Large Language Models as Data Preprocessors

[1]Haochen Zhang, [2]Yuyang Dong, [1,3]Chuan Xiao, [2]Masafumi Oyamada
[1]Osaka University, [2]NEC Corporation, [3]Nagoya University
{chou.koushin,chuanx}@ist.osaka-u.ac.jp,{dongyuyang,oyamada}@nec.com

ABSTRACT

Large Language Models (LLMs), typified by OpenAI's GPT, have marked a significant advancement in artificial intelligence. Trained on vast amounts of text data, LLMs are capable of understanding and generating human-like text across a diverse range of topics. This study expands on the applications of LLMs, exploring their potential in data preprocessing, a critical stage in data mining and analytics applications. Aiming at tabular data, we delve into the applicability of state-of-the-art LLMs such as GPT-4 and GPT-4o for a series of preprocessing tasks, including error detection, data imputation, schema matching, and entity matching. Alongside showcasing the inherent capabilities of LLMs, we highlight their limitations, particularly in terms of computational expense and inefficiency. We propose an LLM-based framework for data preprocessing, which integrates cutting-edge prompt engineering techniques, coupled with traditional methods like contextualization and feature selection, to improve the performance and efficiency of these models. The effectiveness of LLMs in data preprocessing is evaluated through an experimental study spanning a variety of public datasets. GPT-4 emerged as a standout, achieving 100% accuracy or F1 score on 4 of these datasets, suggesting LLMs' immense potential in these tasks. Despite certain limitations, our study underscores the promise of LLMs in this domain and anticipates future developments to overcome current hurdles.

VLDB Workshop Reference Format:
[1]Haochen Zhang, [2]Yuyang Dong, [1,3]Chuan Xiao, [2]Masafumi Oyamada. Large Language Models as Data Preprocessors. VLDB 2024 Workshop: Tabular Data Analysis Workshop (TaDA).

1 INTRODUCTION

Large Language Models (LLMs), such as OpenAI's GPT and Meta's LLaMA, are becoming an increasingly important aspect of the AI landscape. These models, essentially ML systems, are trained on vast amounts of text data and characterized by an augmented number of parameters. They are capable of understanding and generating text across a diverse range of topics, thereby finding applications in numerous tasks. Consequently, research involving LLMs has garnered significant attention from both academia and industry. Recent endeavors have successfully leveraged LLMs for data management and mining. For instance, LLMs have been used for SQL generation [16], database diagnosis [5], data wrangling [12], and data analytics [2].

Proceedings of the VLDB Endowment. ISSN 2150-8097.

This paper investigates the potential of utilizing state-of-the-art (SOTA) LLMs for data preprocessing, a crucial step that refines data before it can be harnessed for downstream data mining and analytics applications. Given their comprehensive understanding of language semantics and structures, LLMs can identify errors or matches in text data. For example, they are capable of detecting spelling mistakes, grammar issues, contextual discrepancies, and near-duplicate records. Consequently, the application of LLMs in data preprocessing can pave the way for tackling tasks such as error detection, data imputation, schema matching, and entity matching.

While LLMs hold considerable potential for data preprocessing tasks, it is critical to comprehend their capabilities and limitations for effective application. Thus, as a preliminary study on employing LLMs for data preprocessing, this paper provides the following contributions.

(1) We examine the inherent knowledge and superior reasoning and learning abilities of LLMs, which can be further enhanced through zero- and few-shot prompting. These strengths position LLMs as competitive candidates for various data processing tasks. However, their computational expense and potential inefficiencies present challenges. We provide an analysis of these strengths and limitations in the context of data preprocessing.

(2) We propose a framework for LLM-based data preprocessing. This framework integrates a series of SOTA prompt engineering techniques, including zero-shot instructions, few-shot examples, batch prompting, as well as traditional approaches such as contextualization and feature selection. We specifically instruct LLMs to follow an answer format and reason before providing an answer to enhance performance. Few-shot examples are used to condition LLMs so that they can learn error criteria, means of imputation, matching conditions, etc. Batch prompting amalgamates multiple data instances in a prompt to reduce token and time costs.

(3) We conduct experiments on 12 datasets for four data preprocessing tasks. We evaluate popular LLMs such as GPT-3.5, GPT-4, and GPT-4o. The results indicate that GPT-4 generally outperforms existing solutions, achieving 100% accuracy or F1 score on 4 out of 12 datasets. GPT-3.5 also delivers competitive performance and is recommended for data preprocessing. GPT-4o delivers inconsistent performance: competitive on data imputation and entity matching but mediocre on error detection and schema matching. The evaluation also sheds light on the effects of the proposed components of the solution framework on accuracy and efficiency.

2 PRELIMINARIES

2.1 Data Preprocessing

In this initial exploration of large language models (LLMs) for data preprocessing, we concentrate on tabular data. We target the following tasks: error detection (ED), data imputation (DI), schema matching (SM), and entity matching (EM). Other typical data preprocessing tasks, such as data fusion and data wrangling, are reserved for future work. Diverging from the traditional definition that presents the entire dataset and finds or fixes all the errors (or matches, etc.) within, we define the problem by handling one record (or a pair) at a time, so the prompt to an LLM can be easily written. We term each input object a *data instance*, i.e., a tuple for ED and DI, a pair of attributes for SM and a pair of tuples for EM.

2.2 Large Language Models

LLMs have become one of the hottest topics in the AI research community [20]. We discuss the strengths and limitations of using LLMs for data preprocessing.

Strengths. (1) With their comprehensive understanding of language semantics and structures, and the knowledge acquired through training on vast amounts of text data, LLMs are general problem solvers capable of identifying errors, anomalies, and matches in textual data, without needing human-engineered rules [13] or fine-tuning for specific tasks. (2) Most LLMs provide a prompting interface with which users can interact and assign tasks in natural language, contrasting with existing data preprocessing solutions that require computer programming or specific tools (e.g., HoloClean [15] and Magellan [8]). (3) LLMs are excellent reasoners [7], enabling them to not only return data preprocessing results but also provide the reasons for these results. In this sense, their answers are more interpretable than those of other DL approaches. (4) LLMs can be conditioned by few-shot prompting [1]. As such, we can tune the criteria for data preprocessing tasks (e.g., the degree of matching) using few-shot examples.

Limitations. (1) For data preprocessing, one of the major limitations is the difficulty in domain specification [12]. When dealing with data from highly specialized domains, training LLMs can be costly and sometimes even impossible due to frozen parameters. (2) LLMs sometimes generate text that is plausible-sounding but factually incorrect or nonsensical, as they lack a fundamental understanding of the world and rely solely on the patterns they learned during training. (3) LLMs often require substantial computational resources, thereby increasing the cost of use and compromising the efficiency and scalability of data preprocessing on large-scale data.

3 METHOD

We design a prompt template as follows.

```
You are a database engineer.
[Zero-shot prompt]
[Few-shot prompt]
[Batch prompt]
```

Initially, we instruct the LLM to impersonate a database engineer. Other prompt components are marked within [] and will be discussed throughout this section.

3.1 Zero-shot Prompting

Zero-shot prompting is a technique that guides LLMs to generate the desired output. It has been demonstrated to effectively enhance the reasoning abilities of LLMs [7]. We employ zero-shot prompting to specify both the task and the answer format. Specifically, we adhere to the chain-of-thought paradigm [18], in which the LLM is expected to reason before delivering the answer. An example of a zero-shot prompt for DI is as follows:

```
You are requested to infer the value of the "city" attribute based on the values of other attributes.
MUST answer each question in two lines. In the first line, you give the reason for the inference. In the second line, you ONLY give the value of the "city" attribute.
```

We design specific zero-shot prompts for ED and DI. For ED, since we provide the entire record r but ask the LLM to detect an error in one attribute r_j at a time, the LLM might erroneously identify an error in attribute $r_{j'}$, where $j' \neq j$. To avoid this, we prompt the LLM to confirm the target attribute with: Please confirm the target attribute in your reason for inference. For DI, we provide a hint about the data type of the attribute to be imputed. For example, given the hint The "hoursperweek" attribute can be a range of integers, the LLM will respond with a range instead of a single number.

3.2 Few-shot Prompting

Few-shot prompting [1] involves providing a small selection of examples to condition the LLM for tasks that deviate from its pretraining objectives (e.g., text completion and code generation). We apply few-shot prompting by manually selecting a subset of data instances from the dataset and labeling them. For example, the few-shot examples for DI are presented as follows:

```
Users:
Question 1: Record is [Data Instance 1]. What is the city?
...
Assistant:
Answer 1: [Reason 1]
[Answer 1]
...
```

The data instances here adhere to the contextualization introduced in Section 3.3. Users are required to provide plausible reasoning for few-shot examples. For instance, given [name: "carey's corner", addr: "1215 powers ferry rd.", phone: "770-933-0909", type: "hamburgers", city: ???] as [Data Instance 1], [Reason 1] would be The phone number "770" suggests that the city should be either Atlanta or Marietta in Georgia. The addr attribute suggests a place in Marietta., and [Answer 1] would be Marietta.

3.3 Contextualization

Given that LLMs intake raw text as input, we convert the contents in each data instance to a text sequence in the following format:

[x_1.name: "x_1.value", . . ., x_n.name: "x_n.value"]

x_i denotes the i-th attribute of a data instance, `name` denotes to the attribute name, `value` denotes the cell value, and n is the number of input attributes. Specifically, we use ??? to denote missing values for DI, and x_1.name = name and x_2.name = description for SM.

3.4 Feature Selection

If metadata is available, users can manually select a subset of features to improve performance. For instance, when imputing a restaurant's location, the phone number and street name are relevant

Table 1: Comparison with baselines, measured in accuracy (%) for data imputation and F1 score (%) for the other tasks. LLMs are equipped with the best setting. "N/A" denotes not applicable or not reported in their original papers.

	Error Detection		Data Imputation		Schema Matching	Entity Matching						
Methods	Adult	Hospital	Buy	Restaurant	Synthea	Amazon-Google	Beer	DBLP-ACM	DBLP-Google	Fodors-Zagats	iTunes-Amazon	Walmart-Amazon
HoloClean	54.5	51.4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HoloDetect	99.1	94.4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
IPM	N/A	N/A	96.5	77.2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SMAT	N/A	N/A	N/A	N/A	38.5	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Magellan	N/A	N/A	N/A	N/A	N/A	49.1	78.8	98.4	92.3	**100**	91.2	71.9
Ditto	N/A	N/A	N/A	N/A	N/A	**75.6**	94.4	**99.0**	95.6	**100**	97.1	86.8
Unicorn	N/A	N/A	N/A	N/A	N/A	N/A	90.3	N/A	95.6	**100**	96.4	86.9
Unicorn ++	N/A	N/A	N/A	N/A	N/A	N/A	87.5	N/A	**96.2**	97.7	98.2	86.9
Table-GPT	N/A	N/A	N/A	N/A	N/A	70.1	96.3	93.8	92.4	97.7	92.9	82.4
GPT-3	**99.1**	**97.8**	98.5	88.4	45.2	63.5	**100**	96.6	83.8	**100**	98.2	87.0
GPT-3.5	92.0	90.7	98.5	94.2	57.1	66.5	96.3	94.9	76.1	**100**	96.4	86.2
GPT-4	92.0	90.7	**100**	**97.7**	**66.7**	74.2	**100**	97.4	91.9	**100**	**100**	**90.3**
GPT-4o	83.6	44.8	**100**	90.7	6.6	70.9	90.3	95.9	90.4	93.6	98.2	79.2

Table 2: Ablation study, measured in accuracy (%) for data imputation and F1 score (%) for the other tasks, using GPT-3.5. ZS-T denotes zero-shot task specification. FS denotes few-shots. B denotes batch prompting. ZS-R denotes zero-shot reasoning.

	Error Detection		Data Imputation		Schema Matching	Entity Matching						
Components	Adult	Hospital	Buy	Restaurant	Synthea	Amazon-Google	Beer	DBLP-ACM	DBLP-Google	Fodors-Zagats	iTunes-Amazon	Walmart-Amazon
ZS-T	25.9	18.4	86.2	81.4	18.2	54.7	83.3	94.7	58.5	92.7	80.0	81.5
ZS-T+B	37.8	19.1	83.1	81.4	17.4	60.1	78.3	94.9	59.6	92.7	83.9	81.6
ZS-T+B+ZS-R	46.3	26.2	89.2	65.1	5.9	45.8	50.0	72.6	47.6	92.7	82.0	60.7
ZS-T+FS	59.3	59.4	96.9	90.7	57.1	66.3	**96.3**	**97.0**	74.6	**100**	**96.4**	85.6
ZS-T+FS+B	58.1	56.1	96.9	86.2	53.3	**66.5**	**96.3**	96.2	**76.1**	97.8	94.7	**86.2**
ZS-T+FS+B+ZS-R	**92.0**	**90.7**	**98.5**	**94.2**	**61.5**	60.1	92.3	95.7	60.0	97.8	**96.4**	84.0

features, while the restaurant's name and type (Asian, Italian, etc.) are irrelevant. Therefore, users may choose to use only the phone number and street name as attributes in the above prompt.

3.5 Batch Prompting

Considering the significant token and time cost of LLMs, batch prompting [3] was proposed to enable the LLM to run inference in batches, rather than processing one sample at a time. To implement this technique, multiple data instances are presented in a single prompt, and the LLM is instructed to answer all of them. For example, for DI, the prompt is the same as the first part of few-shot prompting (i.e., the part before Assistant:). We propose two modes for batching: the first is random batching, where data instances are randomly assigned to a batch; and the second is cluster batching, where we perform clustering on the dataset, and then random batching is conducted within each cluster.

4 EXPERIMENTS

4.1 Experimental Setup

We use the datasets evaluated in [12]. We evaluate three LLMs: GPT-3.5-turbo-0301 (referred to as GPT-3.5), GPT-4-0314 (referred to as GPT-4), and GPT-4o-2024-05-13 (referred to as GPT-4o). The temperature parameter for these models is set at 0.35. For SM tasks, we use 3 few-shot examples, and for other tasks, we use 10. The default batch prompting method is random batching. The batch size ranges for GPT-3.5, GPT-4, and GPT-4o are [10, 20], [10, 15], and [5, 10], respectively. As baselines, we employ GPT-3 (text-davinci-002) with the best settings [12] for all four tasks, and HoloClean [15] and HoloDetect [6] for ED, IPM [11] for DI, SMAT [19] for SM, and Magellan [8], Ditto [10], Unicorn/Unicorn ++ [17], and Table-GPT [9] for EM. As these baselines have been evaluated in [12], we use these results as a reference. Open LLMs like LLaMA are not considered here, as they are generally less competitive than close models [4].

4.2 Experimental Results

Performance comparisons are presented in Table 4.2. GPT-4 surpasses GPT-3 on three out of the four tasks: DI, SM, and EM. For DI and SM, and achieves superior performance than previous methods, particularly for SM. Moreover, GPT-4 emerges as the victor on 4 out of 7 datasets for EM. GPT-3.5 also presents strong competition, outperforming GPT-3 on DI and SM. GPT-4o is generally on a par with GPT-3.5 on DI and EM, but turns out to be mediocre on ED and SM, showcasing inconsistent performance. Table-GPT, as GPT-3.5 fine-tuned for processing tabular input, roughly exhibits reduced performance from GPT-3.5 on EM. Consequently, we recommend users to either employ larger models or fine-tune its parameters for

Table 3: Batch size evaluation, measured on the Adult dataset for ED, using GPT-3.5 without few-shot prompting.

Batch size	F1 score (%)	Tokens (M)	Cost ($)	Time (hrs)
1	44.0	4.07	8.14	4.76
2	45.9	2.38	4.75	2.70
4	45.1	1.87	3.74	2.06
8	45.0	1.61	3.21	1.82
15	**46.3**	**1.49**	**2.99**	**1.60**

these tasks, and avoid the model with more HCI focus (i.e., GPT-4o) for the time being. We also observe Ditto, a non-GPT method, excelling on a few datasets. For ED, our performance is not as competitive as the GPT-3 results reported in [12]. The prompts used for GPT-3 in [12] are not directly applicable for GPT-3.5 and GPT-4. We believe the results of ED warrant further investigation, such as a case-by-case comparison.

To assess the effectiveness of our prompting strategy, we test GPT-3.5, as it is more cost-effective and faster than GPT-4, while delivering notable performance in the above evaluations. This makes it a more desirable choice for applications dealing with large datasets. The results are reported in Table 2. We start with GPT-3.5 prompted with only task specification (i.e., without reasoning, as shown in the first line of the prompt in Section 3.1) through zero-shot prompting. The result quality for ED and SM is very low, and roughly below 90% for DI and EM. The inclusion of few-shot examples improves all performances, exceeding 50% for ED and SM and reaching approximately 90% for the others. Batch prompting generally has a slight negative effect on result quality. With zero-shot reasoning, the performances of ED, DI, and SM are further improved, with ED over 90% and SM over 60%. However, there is little improvement observed for EM, potentially due to GPT-3.5's reasoning limitations and the lack of adequate input information for reasoning.

Feature selection proves useful for GPT-4. For instance, for entity matching on the Beer dataset without few-shot prompting, the F1 scores before and after feature selection are 74.1% and 90.3%, respectively. In terms of batch prompting, we compare random batching with cluster batching, where data instances are clustered using k-means over their Sentence-BERT [14] embeddings. For entity matching on the Amazon-Google dataset without few-shot prompting, F1 scores increase from 45.8% to 50.6% when switching from random to cluster batching, illustrating the effectiveness of cluster batching.

We explore the impact of batch size and present the results in Table 3. As batch size augments, there is a significant reduction in the number of tokens, dropping from over 4M without batch prompting to 1.5M with a batch size of 15. Both the cost and processing time follow similar trends, decreasing from $8.14 to $2.99 and from 4.8 hours to 1.6 hours, respectively. Concurrently, the F1 score experiences minor fluctuations, even displaying an increase when the batch size is set to 15. This is because GPT-3.5 can identify commonalities in questions and generate consistent solutions for all data instances in the batch, thereby enhancing overall performance.

ACKNOWLEDGMENTS

This work is supported by JSPS Kakenhi 23K17456, 23K25157, 23K28096, and CREST JPMJCR22M2.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[2] L. Cheng, X. Li, and L. Bing. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*, 2023.

[3] Z. Cheng, J. Kasai, and T. Yu. Batch prompting: Efficient inference with large language model APIs. *arXiv preprint arXiv:2301.08721*, 2023.

[4] W.-L. Chiang, L. Zheng, Y. Sheng, L. Dunlap, A. Angelopoulos, C. Chou, T. Li, and S. Zhuang. LMSYS chatbot arena leaderboard, 2024.

[5] csunny. DB-GPT: Revolutionizing database interactions with private LLM technology, 2023.

[6] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas. HoloDetect: Few-shot learning for error detection. In *SIGMOD*, pages 829–846, 2019.

[7] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[8] P. Konda, S. Das, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, et al. Magellan: toward building entity matching management systems over data science stacks. *PVLDB*, 9(13):1581–1584, 2016.

[9] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D. R. Fainman, D. Zhang, and S. Chaudhuri. Table-GPT: Table-tuned GPT for diverse table tasks. *arXiv preprint arXiv:2310.09263*, 2023.

[10] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *PVLDB*, 14(1):50–60, 2020.

[11] Y. Mei, S. Song, C. Fang, H. Yang, J. Fang, and J. Long. Capturing semantics for imputation with pre-trained language models. In *ICDE*, pages 61–72, 2021.

[12] A. Narayan, I. Chami, L. Orr, and C. Ré. Can foundation models wrangle your data? *PVLDB*, 16(4):738–746, 2022.

[13] S. Razniewski, A. Yates, N. Kassner, and G. Weikum. Language models as or for knowledge bases. *arXiv preprint arXiv:2110.04888*, 2021.

[14] N. Reimers. Sentence bert. https://www.sbert.net/, 2022.

[15] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(10):1190–1201, 2017.

[16] I. Trummer. Codexdb: Synthesizing code for query processing from natural language instructions using gpt-3 codex. *PVLDB*, 15(11):2921–2928, 2022.

[17] J. Tu, J. Fan, N. Tang, P. Wang, G. Li, X. Du, X. Jia, and S. Gao. Unicorn: A unified multi-tasking model for supporting matching tasks in data integration. *PACMMOD*, 1(1):1–26, 2023.

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[19] J. Zhang, B. Shin, J. D. Choi, and J. C. Ho. Smat: An attention-based deep learning solution to the automation of schema matching. In *ADBIS*, pages 260–274, 2021.

[20] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.