LLMs Automatically Generate Code to Data Clean

Nam Huynh
nam@ou.edu
University of Oklahoma
Norman, USA

Beiyu Lin
beiyu.lin@ou.edu
University of Oklahoma
Norman, USA

I. Abstract

Data cleaning commonly requires highly-trained experts with knowledge in both coding and algorithms to handle its problems and challenges, such as imputing missing values, correcting data types, handling duplicates, standardizing format, and schema matching. As data cleaning has been extensively and commonly used in diverse disciplines ranging from healthcare to transportation, we want to design technology to open up the possibility of data mining to everyone. This can be achieved by utilizing a Large Language Model-based interface (e.g., ChatGPT) to automatically generate codes that would assist non-experts in doing data cleaning in their fields. We start by applying ChatGPT to automatically generate code for the two basic tasks in datasets: Input-Output specification and Format Standardization. For this preliminary study, we use a real-life traffic flows in Las Vegas parks, mainly used in the field of Architecture and City Landscape Design for domain experts with very limited coding knowledge and capacities. By applying ChatGPT version 4.0, we can successfully clean the data in term of the two specific tasks. Our results show that ChatGPT can significantly help with the data cleaning process for people without rich coding knowledge and offer the potential for data mining to everyone.

II. Introduction

According to a report by IDC, the global data sphere is expected to grow from 33 zettabytes in 2018 to 175 zettabytes by 2025 because of various sources, including the Internet of Things (IoT) and social media [1]. The exponential growth of data globally can cause problems related to data quality, storage memory, etc. This challenge highlights the critical need for data cleaning to ensure the accuracy and consistency of the data. In parallel with data growth, the development of large language models (LLMs) has shown great potential for automating data cleaning tasks through human communicable and expressive natural languages to help AI models understand the tasks and generate the corresponding code. This enables tremendous amounts of users with diverse coding capacities ranging from zero knowledge to experts. For instance, an LLM model called Codex, which powers Github Copilot to assist users with its code-writing capabilities, has automatically generated code for various tasks including code generation from natural language descriptions [2]. Users can utilize this model to specify their data cleaning tasks and obtain the code for them.

Data cleaning is a crucial step in data preparation for the processing phase of any data analysis or machine learning project. To address the need for data cleaning, we apply an approach that utilizes LLMs to automate code generation through two main data cleaning tasks: Input-Output Specification and Data Standardization. The approach focuses on these tasks to simplify the data cleaning process and establish a basic framework for future research.

III. Related Work

Researchers have studied and developed various methods and approaches to improve the code generated by LLMs as well as their results after applying the automatically generated code to the data. For example, GIFT4CODE is introduced to help improve how well LLMs understand our instructions. GIFT4CODE utilizes prompt engineering technique by showing desired examples with several specific conditions such as the output structure or format via natural language instructions[3]. This method ensures the generated code matches the desired output with a higher success rate. However, the quality of examples and imperfect data can lead to incorrect code generation.

Researchers have also investigated a solution for simplifying and automating the data standardization process, a crucial task in data cleaning [4]. A method integrates a Python library - Dataprep.Clean, which provides unified, declarative APIs for standardizing column types with minimal code. This method combined with LLM-base agents, called CleanAgent, automates the entire standardization process [4]. This uses natural language instructions to generate and execute the necessary code, reducing the need for human intervention. However, this method depends heavily on the performance of LLMs

In addition to these methods, prompt engineering, a technique that guides LLMs to generate accurate and relevant outputs, can cause problems as the limitations in scripts can affect how well LLMs understand and interpret given instructions. Researchers evaluated the performance of GPT models on a name matching task by testing various name pair differences such as order, case, nicknames, and middle names [5]. This example highlights how prompt engineering can significantly enhance the accuracy and reliability of LLMs' outputs. Therefore, through detailed instructions, we can guide LLMs to generate code with the desired output.

IV. Experiments and Results

For this preliminary study, we apply LLMs to automatically generate the code for data cleaning tasks for a real-life dataset that is usually processed by domain experts with very limited coding knowledge and capacities. It would be time consuming, expensive, and infeasible for domain experts to process the large dataset. We hope that our preliminary study and result would pave the way for us to dive deep into the next step. We perform the below two basic parts in data cleaning:

- **Experiment 1: Input-Output Specification:** Specify how the given data input should be structured and how the data output should look like.
- **Experiment 2: Data standardization:** Convert data into a consistent and desired format.

We use the raw data recording the number of visits to Las Vegas's parks. It contains 340,885 rows with 78 columns. Due to the large data size, several challenges arise such as the high volume of data making manual cleaning time-consuming, inconsistent and unwanted data formats, etc.

In these experiments, we compare the outputs of code generated from an LLM model - ChatGPT version 4.0 with the outputs manually corrected as ground truth data. For Experiment 1, we extract data from 4 specific columns including 'date_range_start', 'date_range_end', 'location_name', and 'visits_by_day'. Figures 1 and 2 present an illustration of 340,885 rows and 4 chosen columns from the given data file. These figures show identical results of the illustration for extracted data. Experiment 2 changes all extracted data's date format from 'YYYY-MM-DD' to 'MM/DD/YYYY'. Figure 3 shows the format changed in date columns 'date_range_start' and 'date_range_end' as both outputs demonstrate the same data information.

```
PS C:\Users\NAMHL\Desktop\Research> python .\LLMcodefordataclean.py
  date_range_start date_range_end       location_name                 visits_by_day
0       2022-12-26     2023-01-02       Tree Top Park               [6,2,2,4,3,4,0]
1       2022-12-26     2023-01-02       Tree Top Park               [6,2,2,4,3,4,0]
2       2022-12-26     2023-01-02       Tree Top Park               [6,2,2,4,3,4,0]
3       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
4       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
5       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
6       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
7       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
8       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
9       2022-12-26     2023-01-02  Mountain Crest Park  [440,401,375,380,333,353,99]
```

Fig. 1. Our code's output on VS code for Experiment 1

date_range_start	date_range_end	location_name	visits_by_day
2022-12-26	2023-01-02	Tree Top Park	[6, 2, 2, 4, 3, 4, 0]
2022-12-26	2023-01-02	Tree Top Park	[6, 2, 2, 4, 3, 4, 0]
2022-12-26	2023-01-02	Tree Top Park	[6, 2, 2, 4, 3, 4, 0]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]
2022-12-26	2023-01-02	Mountain Crest Park	[440, 401, 375, 380, 333, 353, 99]

Fig. 2. ChatGPT 4.0 generated code's output for Experiment 1

```
PS C:\Users\NAMHL\Desktop\Research> python .\LLMcodefordataclean.py
  date_range_start date_range_end        location_name                 visits_by_day
0       12/26/2022     01/02/2023        Tree Top Park               [6,2,2,4,3,4,0]
1       12/26/2022     01/02/2023        Tree Top Park               [6,2,2,4,3,4,0]
2       12/26/2022     01/02/2023        Tree Top Park               [6,2,2,4,3,4,0]
3       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
4       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
5       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
6       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
7       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
8       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
9       12/26/2022     01/02/2023  Mountain Crest Park  [440,401,375,380,333,353,99]
```

date_range_start	date_range_end	location_name	visits_by_day
12/26/2022	01/02/2023	Tree Top Park	[6,2,2,4,3,4,0]
12/26/2022	01/02/2023	Tree Top Park	[6,2,2,4,3,4,0]
12/26/2022	01/02/2023	Tree Top Park	[6,2,2,4,3,4,0]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]
12/26/2022	01/02/2023	Mountain Crest Park	[440,401,375,380,333,353,99]

Fig. 3. Our code's output vs ChatGPT 4.0 generated code's output after Experiment 2

V. Acknowledgments

This work is funded by the University of Oklahoma. We also thank Dr. Xiwei Shen at the University of Nevada Las Vegas for sharing the dataset.

References

[1] Reinsel, D., Gantz, J., Rydning, J. (2018). The Digitization of the World From Edge to Core. IDC Whitepaper, sponsored by Seagate. https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

[2] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374.

[3] Yeming Wen, Pengcheng Yin, Kensen Shi, Henryk Michalewski, Swarat Chaudhuri, and Alex Polozov. 2024. Grounding Data Science Code Generation with Input-Output Specifications. arXiv preprint arXiv:2402.08073.

[4] Danrui Qi and Jiannan Wang. 2024. CleanAgent: Automating Data Standardization with LLM-based Agents. arXiv preprint arXiv:2403.08291.

[5] Todd. 2023. An Exploration of Using LLMs to Automate a Data Cleaning Task. Medium. https://medium.com/@todd_76597/an-exploration-of-using-llms-to-automate-a-data-cleaning-task-b99c271c71c2.