Генерация pandas-кода для ответов по большим таблицам с помощью LLM

А. А. Вяткин

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук

aav@dscs.pro

В. Д. Олисеенко

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук

vdo@dscs.pro

Аннотация. **Большие языковые модели на текущий момент применяются в различных областях и задачах. В частности, они способны анализировать разного рода структурированные/неструктурированные данные, в том числе таблицы. В данной статье исследуется вопрос ответов на большие таблицы с помощью генерации pandas-кода. Приводится описание метода генерации кода, при котором в промпте, подаваемом LLM, описывается лишь часть таблицы – названия столбцов и примеры строк. Экспериментально показано, что увеличение количества подаваемых строк может ухудшить результаты модели. Для проведения замеров использовался датасет DataBench, из которого были взяты вопросы с ответами в виде числа или категориального значения. В качестве моделей для сравнения взяты GigaChat Max и GPT-4o-mini. GigaChat Max достигает точности 78.6 %, GPT-4o-mini – 87 %. GPT-4o-mini по ответам в виде чисел (88.5 %) и категориальным значениям (86.6 %) на текущий момент достигает SOTA (state-of-the-art) значений на бенчмарке DataBench. Результаты исследования могут быть использованы для оптимизации систем аналитики данных на основе языковых моделей.**

Ключевые слова: большие языковые модели; анализ больших таблиц; генерация pandas-кода

I. Введение

В последнее время большие языковые модели (Large Language Models, LLM) находят широкое применение в различных областях, таких как финансы [1], здравоохранение [2], образование [3], юриспруденция [4], рынок труда [5]. Эти модели способны анализировать текстовые данные различного формата, включая возможность отвечать на вопросы на основе табличной информации [6].

Однако большинство существующих бенчмарков для оценки работы LLM с таблицами ограничивается небольшими таблицами, которые полностью помещаются в контекст модели [7–8]. При таких условиях модель способна сразу дать ответ по таблице. В реалиях же бизнес-задач часто встречаются таблицы значительно больших размеров, что затрудняет использование методов, которые оперируют с полным набором подаваемых модели табличных данных. Одним из возможных решений этой проблемы является генерация кода, который анализирует таблицы. В частности, код может генерироваться на Python с использованием библиотеки Pandas.

Статья выполнена в рамках научно-исследовательской работы по государственному заданию ФИЦ РАН № FFZF-2024-0003.

Из открытых наборов данных, найденных авторами статьи и позволяющих оценить ответы на вопросы по большим таблицам с помощью LLM, лишь датасет DataBench [9] предоставляет такую возможность. В данной работе [9] также используется метод, генерирующий код на Pandas. Однако в статье почти не проанализированы варианты форматов данных, подаваемых в контекст модели, а также контекст содержит лишь минимальную информацию о наборе данных. В частности, отсутствуют сведения о возможных уникальных значениях и количестве пропущенных данных, которые позволили бы LLM лучше проанализировать таблицу.

В этой статье будет расширен подход, использованный в [9], и, в результате, описан метод генерации кода на Pandas. Также будет рассмотрено, как количество строк в подаваемой части таблицы влияет на релевантность ответов. Теоретическая значимость данной работы заключается в описании способа обработки больших таблиц с помощью генерации pandas-кода. Практическая значимость работы заключается в результатах тестирования этого способа, определении оптимальных параметров.

II. Оценка анализа таблиц

Ответы на вопросы, касающиеся табличных данных, представляют собой важную задачу в области автоматического ответа на вопросы (QA, Question Answering). Основной целью данной задачи является извлечение информации из табличных данных путем обработки запросов на естественном языке [10]. LLM же являются одним из типов моделей, способных эффективно обрабатывать табличные данные и давать ответы на вопросы по ним.

Традиционно адаптация LLM для понимания табличных данных требовала внесения изменений в их архитектуру, таких как добавление специальных позиционных эмбеддингов и улучшение механизмов внимания [10]. Однако с развитием еще более крупных LLM, таких как GPT-4, Claude и Gemini [11], появился новый подход, позволяющий лишь изменением подаваемого запроса использовать LLM для генерации кода, который анализирует таблицы [12]. В то время как генерация SQL-кода весьма распространена в данном контексте, создание кода на более универсальном языке программирования Python (например, с использованием пакета Pandas) предоставляет дополнительные возможности для анализа табличных данных.

Для оценки способности LLM анализировать таблицы на данный момент существует большое число бенчмарков [7–8]. Однако большинство из них фокусируется на небольших, хорошо структурированных таблицах, часто основанных на данных из Википедии, и содержит незначительное количество пропусков [7]. Некоторые наборы данных используют большие таблицы, но непосредственно не рассматривают задачу ответа на вопросы по ним, сосредоточиваясь на сравнении, например, эталонных и сгенерированных SQL-запросов [13]. Авторами статьи найден лишь один открытый набор данных, использующий больше таблицы напрямую для задачи QA — DataBench [9].

DataBench представляет собой бенчмарк, состоящий из 1300 вопросов, составленных разметчиками, на основе 65 крупных таблиц. В среднем, каждая таблица содержит около 50 000 строк и около 25 столбцов. Ответы на вопросы делятся на 9 категорий, включая числовые, категориальные значения, даты, булевы значения, списки чисел и списки категориальных значений.

В рамках оценки моделей в [9] были сравнены два подхода к генерации ответов: формирование ответа самой моделью с подаваемой целиком таблицей и генерация кода на Pandas. Для обеспечения сопоставимости подходов при сравнении использовались укороченные версии таблиц, полностью помещающиеся в контекст модели. Эксперименты показали, что генерация кода на Pandas практически по всем типам ответов превосходит подачу таблицы в контексте целиком [9]. При этом для генерации кода в запросе к модели указывались следующие элементы:

- названия столбцов;
- неполный пример генерируемого кода на Pandas, включающий функцию для обработки таблицы, представляемую в виде `DataFrame`;
- тип выходных данных;
- примеры, типы значений Pandas, используемых в таблице;
- сам вопрос.

Единственной вариацией в запросе было наличие или отсутствие типов значений Pandas. Помимо этого, в статье [9] не рассматриваются такие аспекты, как количество строк, подаваемых модели, а также информация о пропущенных и уникальных значениях, которые могут помочь модели лучше понять таблицу. Текущее же исследование направлено на более глубокий анализ и описание контекста, подаваемого модели для генерации кода.

III. Эксперимент

В данной главе мы представляем эксперимент, целью которого является оценка способности LLM к генерации корректных функций на Python для обработки табличных данных с использованием библиотеки Pandas. В качестве используемых моделей будут выступать GigaChat Max [14] и GPT-4o-mini [11]. Набором данных послужит бенчмарк DataBench [9]. В эксперименте рассмотрим то, как модели отвечают на вопросы, которые требуют получения единственных и конкретных значений двух типов данных, таких как категориальные значения и числа. Эти типы данных представляются наиболее распространенными, и, так как в текущей работе идет сосредоточение на описании общего метода генерации кода, то для первоначальной проверки будем использовать только их. Всего в результате из 1300 вопросов DataBench было выбрано 523 вопроса. Эксперимент заключался в генерации кода для каждого из 523 вопросов. Далее более подробно опишем метод генерации кода.

Для каждой таблицы и вопроса моделям предлагались автоматически формируемые промпты (запросы к моделям), что позволяло моделям создавать функции на Python, которые должны были анализировать таблицу, представленную в виде `DataFrame`. Функции имели фиксированное имя — `get_value`. Пример формата сгенерированного кода указан на рис. 1.

```
import pandas as pd


def get_value(df: pd.DataFrame):
    """[Описание функции, созданное моделью]"""
    # Код, рассчитывающий значение
    return value
```

Рис. 1. Пример формата кода, сгенерированного LLM

В каждом запросе модели указывалась следующая информация:

- Часть таблицы, состоящая из заголовков таблицы, а также строк из таблицы в формате CSV. Всего рассматривалось 4 варианта количества строк: 2, 5, 10, 50. Половина из этих строк была первыми строками, половина – случайными.
- Типы данных столбцов – типы данных Pandas. Все типы данных доопределялись автоматически с помощью Pandas (функция `convert_dtypes`).
- Для каждого столбца указывался процент пропущенных значений.
- Для всех столбцов, тип которого в итоге после преобразований являлся строкой, также подавались все уникальные значения для столбца. Максимум подавалось 100 значений на столбец.
- Отдельно модели указывалось на выбор значений из правильного столбца, без указания конкретного столбца; на важность использования правильных, неизмененных названий столбцов.

В случаях, когда сгенерированный код вызывал ошибку при выполнении, модель получала сгенерированный ею текст и сообщение об ошибке. Если ошибка повторялась, предоставлялись все предыдущие сообщения об ошибках и сообщения, сгенерированные моделью. Максимальное число попыток перегенерации кода – 5.

Наконец, итоговой метрикой служила точность – доля правильных ответов, полученных после вызова функции `get_value` с поданной в качестве аргумента таблицей. Ответ модели засчитывался, если он полностью совпадал с эталонным ответом.

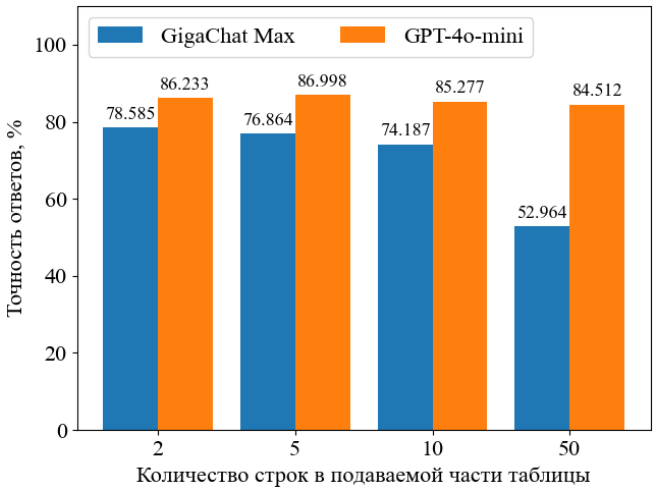

Рис. 2. Точность ответов проверяемых моделей

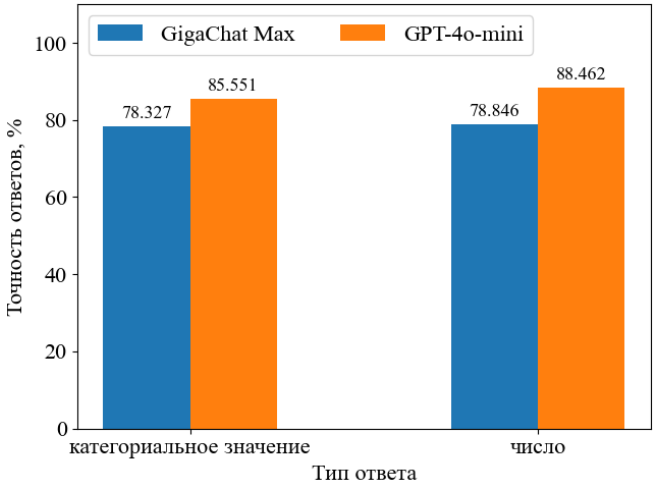

Рис. 3. Точность ответов моделей по типу возвращаемого ответа. Показаны лучшие результаты: для GigaChat Max показана точность при 2 подаваемых строках, для GPT-4o-mini – при 5

Например, не засчитывался список, который содержит единственное значение, являющееся правильным ответом. Полученные результаты указаны на рис. 2–3.

IV. Анализ результатов

Проанализируем результаты эксперимента, описанного в предыдущей главе. На рис. 2 представлена точность ответов моделей в зависимости от количества подаваемых строк таблицы. Модель GigaChat Max продемонстрировала лучший результат с двумя подаваемыми строками, составивший 78.6 %. Модель GPT-4o-mini получила еще более высокие показатели, достигая максимума в 87% при 5 подаваемых строках. Однако следует отметить, что дальнейшее увеличение количества строк, подаваемых в модель, негативно сказалось на качестве ответов. У GigaChat Max снижение значений метрик составило 25.6 пункта при 50 подаваемых строках, в то время как у GPT-4o-mini это снижение составило 2.5 пункта. Эти данные указывают на то, что дальнейшее увеличение количества строк не предоставляет модели полезной информации, причем GPT-4o-mini проявила большую устойчивость к такой лишней информации по сравнению с GigaChat Max.

На рис. 3 видно, что качество ответов обеих моделей в значительной степени не зависит от типов ответа. Тем не менее, для вопросов, требующих числовых ответов, обе модели показали лучшие результаты: GigaChat Max достиг 78.8 % (против 78.3 % для категориальных ответов), а GPT-4o-mini — 88.5 % (против 85.5 %). Также стоит отметить, что обе модели превысили максимальные показатели, зафиксированные в работе [9] для модели ChatGPT-3.5, которые составили 75.9 % для числовых ответов и 73.3 % для категориальных. На текущий момент достигнутые значения представляют собой SOTA (state-of-the-art) показатели. Тем не менее, необходимо провести более детальное сравнительное исследование между подходами, описанными в статье [9] и в данной работе, так как достижение таких метрик может быть обусловлено в значительной степени увеличением вычислительной мощности моделей.

V. Заключение

В данной статье рассматривается использование больших языковых моделей для генерации кода на языке pandas в ответах на вопросы, касающихся больших таблиц. Описывается метод генерации кода, в рамках которого в промпт модели включаются в том числе ограниченные данные о таблице, такие как названия столбцов и примеры строк. Анализировались разные варианты количества подаваемых строк. Эксперимент с использованием части датасета DataBench [9] для оценки производительности моделей GigaChat Max и GPT-4o-mini показал, что последняя модель демонстрирует лучшие результаты, устанавливая SOTA значения для различных типов ответов: 88.5 % для числовых и 86.6% для категориальных. В целом, по выбранному набору данных модель GPT-4o-mini достигла 87% при пяти подаваемых строках, в то время как GigaChat Max показал 78.6 % при двух строках. Дальнейшее увеличение числа строк привело к ухудшению значений метрик.

Исследование возможностей анализа данных с использованием LLM открывает новые перспективы для оптимизации систем аналитики, что может существенно повысить эффективность работы с данными. В продолжении текущего исследования, дальнейшие работы могут быть направлены на более тщательное сравнение подходов, описанных в данной работе и в [9], а также на изучение различных методов представления информации, подаваемой модели, включая, например, влияние формата таблицы.

Список литературы

[1] Wu S., Irsoy O., Lu S., Dabravolski V., Dredze M., Gehrmann S., Kambadur P., Rosenberg D., Mann, G. Bloomberggpt: A large language model for finance // arXiv preprint arXiv:2303.17564. 2023.

[2] Thirunavukarasu A.J., Ting D.S.J., Elangovan K., Gutierrez L., Tan T.F., Ting D.S.W. Large language models in medicine // Nature medicine. 2023. Vol. 29, № 8. P.1930–1940.

[3] Kasneci E., Seßler K., Küchemann S., Bannert M., Dementieva D., Fischer F., Gasser U., Groh G., Günnemann S., Hüllermeier E., Krusche S. ChatGPT for good? On opportunities and challenges of large language models for education // Learning and individual differences. 2023. Vol. 103. p.102274.

[4] Janatian S., Westermann H., Tan J., Savelka J., Benyekhlef K. From text to structure: Using large language models to support the development of legal expert systems // Legal Knowledge and Information Systems. IOS Press. 2023. P. 167–176.

[5] Eloundou T., Manning S., Mishkin P., Rock D. GPTs are GPTs: Labor market impact potential of LLMs // Science. 2024. Vol. 384, № 6702. P.1306–1308.

[6] Sui Y., Zhou M., Zhou M., Han S., Zhang D. Table meets llm: Can large language models understand structured table data? a

[7] benchmark and empirical study // Proceedings of the 17th ACM International Conference on Web Search and Data Mining. 2024. P. 645–654.

[8] Wu X., Yang J., Chai L., Zhang G., Liu J., Du X., Liang D., Shu, D., Cheng X., Sun T., Li T. TableBench: A comprehensive and complex benchmark for table question answering // Proceedings of the AAAI Conference on Artificial Intelligence. 2025. Vol. 39, №. 24. P. 25497–25506.

[9] Pasupat P., Liang P. Compositional Semantic Parsing on Semi-Structured Tables // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015. P.1470–1480.

[10] Grijalba J.O., Lopez L.A.U., Martínez-Cámara E., Camacho-Collados J. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs // Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). 2024. P. 13471–13488.

[11] Jin N., Siebert J., Li D., Chen Q. A survey on table question answering: recent advances // China Conference on Knowledge Graph and Semantic Computing. 2022. P. 174–186.

[12] Zhao F.F., He H.J., Liang J.J., Cen J., Wang Y., Lin H., Chen F., Li T.P., Yang J.F., Chen L., Cen L.P. Benchmarking the performance of large language models in uveitis: a comparative analysis of ChatGPT-3.5, ChatGPT-4.0, Google Gemini, and Anthropic Claude3 // Eye. 2022. P.1–6.

[13] Shi L., Tang Z., Zhang N., Zhang X., Yang Z. A survey on employing large language models for text-to-sql tasks // arXiv preprint arXiv:2407.15186. 2024.

[14] Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Ma J., Li I., Yao Q., Roman S., Zhang Z. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018. P. 3911–3921.

[15] Pugachev A.A., Kharchenko A.V., Sleptsov N.A. Transforming the future: a review of artificial intelligence models // RUDN Journal of Studies in Literature and Journalism. Vol 28, No 2 (2023) Pages: 355-367.