Paper ML10

Natural Language to Code: AI-Powered Automation for CDISC-Compliant Datasets

Vina Ro, Eli Lilly and Company, Indianapolis, United States

ABSTRACT

This paper presents an innovative approach to automating the creation of Clinical Data Interchange Standards Consortium (CDISC)-compliant datasets, which traditionally demand extensive manual effort and advanced programming skills. By leveraging advanced generative AI technologies, this methodology transforms natural language coding specifications for SDTM and ADaM datasets into R code. The process employs a two-step workflow: First, a Retrieval-Augmented Generation (RAG) framework enhances the precision and logic of the coding specifications using a Large Language Model (LLM) and prompt engineering. Second, the LLM generates highly accurate R code from the enhanced specifications. Testing on real-world studies demonstrated 88% accuracy in variable generation, with most errors stemming from data or human factors, not the AI system. This approach has reduced dataset creation time by an estimated 60%, showcasing its potential to improve efficiency, consistency, and scalability across clinical trials.

INTRODUCTION

The creation of Clinical Data Interchange Standards Consortium (CDISC)-compliant datasets for clinical trials is a complex, labor-intensive process that demands specialized expertise. Standardized under the Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM), these datasets are crucial for regulatory submissions but require significant programming knowledge and adherence to clinical trial standards. The manual nature of this process often results in inefficiencies, redundancy, and inconsistencies across studies, with delays and bottlenecks arising from the need for advanced programming skills that are not always readily available. To address these challenges, we have developed an AI-powered solution that automates the transformation of natural language coding specifications into CDISC-compliant datasets. By combining a RAG-inspired method with LLMs, our system enhances accuracy, scalability, and ease of implementation. This paper outlines the methodology, evaluates the system's performance, and explores its potential to streamline the dataset creation process, ultimately improving the efficiency and reliability of clinical trials.

METHOD

3.1 OVERVIEW

In this study, we introduce a novel application for automating the generation of SDTM datasets through a structured, multi-layered framework. As depicted in *Figure 1*, the system architecture consists of four layers: the Input Layer, AI Processing Layer, Output Layer, and Execution Layer, each serving a critical role in streamlining the dataset generation process while incorporating necessary human oversight.

In the Input Layer, users provide study-specific specifications that guide subsequent processing. In the AI Processing **Layer**, these specifications are enhanced using a LLM to generate structured coding prompts. At this stage, human review is required to ensure accuracy and contextual appropriateness before prompts are finalized. Once approved, the refined coding prompts are used to generate an initial R script for SDTM dataset creation. The Output Layer delivers this AI-generated R script as a first draft, requiring a second round of human validation to verify correctness and compliance to study requirements. Finally, in the Execution Layer, the validated R script is executed in an on-premises environment, producing the SDTM dataset.

This paper focuses on illustrating the development of the AI Processing Layer, specifically the methodologies and data-driven strategies employed to enhance the accuracy and reliability of the Output Layer.

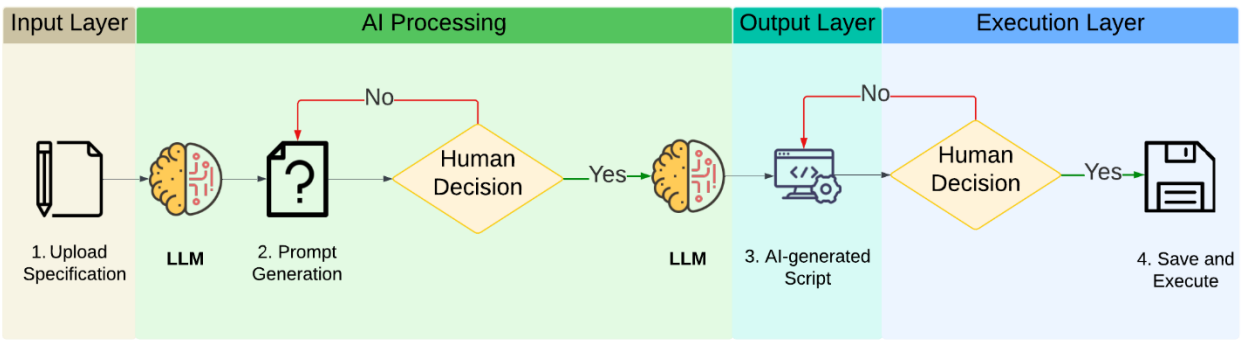

Figure 1. Application overview

3.2 AI PROCESSING WORKFLOW

Leveraging LLMs, the application converts free-text coding specifications into structured AI-generated coding prompts (AI prompts) that break down the original algorithms into clear, step-by-step instructions. A database was created to ensure high accuracy in enhancing the accuracy in converting free-text coding specifications to AI Prompts. These AI prompts are designed to enhance specificity, ensuring that each transformation is explicitly defined and aligned with dataset creation requirements. By structuring the coding logic in a more systematic manner, the AI prompts improve clarity, facilitate accurate interpretation, and enhance the precision of the code generation process.

3.2.1 DATABASE CREATION

A comprehensive database was established to systematically catalog SDTM dataset names, variable names, existing free-text specifications, and their AI prompt counterparts. To make sure everything was covered, detailed examples were added to show how to make all kinds of SDTM variables across a wide range of therapeutic areas and SDTM datasets. The database was subjected to rigorous validation and testing to confirm its adherence to organizational standards and its ability to consistently generate accurate R code on the first attempt when processed by an LLM. An example entry of a free-text specification and its corresponding AI prompt counterpart is shown in *Table 1*:

Dataset	Variable	Specification	AI Prompt
DM	DTHDTC	Pull SUBJID, DTHDAT from df_disposition. Convert to ISO8601 format. Merge onto demographics data by SUBJID. Can be null (if no death for a subject).	1. Pull SUBJID, DTHDAT from df_disposition where DTHDAT is not null. Convert DTHDAT into a date object. Name result as df_disposition_subset. 2. Do a left join with df_demographics on the left and df_disposition_subset. 3. Create DTHDTC column in df_demographics and assign values as: DTHDAT formatted to ISO8601 with format_ISO8601().

Table 1. Example of a free-text specification and its corresponding AI prompt

3.2.2 SPECIFICATION-ALIGNED MATCHING

Specification-Aligned Matching (SAM) is a structured retrieval methodology developed within the Data and Analytics Group at Eli Lilly and Company to improve the accuracy of coding specification alignment. Drawing inspiration from RAG frameworks, SAM differs fundamentally in its approach. While RAG incorporates retrieved document content into generative modeling, SAM is explicitly designed to retrieve and align pre-existing specifications within a structured database. This ensures consistency and reproducibility in variable transformations.

The process, illustrated in *Figure 2*, begins with a meta-filtering mechanism that systematically isolates relevant records based on dataset and variable names. Once filtered, a vector-based similarity search is performed using cosine similarity, a measure that quantifies the closeness between the query specification and stored entries within the database. Let $A \in R^n$ represent the vector encoding of the input specification and $B \in R^n$ denote the vector representation of an entry in the free-text specification column of the database. The cosine similarity between A and B is computed as:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

where

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

is the dot product of the two vectors,

$$\|A\| = \sqrt{\sum_{i=1}^{n} A_i^2}, \qquad \|B\| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

are the Euclidean norms of the respective vectors, and θ is the angle between the two vectors.

The retrieval process ranks database entries according to their cosine similarity scores, ensuring the selection of the most relevant specification. This selected specification, along with its corresponding AI prompt, is then used to generate a new AI prompt for the input specification, which is provided to the LLM for further processing.

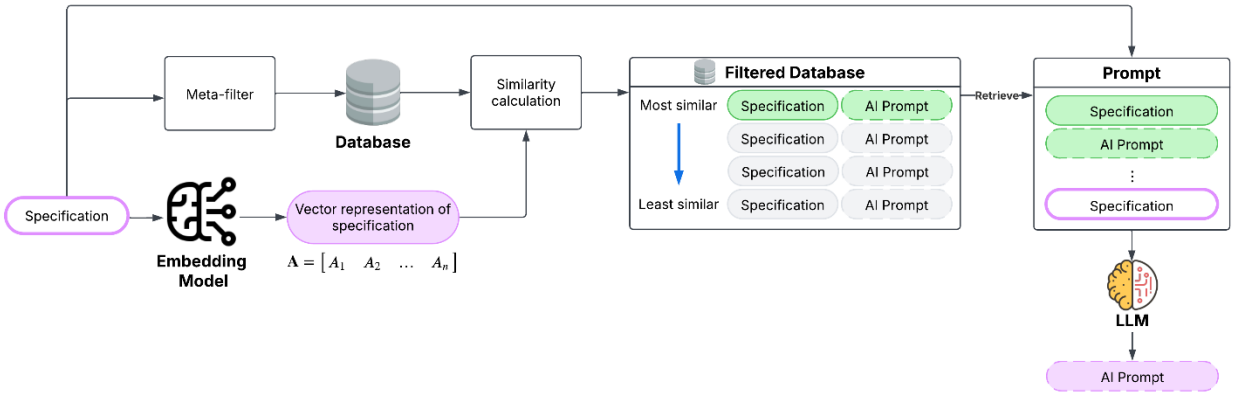

Figure 2. Specification-Aligned Matching process

3.2.3 PROMPT ENGINEERING

Once the most relevant specification and AI prompt pair has been retrieved from the database, the next phase involves constructing a structured prompt, shown in *Figure 3*, to guide the LLM in generating the corresponding AI prompt for the input specification.

The prompt is designed to serve multiple functions by utilizing multiple prompt engineering techniques. It begins with context setting, where the initial lines explicitly define the LLM's role—refining R-based transformation algorithms for SDTM dataset construction. To maintain a focus on high-level algorithmic refinement, the prompt explicitly instructs the model to exclude direct R code output. Following this, rule setting establishes a set of predefined guidelines, developed through extensive empirical testing, that the LLM must adhere to when revising the transformation logic.

The core of the prompt incorporates one-shot prompting, inserting a retrieved example derived from the SAM process. This example serves as a reference for the LLM, guiding it in a clear direction in structuring the output. Finally, the user query—the specific SDTM variable targeted for generation—is appended to the prompt, ensuring that the output remains directly relevant to the user's intended specification.

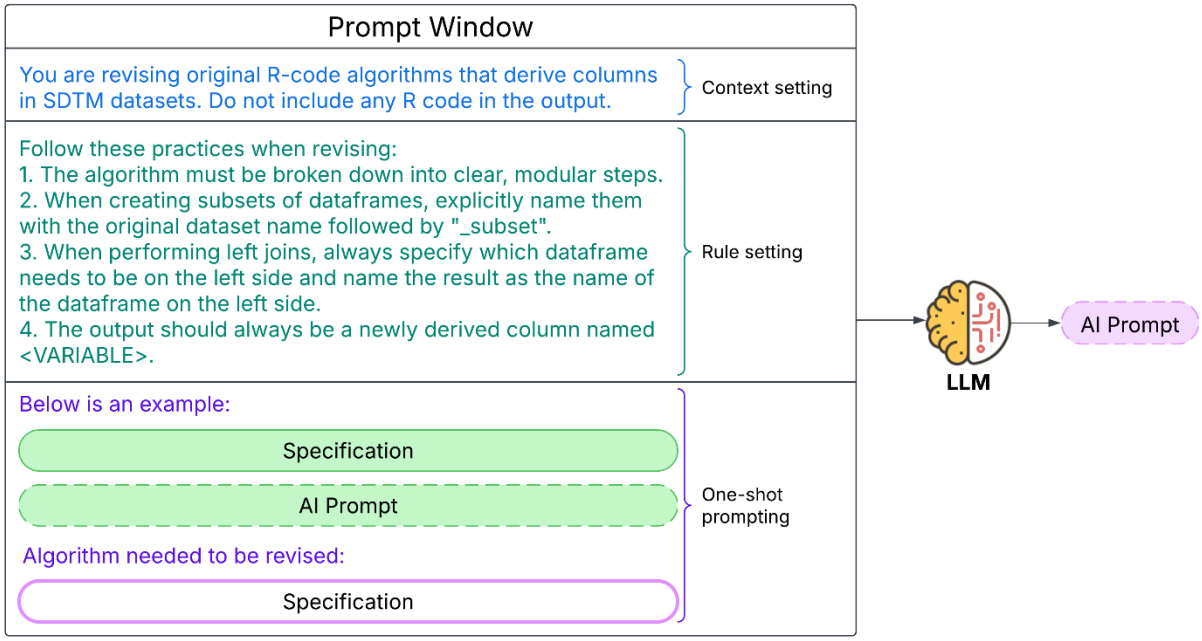

Figure 3. Example prompt for attaining the AI prompt from a given specification input

3.2.4 CODE GENERATION PROCESS

The AI prompt produced in the preceding step is subsequently fed into the large language model (LLM), which facilitates the transformation of the structured AI prompt into the corresponding R code. The process for generating R code is straightforward (*Figure 4*): the LLM is simply instructed to convert the structured AI prompt into an appropriate R code snippet. This procedure is repeated systematically for each variable in the dataset, ensuring that a unique and tailored code snippet is generated for every specified variable. Once all the variables have been processed, the individual code snippets are aggregated and compiled into a unified R script. The final output is a near-complete draft of the R script, which requires only minimal adjustments by statistical programmers to ensure accuracy and compliance. This automated approach significantly reduces the manual effort traditionally required for script development, streamlining the workflow and enhancing overall efficiency. As a result, the generation of SDTM-compliant datasets is expedited, allowing for faster and more consistent data processing in clinical trials.

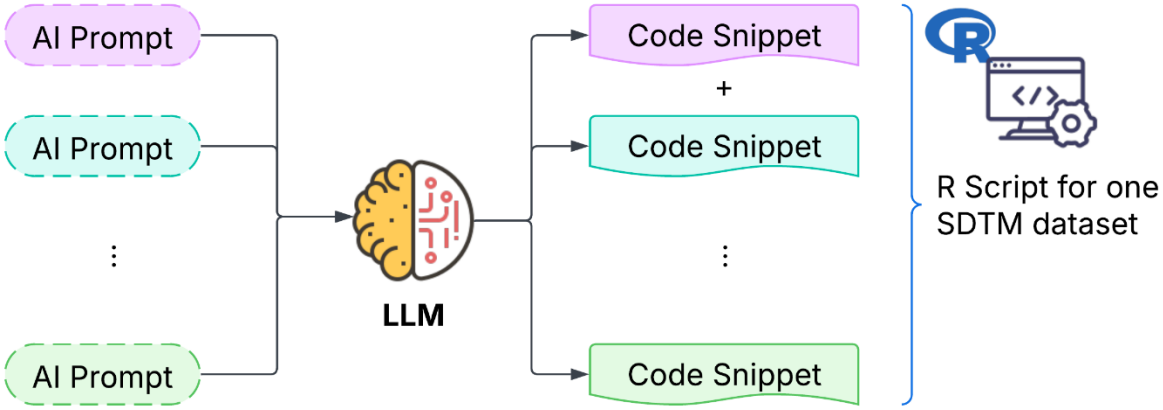

Figure 4. Process of code generation layer using LLMs

RESULT

The system was tested on an ongoing study and demonstrated a high degree of accuracy (*Figure 5*), successfully generating 88% of the variables of SDTM datasets without error, underscoring its robustness in translating specifications into executable code. Further analysis of the remaining 12% of cases revealed that 93% of errors were attributable to non-AI factors, such as data inconsistencies, incorrect specifications, or other human-related issues. In contrast, only 7% of errors were attributed to limitations in the AI-generated code itself , such as the use of incorrect parameter names within specific R functions. This suggests that while the system is highly reliable, its performance is still dependent on the quality and integrity of the input data.

In terms of efficiency, the system significantly accelerated the dataset creation process, reducing the time required for variable generation from several hours to an average of 20 minutes. This improvement was particularly evident in complex datasets, such as the SDTM LB dataset, where manual coding is traditionally time-consuming.

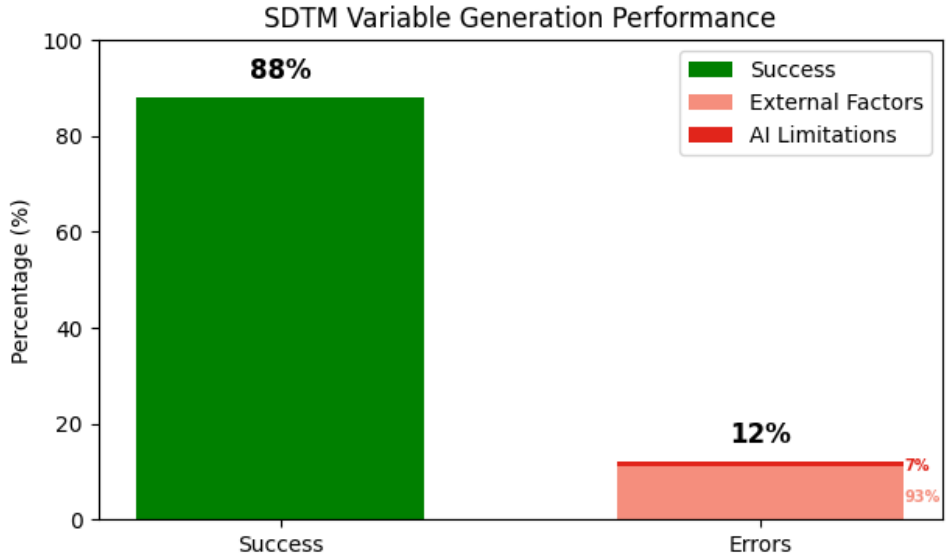

Figure 5. Results of SDTM generation

CONCLUSION AND FUTURE STEPS

This paper presents an innovative AI-powered solution for automating the creation of CDISC-compliant datasets, leveraging the developed SAM method and LLMs to achieve notable improvements in accuracy, efficiency, and scalability. The system demonstrated an 88% success rate in variable generation during testing, with the majority of errors arising from external factors rather than inherent flaws in the AI mechanism itself. This approach has the potential to significantly transform the clinical trial industry by reducing dependency on specialized programming expertise, improving consistency, and streamlining the dataset creation process. Future efforts aim to extend this system to the generation of ADaM datasets and Tables, Figures, and Listings (TFLs). Nevertheless, the system is subject to certain limitations. Its performance is inherently dependent on the quality and completeness of input data, wherein errors, ambiguities, or inconsistencies in specifications may adversely affect accuracy. While the AI-generated code demonstrates a high degree of reliability, complex transformations and edge cases may still necessitate manual review and refinement to ensure correctness. Furthermore, the system's reliance on a predefined retrieval process may limit its ability to discern subtle nuances in specifications, potentially resulting in suboptimal matches. To enhance adaptability and expand its applicability across diverse clinical trial datasets, ongoing refinements, maintenance, and iterative improvements to the specification database are essential.

ACKNOWLEDGMENTS

The author extends sincere gratitude to Sathish Sundaram, Ravi Gupta, Soma Bhadra, and Ian Sturdy for their valuable contributions to this project. Their insights, expertise, and support were instrumental in the development and refinement of this work.

CONTACT INFORMATION

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

Vina Ro
Eli Lilly and Company
Lilly Corporate Center, Indianapolis IN 46285 U.S.A.
vina.ro@lilly.com