Paper SM14

Using large language models for consistency checking tables, listings, and graphs

Yingying Wei, Genentech, South San Francisco, USA
Will Harris, Genentech, South San Francisco, USA

ABSTRACT

Programmatic data quality checking is essential for accurately summarizing clinical trial results. Automated checks are common at various levels of clinical trial data, including on the raw datasets from the case report forms (CRFs), SDTMs, and ADaMs. Unfortunately automated checking is often difficult for tables, listings, and graphs (TLGs) due to their unstructured nature. Large language models (LLMs) however excel at parsing unstructured text and can potentially automate TLG quality checks.

Consistency checking ensures information is cross-referenced within or across TLGs, for example verifying patient numbers in treatment arms across separate tables. Currently, this type of checking is often performed in an ad hoc nature and is generally manual and thus tedious and time-consuming. An automated framework for TLG cross-checking could reduce resource use and help improve quality of clinical trial summaries. We present a proof of concept framework for using LLMs to automate TLG consistency checks.

INTRODUCTION

Data quality checking is typically performed across multiple functions on clinical trials. Data managers typically have robust edit checks in place on raw CRFs, clinicians routinely perform data quality reviews, and data scientists rely on tools like Pinnacle 21 [1] to flag issues at the SDTM and ADAM level. One common example of an automated edit check at the CRF level is flagging records where an adverse event is reported without a severity grade. Tabular datasets with strong standards and well defined data models enable robust data quality checking.

Similar quality checking is commonly performed on TLGs of clinical trial summaries. This is often performed by multiple functions, for example programmers, statisticians, and clinicians typically all review TLGs to ensure accuracy and consistency. Some of these checks might include:

- Are the number of patients per treatment group reported consistently across TLGs?
- Are the number of patients reporting adverse events reported consistently across adverse TLGs?
- Do the number of deaths on a death summary table match the number of deaths on an adverse events summary table?

Unfortunately the unstructured nature of TLGs does not lend itself to automated consistency checking. This means that TLGs are typically tediously cross checked manually, often by multiple reviewers.

Large language models excel at parsing unstructured text data and therefore have the potential to assist in automatically performing quality checks directly on TLGs. Data cleaning and quality control are resource intensive aspects of clinical trials, so any efforts to streamline those processes have the potential of speeding up drug development.

The concept of unit testing is used often in software development where a series of small checks are frequently run on software to help ensure consistent functionality. A tool that can reliably perform a large number of consistency checking on TLGs and concisely return flagged issues in a user friendly summary has the potential to increase

accuracy of clinical trial summaries while minimizing tedious and inefficient review. This paper investigates if large language models can assist in the task of consistency checking TLGs and whether a simple and reliable framework for doing so is feasible.

METHODS

Data Considerations

Three sets of TLGs were used for evaluating LLMs. One set was created using the chevron [2] R package and based on fake ADaM data generated by the scda [3] R package. This fake data had three arms. Two additional sets of TLGs from actual Roche trials generated using Roche's internal SAS macro library were also used for testing. One study had five arms whereas the other had two arms. TLGs included standard summaries of baseline demographics, overview of adverse events, adverse events by system organ class and preferred term, adverse events by grade, and deaths. For the fake study these tables specifically used the chevron package's DMT01, AET01, AET02, AET04, and DTHT01 templates. Internal summaries using Roche's SAS macro library were similar. Handling of all study data was done in compliance with Roche's legal standards.

Evaluating LLMs

An internal instance of OpenAI's ChatGPT 4o model was available for Roche use. In order to additionally investigate open source solutions the lightweight Meta Llama 3 8B Instruct model was deployed on our internal system. Both of these models had an application programming interface (API) and could be interacted with using R's httr [4] and jsonlite [5] packages.

TLGs were read in as text files, converted to a single string, and passed directly with questions to the model. Two different sets of questions were established, one easy and one harder. The easy set of questions focused on whether the LLM could return single data points from a TLG. Specifically the questions were as follows:

- What is the number of patients in a given arm on a given table?
- How many patients have at least one adverse event in a given arm on a given table?
- How many deaths occurred on the trial in a given arm on a given table?

The harder set of questions were similar but focused on whether specific values could be returned across all arms in an easily parsable format. Specifically these questions were:

- Can the total number of patients across all arms be returned in machine readable format for a given table?
- Can the number of patients across all arms with at least one adverse event be returned in machine readable format for a given table?
- Can the number of patient deaths across all arms be returned in machine readable format for a given table?

Measuring Accuracy

Determining the ability of LLMs to successfully perform cross checking of TLGs involves two potential points of failure. First the desired information from TLGs must be correctly extracted. Next extracted datapoints must be correctly compared. To eliminate one of these points of failure we focused only on whether LLMs could correctly extract information from TLGs and not if they could subsequently make the correct comparison. The rationale was that information extracted by LLM could be passed to traditional code for a more reliable comparison.

Therefore to measure accuracy we looked at how frequently the LLM could correctly return the expected value from a TLG. To account for randomness in LLM responses the same question was asked several times for a given TLG. Where applicable the same question was asked to multiple TLGs.

Evaluating OpenAI's Custom GPTs

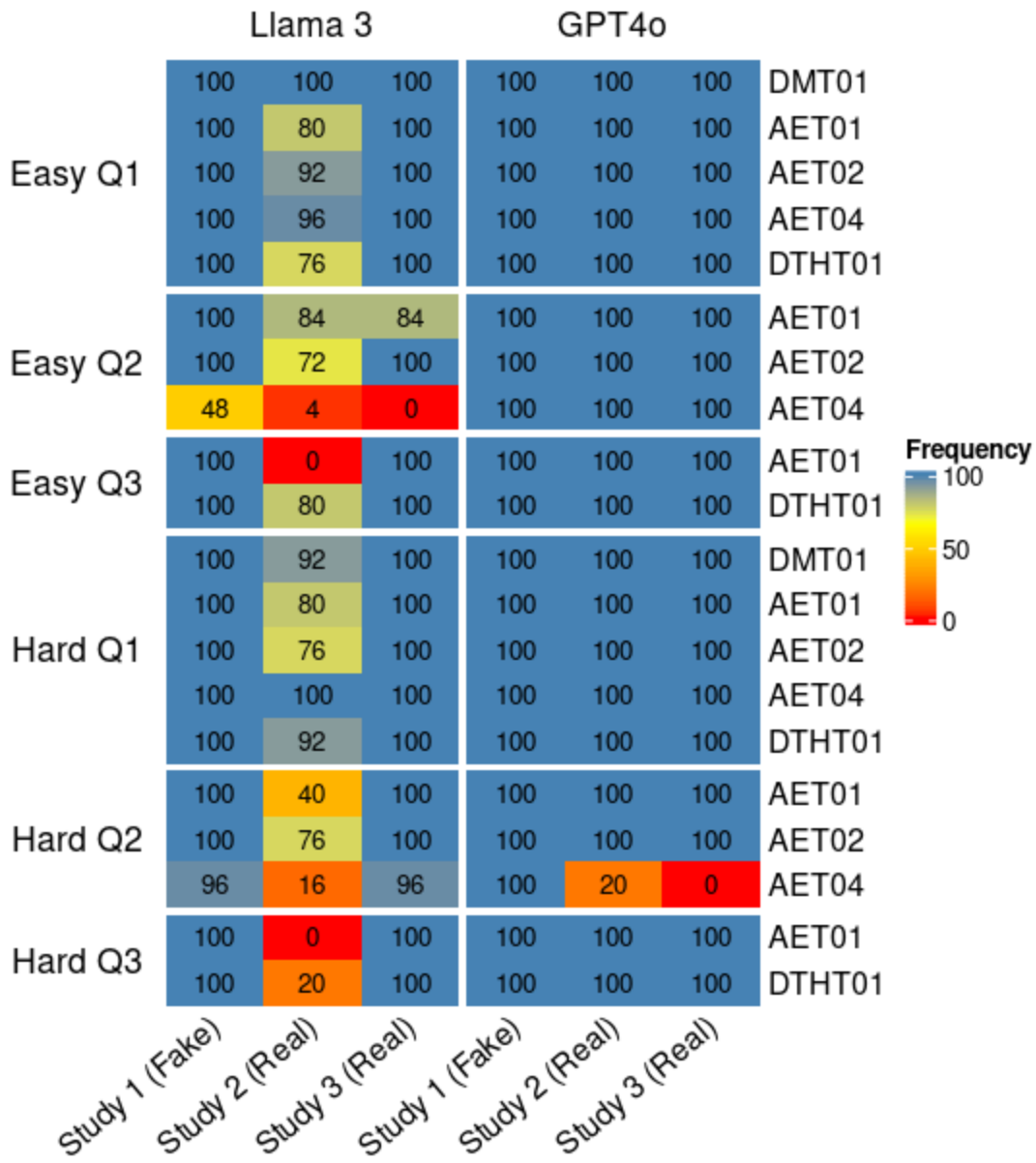
Additionally we further investigated OpenAI's functionality for creating custom GPTs trained on a specific set of source documents using ChatGPT 4o. For this experiment, a set of TLGs based on fake data was uploaded and a custom GPT trained on those outputs was created. Similar questions as previously outlined were asked. Unfortunately the ability to interact programmatically with this custom GPT was not available at the time. Because of this we were not able to obtain accuracy metrics based on asking a given question multiple times and instead report selected responses directly.

RESULTS

LLMs with APIs

Initial experiments focused on implementing retrieval augmented generation (RAG), i.e. dividing given tables into small chunks and passing the most relevant chunk to the LLM along with the prompt. This did not work well in our limited initial experiments. However we quickly found that passing an entire table as a single string with a given question to the LLM worked fairly well.

The percent of correct answers based on asking each question 25 times across each trial are presented in Figure 1. GPT4o clearly outperformed Llama 3, which is expected due Llama 3 being a much smaller model. Llama 3 had at least one error on all but one of the questions for Study 2. Notably Study 2 was the trial with the most arms of the trials we evaluated. Surprisingly however Llama 3 performed fairly well on the harder questions for the other two trials. Extracting the number of patients with at least one adverse event in machine readable format (Hard Question 2) from the adverse events by grade table seemed to be a problem area for both models. Llama 3 ran faster than GPT4o, likely due to the model sitting on our internal systems. GPT4o typically ran 25 iterations of a single question in around 2 minutes.



OpenAI's Custom GPT

OpenAI's custom GPT trained on TLGs from our fake study appeared to correctly extract information across multiple TLGs and return results in machine readable format with limited prompting. Figure 2 has an example where patient frequencies per arm are correctly returned in machine readable format with limited prompting. Figure 3 has a similar example where patient frequencies for each system organ class are correctly returned per arm in machine readable format.

list out the patient frequencies by arm for each table. Make this in JSON format

json

Copy code


```
{
    "LBT01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "AET01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "AET02": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "AET04": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "DMT01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15,
        "ALL PATIENTS": 45
    },
    "DST01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "DTH01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    },
    "EXT01": {
        "A: DRUG X": 15,
        "B: PLACEBO": 15,
        "C: COMBINATION": 15
    }
```

For each arm list of the number of unique patients with an adverse event in each system organ class but not preferred term in the aet02 table. put this in json format, with no additional information.

json Copy code


```
{
    "A: DRUG X": {
        "CL B.2": 11,
        "CL D.1": 9,
        "CL A.1": 7,
        "CL B.1": 5,
        "CL C.2": 6,
        "CL D.2": 2,
        "CL C.1": 4
    },
    "B: PLACEBO": {
        "CL B.2": 8,
        "CL D.1": 5,
        "CL A.1": 6,
        "CL B.1": 6,
        "CL C.2": 4,
        "CL D.2": 5,
        "CL C.1": 4
    },
    "C: COMBINATION": {
        "CL B.2": 10,
        "CL D.1": 11,
        "CL A.1": 10,
        "CL B.1": 8,
        "CL C.2": 8,
        "CL D.2": 7,
        "CL C.1": 5
    }
}
```


CONCLUSION

These results suggest that LLMs extract information from TLGs with good reliability. Llama 3's accuracy was reasonable for a small model, though it had major problems with almost all questions for Study 2. That trial had more arms than the other two (five) raising the question of whether that caused a problem for the model. GPT-4o performed well across all three trials, only stumbling on one question. OpenAI's custom GPT also seemed to perform very well in limited exploration. All models investigated were able to provide results in machine readable format

which is critical for making downstream programmatic comparisons. There were no major concerns with the time it took to retrieve answers to questions.

One major limitation to the approach of converting entire tables to a single string and passing it to the LLM along with the question is prompt size constraints. In large clinical trials TLGs can often be very long, for example summaries of adverse events by grade or summaries of labs by visit can span dozens of pages. There is a limit to the size of the prompts LLMs can receive so in order to continue using this approach some form of pre-processing would have to be implemented for lengthy TLGs. However OpenAI's custom GPTs get around this limitation since all source documents are uploaded for the custom GPT to be trained on. Unfortunately at the time of writing an API was not available for these custom GPTs.

It should be noted that a framework for consistency checking TLGs could also be established without LLMs using analysis results datasets, for example those generated by the cards [6] R package. One advantage of using LLMs for consistency checking is that they are not tied to a particular software used for generating TLGs.

Given programming API access in the future, one promising future state might consist of creating a custom GPT based on all TLGs from a given analysis. Under this approach, not only could unit tests for consistency be executed but the custom GPT could also be made available to team members for general results queries as well. Overall, these results suggest that a reliable framework built around LLMs for consistency checking of TLGs can be developed. Such a framework has the potential to reduce tedious manual review, improve accuracy of clinical trial summaries, and speed up drug development.

REFERENCES

1. https://www.pinnacle21.com/
2. https://cran.r-project.org/web/packages/chevron/index.html
3. https://github.com/insightsengineering/scda
4. https://cran.r-project.org/web/packages/httr/index.html
5. https://cran.r-project.org/web/packages/jsonlite/index.html
6. https://cran.r-project.org/web/packages/cards/index.html

ACKNOWLEDGMENTS

The authors would like to acknowledge Youshi Zhang for his contributions to this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Yingying Wei
Company: Genentech
Email: weiy38@gene.com