CSC306- Natural Language Processing Final Project Plan

Baibhav Barwal, Kurtik Appadoo and Neil Daterao

Abstract

Large Language Models (LLMs) struggle with reasoning over tabular data, particularly in multi-column queries and boolean questions. This project replicates the DataBench study (LREC-COLING 2024) to evaluate LLM performance on tabular question answering. Using 65 real-world datasets and 1,300 questions, we will compare two approaches: In-Context Learning (ICL)- where models answer questions directly from tables, and Code-Based Question Answering-where models generate Python code to extract answers. By verifying the original findings and testing additional models, we aim to provide insights into LLM limitations in structured data reasoning and inform future improvements.

1 Introduction

Large Language Models (LLMs) have transformed natural language processing (NLP), yet their ability to handle structured tabular data remains limited. Unlike traditional NLP tasks, tabular question answering (QA) requires numerical reasoning, multi-column extraction, and structured data processing.

Osés Grijalba et al. (2024) found that even state-of-the-art models struggle with tabular QA, particularly in boolean questions, multi-column reasoning, and output formatting. The DataBench benchmark highlights the performance gap between open-source models (e.g., LLaMA, CodeLlama) and proprietary models (e.g., ChatGPT), revealing challenges in structured data comprehension.

This project replicates and extends the DataBench study by evaluating LLMs on 65 datasets. We employ two approaches: In-Context Learning (ICL), where models answer questions directly from tables, and Code-Based QA, where models generate Python code to extract answers.

By validating previous findings and assessing additional models, this study aims to explore LLM limitations in tabular data reasoning and contribute to future advancements in structured data processing.

2 Related/ Prior Work

This project builds upon the work of Osés Grijalba et al. in "Question Answering Over Tabular Data with DataBench" (LREC-COLING 2024). Their paper introduces DataBench, a benchmark consisting of 65 real-world datasets and 1,300 questions spanning diverse domains such as finance, health, and sports.

The study evaluates large language models (LLMs) using two distinct approaches:

1. **In-Context Learning (ICL)** – Supplying the model with a table alongside a direct question.
2. **Code-Based Approach** – Prompting the model to generate Python code that extracts the answer from the given table.

The results indicate that LLMs struggle with tabular data, particularly in multi-column reasoning, boolean questions, and formatting outputs. Moreover, open-source models (e.g., LLaMA, CodeLlama) consistently underperform compared to proprietary ones (e.g., ChatGPT).

This project aims to replicate their study, verify its findings, and potentially evaluate additional models.

3 Data and Software

This project is based on **DataBench**, a benchmark designed for Question Answering (QA) over tabular data. It consists of **65 real-world tabular datasets** spanning domains such as business, health, social networks, sports, and travel. Each dataset includes **20 manually generated questions**, totaling **1,300 questions**, covering various answer types:

- Boolean (True/False)

- Categorical values
- Numerical values
- Lists of categories
- Lists of numbers

To accommodate models with different context window sizes, DataBench is available in two versions:

- **Full version**: Contains the complete dataset with all rows and columns.
- **DataBench_lite**: A reduced version with only *20 rows per dataset*.

The dataset is **publicly available** on Hugging Face.

The study evaluates Large Language Models (LLMs) using two approaches:

- **In-Context Learning (ICL)**: The dataset is formatted as a CSV table and included in a prompt, where the model generates an answer in a structured JSON format.
- **Code-Based Approach**: The model generates Python code to extract answers from a Pandas DataFrame.

Models tested include *LLaMA-2 (7B & 13B)*, *CodeLlama (7B & 13B)*, and *ChatGPT-3.5*. Required libraries include **pandas**, **numpy**, **transformers**, and **llama.cpp**.

To replicate this work, access to **OpenAI API** is needed for closed-source models, while open-source models require local deployment or cloud access. Efficient execution requires GPU resources such as an *NVIDIA A100 or RTX 3090*. The DataBench dataset is freely available, and dependencies can be installed via `pip`. Open-source LLMs can be deployed locally, while API access must be verified for ChatGPT usage.

4 Task Plan

4.1 Week 1: Data Preparation and Model Implementation

The first week will focus on setting up the environment, downloading the dataset, and implementing the models. To begin, we will install the necessary Python libraries, including `pandas`, `numpy`, `transformers`, and `llama-cpp-python`, and verify access to the OpenAI API for closed-source models such as ChatGPT-3.5. GPU or cloud resources will be set up for running open-source models like LLaMA and CodeLlama.

Next, we will prepare the dataset by downloading DataBench from Hugging Face, categorizing question types, and preprocessing the data to ensure it is correctly structured for both in-context learning and code-based answering. The implementation phase involves developing two approaches. For **In-Context Learning (ICL)**, we will format datasets as CSV tables, design prompts, and store model-generated answers in JSON format. For the **Code-Based Answering** approach, we will prompt LLMs to generate Python code for extracting answers, ensuring execution correctness while handling syntax and execution errors.

Initial testing will involve running test queries on small datasets to validate model responses. We will compare outputs across different models and approaches, addressing any formatting, missing values, or execution issues.

4.2 Week 2: Model Evaluation and Analysis

The second week will focus on evaluating model performance, analyzing errors, and documenting findings. Performance evaluation will compare accuracy across models and question types, assess responses from both ICL and Code-Based approaches, and measure formatting errors, hallucinations, and reasoning failures. Failure analysis will identify challenges in boolean, multi-column, and list-based queries while documenting cases where models retrieve correct answers from incorrect columns. Common errors such as syntax issues in generated code or incorrect JSON formatting will also be recorded.

Finally, we will summarize key findings, including accuracy results and error analysis, and prepare a structured report detailing the experimental setup, methodologies, and results. A project presentation will be created to highlight key insights and performance comparisons. All scripts and datasets used will be documented and uploaded for reproducibility.