PharmaSUG 2024 - Paper SI-189

Automating annotation of TLF mocks Using Generative AI

Vidya Gopal, AstraZeneca

ABSTRACT

In this paper we will be discussing the use of Generative AI to automate the mapping and annotation of ADaM variables to TLF mocks to increase the efficiency, quality, and accuracy of TLFs. The TLF mock provides a specification for how the TLF should be created from the ADaM dataset. In a study, annotating the mock is crucial to provide quality and traceability. Annotating a mock is time and labor intensive due to the volume of TLFs and ADaM variables. This proof-of-concept proposal would entail exploring the use of Generative AI prompts to match the variable name given in the ADaM specification document to the general name given in the TLF mock shell. For instance, if the ADSL specification includes a variable named AGECAT, this idea would aim to map this variable with the "Age Group" value in a demographic table template. This can be done through techniques such as similarity analysis of ADaM dataset metadata. Further work can examine approaches using prompt engineering to explore different types of analysis. Once this process is completed, a review can be performed in order to validate the results.

INTRODUCTION

Large language models including, the concept of Generative AI has come into the public discourse in the past couple of years. However, the underlying mathematics and science has been around since at least 1950, since computer scientist Alan Turing published his paper *Computing Machinery and Intelligence*. In his paper, he discusses a concept he calls the "Imitation Game" where he tests a computers ability to be able to produce answers that are indistinguishable from a human being. A computer that successfully passes this test is said to have passed "Turing Test" and is held as a standard to prove that the machine possesses human like intelligence [Turing., 1950].

Recently, ChatGPT-4 has made headlines for passing the Turing test [Mei at al., 2023]. ChatGPT-4 has been shown to be human-like in its ability to process information. This means is that ChatGPT-4 can use machine learning and context clues in order to make informed decisions and give more accurate answers, to yield similar answers to a human. Generative AI serves to bridge the gap between human and computer code by being able to perform complex analyses through English prompts, rather than code.

In this paper using Generative AI, we will explore matching TLF variable labels to ADaM variable names in order to create a shell that can easily be interpreted by both programmers and statisticians. This will lead to better communication and ultimately improved efficiency and productivity. This lower-code approach allows for the developer to experiment with different prompts in order to engineer the prompt that yields the most accurate results.

PROMPT ENGINEERING

In order to get the most accurate results, the Generative AI must be fed information that teaches it to perform the task at hand. Sometimes this can be in the form of more precise prompt or providing context for the machine to define the outlines of the solution sought. The following illustrations show the approaches to prompt engineering that were explored:

DEMOGRAPHY TABLE EXAMPLE

Consider the following ADSL metadata:

	A	B	C	D	E	F	G	H	I	J	K
1	**Unnamed: 0**	**Variable**	**Type**	**Len**	**DLen**	**Format**	**InFormat**	**Label**			
2	1	STUDYID	Char	11	11	$ 11 .		Study Identifier			
3	2	DOMAIN	Char	00 8	00 8	$ 8 .		Domain Abbreviation			
4	3	USUBJID	Char	40 15	01 15	$ 15 .		Unique Subject Identifier			
5	4	SITEID	Char	4	4	$ 4 .		Study Site Identifier			
6	5	REGION	Char	00 8	8	$ 8 .		Region			
7	6	AGE	Num	00 8	10	BEST 10 .		Age			
8	7	AGECAT	Char	10	10	$ 10 .		Age category			
9	8	AGECATN	Num	00	10	BEST10 .		Age category Numeric			
10	9	BRTHDT	Num	00 8	9	DATE9 .		Birth Date			
11	10	SEX	Char	1	1	$ 1 .		Sex			
12	11	SEXN	Num	00 8	10	BEST10 .		Sex (Numeric code]			
13	12	RACE	Char	41	41	$ 41 .		Race			
14	13	RACEN	Num	8	10	BEST10 .		Race (Numeric code]			
15	14	RACEOTH	Char	50	50	$ 50 .		Other Race , Specify			
16	15	RACE CAT	Char	12	12	$ 12 .		Race category			
17	16	RACECATN	Num	8	10	BEST10 .		Race category Numeric			
18	17	RANDFL	Char	1	1	$ 1 .		Randomized Population Flag			
19	18	SAFFL	Char	1	1	$ 1 .		Safety Population Flag			
20	19	ITTFL	Char	1	1	$ 1 .		Intent - To - Treat Population Flag			
21	20	PPROTFL	Char	1	1	$ 1 .		Per - Protocol Population Flag			
22	21	PKFL	Char	1	1	$ 1 .		PK Population Flag			
23	22	COMPLFL	Char	1	1	$ 1 .		Completers Population Flag			
24	23	TERMFL	Char	1	1	$ 1 .		Patient terminated study Flag			
25	24	TERMOT	Num	8	9	DATE9 .		Termination Date			
26	25	TRT1A	Char	30	30	$ 30 .		Actual Treatment Group			
27	26	TRT1AN	Num	8	10	BEST10 .		Actual Treatment Group Numeric Code			
28	27	TRT1P	Char	30	30	$ 30 .		Planned Treatment for Period 1			
29	28	TRT1PN	Num	00	10	BEST10 .		Planned Treatment for Period 1 Numeric			
30	29	RANDOT	Num	00	9	DATE9 .		Randomization Date			
31	30	TRTSTOT	Num	00	9	DATES .		Date of First Exposure to Treatment			
32	31	TRTENDT	Num	00	9	DATE9 .		Date of Last Exposure to Treatment			
33	32	RFSTDTC	Char	00 19	19	$ 19 .		Reference Start Date [char]			
34	33	RFSTDT	Num	00	9	DATE9 .		Reference Start Date			
35	34	RFENDTC	Char	00 19	19	$ 19 .		Reference End Date (char)			
36	35	RFENDT	Num	00	9	DATE9 .		Reference End Date			
37	36	RFENDY	Num	00 8	8	8 .		Reference End Day			
38	37	ICDTC	Char	15	15	$ 15 .		Informed Consent Date			
39	38	ICDT	Num	8	9	DATE9 .		Informed Consent Date - Numeric			
40	39	ETERMSP	Char	200	200	$ 200 .		Reason for terminating the study early			
41	40	ETERMN	Num	8	10	BEST10 .		Reason for terminating early , Numeric			
42											
43											
44											

sheet1 sheet3 +

Note: Data is mocked up and does not represent any real-life scenario.

Consider the following demography table:

Variable	Unnamed: 1	Placebo	Drug 50mg	Drug	Competitor
Age					
	N	5	4	4	3
	Mean (SD)	34.4 (8.71)	30.0 (6.16)	32.0 (6.16)	32.0 (5.29)
	Median	37	32.5	34.5	30
	Q1 - Q3	37.0 - 39.0	26.0 - 34.0	28.0 - 36.0	28.0-38.0
	Min - Max	19 - 40	21-34	23-36	28-38
Age Group					
	18 to 29	1 (20.0 %)	1 (25.0 %)	1 (25.0 %)	1 (33.3 %)
	30 to 39	3 (60.0 %)	3 (75.0 %)	3 (75.0 %)	2 (66.7 %)
	40 to 49	1 (20.0 %)	0 (0.0 %)	0 0.0 %)	0 (0.0 %)
Gender					
	Male	3 (60.0 %)	2 (50.0 %)	2 (50.0 %)	1 (33.3 %)
	Female	2 (40.0 %)	2 (50.0 %)	2 (50.0 %)	2 (66.7 %)
Race					
	White	4 (80.0 %)	4 (100.0 %)	3 (75.0 %)	2 (66.7 %)
	Black or African American	1 (20.0 %)	0 (0.0 %)	1 (25.0 %)	1 (33.3 %)
Country	USA	4 (80.0 %)	4 (100.0 %)	3 (75.0 %)	2 (66.7 %)
	Other	1 (20.0 %)	0 (0.0 %)	1 (25.0 %)	1 (33.3 %)

Note: Table is mocked up and does not represent any real-life scenario.

The first step of this process is constructing the different pieces of the prompt. Since we must match the table label to the ADaM variable, it may initially make sense to construct a prompt like the following:

```
match and return the elements closest to or equal to to these elements: Age, Age Group, Gender, Race, Country from the following list of elements: STUDYID, DOMAIN, USUBJID, SITEID, REGION, AGE, AGECAT, AGECATN, BRTHDT, SEX, SEXN, RACE, RACEN, RACEOTH, RACE CAT, RACECATN, RANDFL, SAFFL, ITTFL, PPROTFL, PKFL, COMPLFL, TERMFL, TERMOT, TRT1A, TRT1AN, TRT1P, TRT1PN, RANDOT, TRTSTOT, TRTENDT, RFSTDTC, RFSTDT, RFENDTC, RFENDT, RFENDY, ICDTC, ICDT, ETERMSP, ETERMN.
```

However, this prompt leaves out certain key information that makes the matching process more difficult, yielding less accurate results. We can then add more qualifying information from the ADSL metadata such as labels and construct a better prompt like the following:

Match this group of labels: Age, Age Group, Gender, Race, Country from the following list of labels A: Study Identifier, Domain Abbreviation, Unique Subject Identifier, Study Site Identifier, Region, Age, Age category, Age category Numeric, Birth Date, Sex, Sex (Numeric code], Race, Race (Numeric code], Other Race , Specify, Race category, Race category Numeric, Randomized Population Flag, Safety Population Flag, Intent - To - Treat Population Flag, Per - Protocol Population Flag, PK Population Flag, Completers Population Flag, Patient terminated study Flag, Termination Date, Actual Treatment Group, Actual Treatment Group Numeric Code, Planned Treatment for Period 1, Planned Treatment for Period 1 Numeric, Randomization Date, Date of First Exposure to Treatment, Date of Last Exposure to Treatment, Reference Start Date [char], Reference Start Date, Reference End Date (char), Reference End Date, Reference End Day, Informed Consent Date, Informed Consent Date - Numeric, Reason for terminating the study early, Reason for terminating early Numeric. Each of these labels to the following variables: STUDYID, DOMAIN, USUBJID, SITEID, REGION, AGE, AGECAT, AGECATN, BRTHDT, SEX, SEXN, RACE, RACEN, RACEOTH, RACE CAT, RACECATN, RANDFL, SAFFL, ITTFL, PPROTFL, PKFL, COMPLFL, TERMFL, TERMOT, TRT1A, TRT1AN, TRT1P, TRT1PN, RANDOT, TRTSTOT, TRTENDT, RFSTDTC, RFSTDT, RFENDTC, RFENDT, RFENDY, ICDTC, ICDT, ETERMSP, ETERMN. Return the closest matches.

, then we will be able to more accurately match the variable to table label. Here is the result of this prompt, successfully matching the table labels to their corresponding ADaM variables:

Based on the provided labels, the closest matches from the last list to the first list are as follows:

- Age: AGE
- Age Group: AGECAT
- Gender: SEX
- Race: RACE
- Country: REGION

These are the closest matches based on the provided labels. If you need further assistance or if there's anything else I can help you with, feel free to ask!

ANNOTATION PROCESS USING R AND CHATGPT

Combining R and ChatGPT allows us to use ChatGPT to solve the mapping problem and R to help place the annotations where they belong on the TLF mock.

DISPOSITION LISTING EXAMPLE

Using ChatGPT's API in R, we can automate the physical annotation process. This method would entail using the Generative AI prompts to extract the variables that match best and then place them on the TLF mockup shell.

First, we need to extract the response to the prompt. The response looks like the following for a disposition listing:

```
"USUBJID|DSSTDY|DSTERM|DSCAT|DSSCAT"
```

Then we split up the string into an array:

```
var <- strsplit(ex7, split = "\\|")
print (var)
var1 <- t (t (data.frame (var)))
```

Then using the following statement, we can add the annotations to the listing.

```
comb <- paste(dscr,'<',c (var1),'>')
```

Finally, we output the annotated TLF to an RTF file:

Subject Identifier < USUBJID >	Study Day < DSSTDY >	Disposition Event < DSTERM >	Reason < DSCAT >	Description < DSSCAT >
xxx	xx	xxxx	xxxx	xxxxx
	xx	xxxx	xxxx	xxxxx
	xx	xxxx	xxxx	xxxxx
	xx	xxxx	xxxx	xxxxx
		xxxx	xxxx	xxxxx
xxxx		xxxx	xxxx	xxxxx
		xxxx	xxxx	xxxxx
		xxxx	xxxx	xxxxx
		xxxx	xxxx	xxxxx

POTENTIAL FUTURE DEVELOPMENTS

This proof of concept can be extended to an app that can map ADaM variables to TLF mock shells on a larger scale using different types of ADaM datasets and TLF mockups. The prompts would require fine tuning to match the ADaM variable to the TLF label. There are a few techniques we can use to improve this process such as uploading the completed ADaM datasets to the Generative AI platform.

As mapping is used in many different processes within the pharmaceutical world, there are other potential use cases that can leverage a similar approach. For example, we could use a similar approach for mapping Raw data to SDTM data, Raw data to eCRFs and SDTM to ADaM data.

It is also possible, but technically challenging to build a pipeline that can automate the process of generating TLF code from specifications using prompt engineering to automate repetitive programming tasks.

CONCLUSION

Generative AI with its user-friendly nature can be used by non-technical user base. The users need to be trained on prompt engineering, but this is an easier process than training someone on a programming language.

Individuals from several different types of backgrounds such as statisticians and programmers can benefit from this methodology, as it creates traceability that cuts down on unnecessary confusion for what

variables need to be used. More work needs to be done to develop and validate a tool such as this before it can be used in a production environment.

REFERENCES

Turing, A.M Oct., 1950 . **COMPUTING MACHINERY AND INTELLIGENCE** . Oxford, England

Mei, Xie, Yuan, Jackson August 12 2023. **A Turing Test of whether AI chatbots are similar to humans**

David Nield August 12 2023 17 Tips to take Your ChatGPT Prompts to the next Level.

RECOMMENDED READING

- *Developing Apps with GPT-4 and ChatGPT – Olivier Caelen, Marie-Alice Blete - O'Reilly Publications*

CONTACT INFORMATION <HEADING 1>

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

Vidya Gopal
Astrazeneca
vidya.gopal@astrazeneca.com