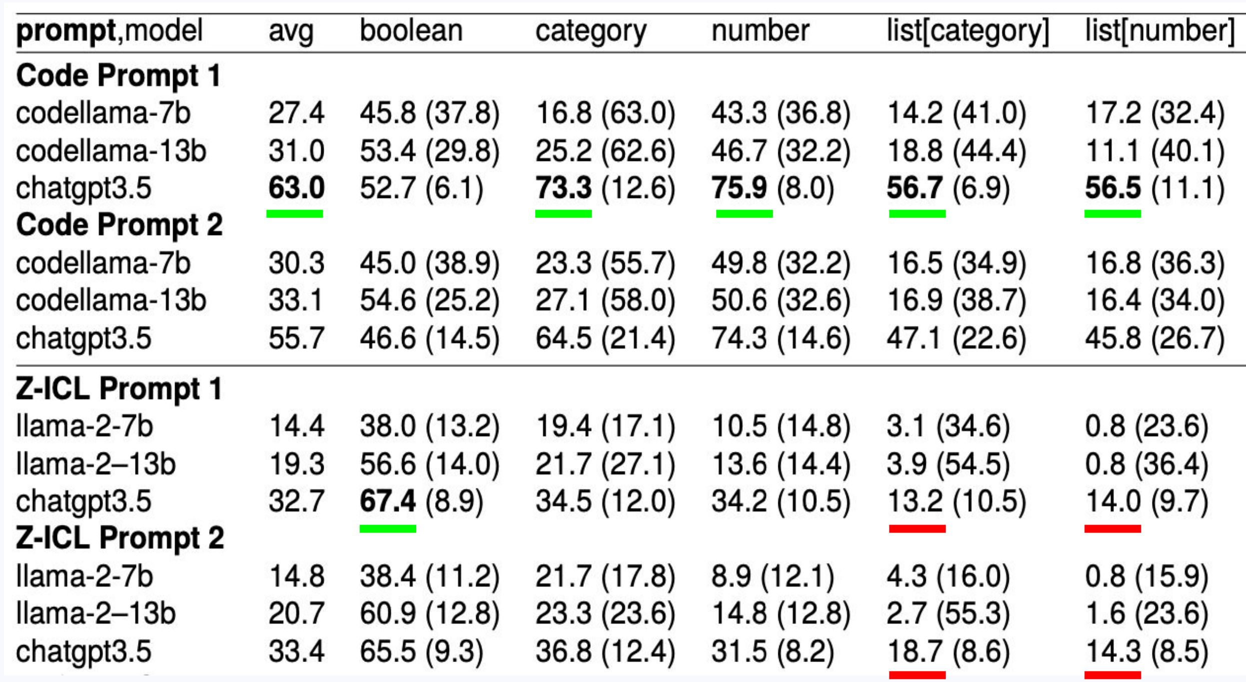


prompt,model	single col	multiple cols
Code Prompt 1		
codellama-7b	33.8 (39.2)	18.5 (46.5)
codellama-13b	37.2 (36.0)	22.3 (50.0)
chatgpt3.5	**67.0** (7.6)	**57.4** (10.9)
Code Prompt 2		
codellama-7b	37.3 (35.4)	20.3 (45.6)
codellama-13b	38.9 (33.3)	24.9 (43.9)
chatgpt3.5	62.9 (13.2)	45.4 (29.5)
Z-ICL Prompt 1		
llama-2-7b	16.2 (19.6)	11.8 (22.1)
llama-2–13b	20.5 (26.7)	17.7 (32.8)
chatgpt3.5	40.3 (10.1)	22.1 (10.7)
Z-ICL Prompt 2		
llama-2-7b	16.5 (14.3)	12.5 (14.9)
llama-2–13b	23.2 (23.1)	17.2 (29.2)
chatgpt3.5	39.7 (8.3)	24.7 (10.9)

Results

Types

- **Code Prompt 1** is generally the best approach.
- Code-based approaches are generally better.
- Booleans are **harder than we thought of** with this approach.
- Could use better prompts so the smaller models learn the list format.

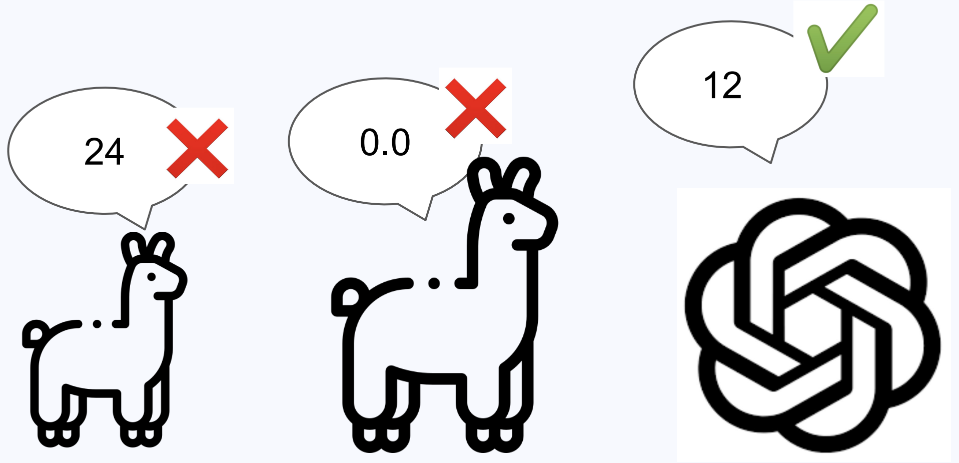


prompt,model	avg	boolean	category	number	list[category]	list[number]
Code Prompt 1						
codellama-7b	27.4	45.8 (37.8)	16.8 (63.0)	43.3 (36.8)	14.2 (41.0)	17.2 (32.4)
codellama-13b	31.0	53.4 (29.8)	25.2 (62.6)	46.7 (32.2)	18.8 (44.4)	11.1 (40.1)
chatgpt3.5	**63.0**	52.7 (6.1)	**73.3** (12.6)	**75.9** (8.0)	**56.7** (6.9)	**56.5** (11.1)
Code Prompt 2						
codellama-7b	30.3	45.0 (38.9)	23.3 (55.7)	49.8 (32.2)	16.5 (34.9)	16.8 (36.3)
codellama-13b	33.1	54.6 (25.2)	27.1 (58.0)	50.6 (32.6)	16.9 (38.7)	16.4 (34.0)
chatgpt3.5	55.7	46.6 (14.5)	64.5 (21.4)	74.3 (14.6)	47.1 (22.6)	45.8 (26.7)
Z-ICL Prompt 1						
llama-2-7b	14.4	38.0 (13.2)	19.4 (17.1)	10.5 (14.8)	3.1 (34.6)	0.8 (23.6)
llama-2–13b	19.3	56.6 (14.0)	21.7 (27.1)	13.6 (14.4)	3.9 (54.5)	0.8 (36.4)
chatgpt3.5	32.7	**67.4** (8.9)	34.5 (12.0)	34.2 (10.5)	13.2 (10.5)	14.0 (9.7)
Z-ICL Prompt 2						
llama-2-7b	14.8	38.4 (11.2)	21.7 (17.8)	8.9 (12.1)	4.3 (16.0)	0.8 (15.9)
llama-2–13b	20.7	60.9 (12.8)	23.3 (23.6)	14.8 (12.8)	2.7 (55.3)	1.6 (23.6)
chatgpt3.5	33.4	65.5 (9.3)	36.8 (12.4)	31.5 (8.2)	18.7 (8.6)	14.3 (8.5)

Results

Code Prompt 1: All the model knows is the name of the column

Siblings_Spouses Aboard
0
1
0
0



How many passengers boarded without any siblings or spouses?

7B

```
df['IsAlive'] = np.where(df['Survived'] == 0, False, True)
df['IsNotOnBoat'] = np.where((df['Siblings_Spouses Aboard'] == 0) & (df['Pclass'] == 3), True, False)
return len(df[df['IsAlive'] == True]) * len(df[df['IsNotOnBoat'] == True])
```

13B

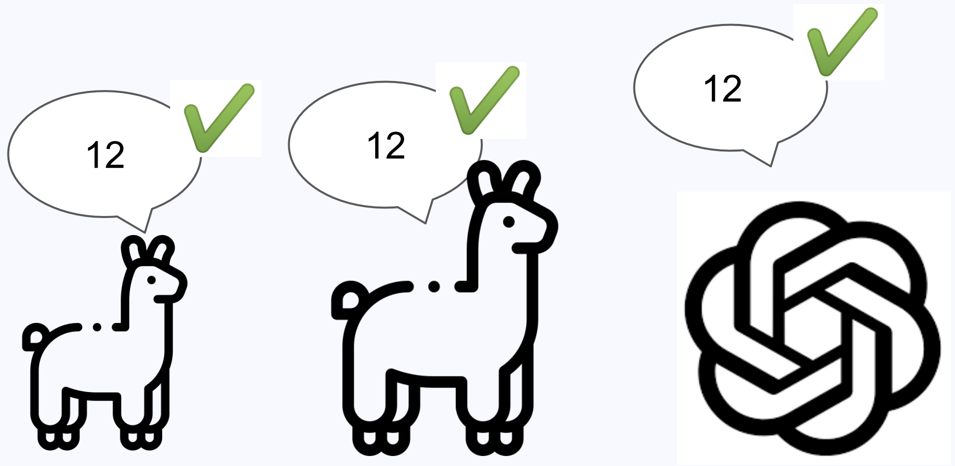
```
return len(df[df['Siblings_Spouses Aboard'].isnull()])
```

```
return len(df[df['Siblings_Spouses Aboard'] == 0])
```

Results

Code Prompt 2: the model knows the types

Siblings_Spouses Aboard
0
1
0
0



How many passengers boarded without any siblings or spouses?

7B

```
df_subset = df.loc[df['Siblings_Spouses Aboard'] == 0]
return len(df_subset)
```

13B

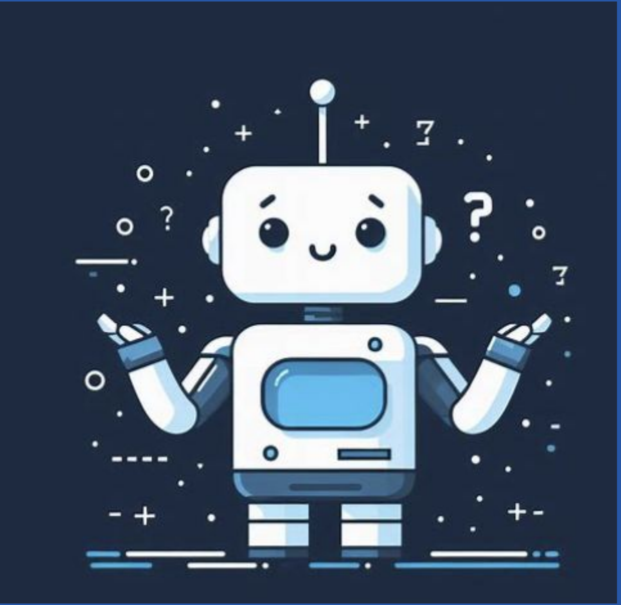
```
return len(df[~df['Siblings_Spouses Aboard'].astype(bool)])
```

```
return len(df[df['Siblings_Spouses Aboard'] == 0])
```

Conclusions

- A type categorization works well enough to diagnose model performance, allowing patterns to emerge.
- There seem to be **differences in model performance** for different data and **question types**.
- **Code approaches work better** in general, they are more explainable and **easier to evaluate**.
- In-Prompt approaches tend to **hallucinate** when giving explanations.

Future Work



- Finding the **right prompting strategies** for evaluation.
- Figuring out **a way out of factoids** that allows for automated evaluation.
- Long term: move from benchmarking towards **fine-tuning a model** specifically for QA over tables.

Thank you!

- You can reach me at jorgeosesgrijalba@gmail.com
- You can further explore DataBench at 🤗
- https://huggingface.co/datasets/cardiffnlp/databench
- You can make a free account and explore all the datasets (and upload your own!) at **graphext.com**

DataBench URL