

Balance between Human and Machine

- How NLP helps translate text specification to code, and vice versa

Hao Chen, Yiwen Wang, AstraZeneca China Biometrics, Shanghai, China

Jaimin Patel, AstraZeneca US Biometrics, Delaware, US

ABSTRACT

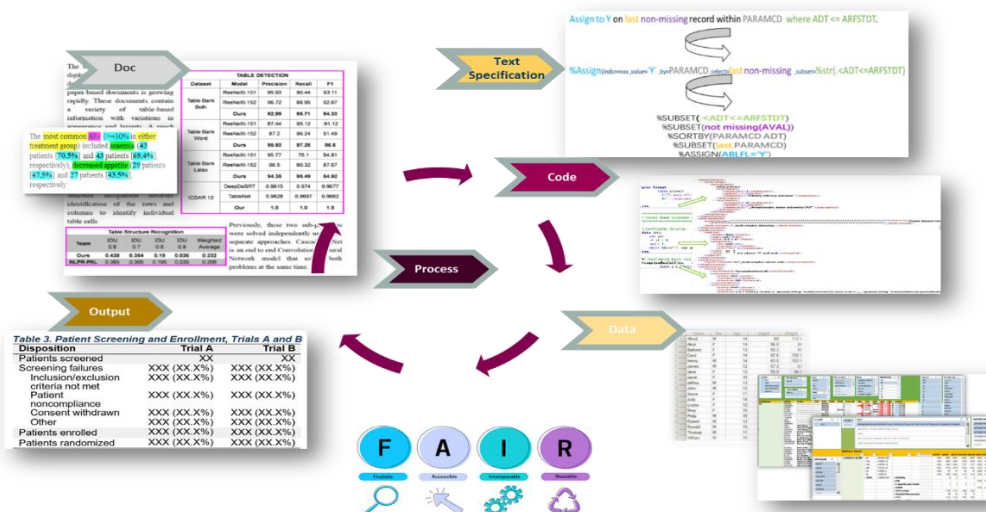
Clinical data analysis algorithms are complex and usually pre-planned in human-readable text specification before execution by machine-readable programming codes. Programmers spend lots of efforts on creating the text specifications and translating them into executable codes, and this work is not one-time effort with the change from source data and code debugging process, so keeping synchronization of the specifications and code is also not an easy work. This paper introduces how natural language processing (NLP) can help us solve this problem, utilizing the natural language understanding (NLU) to help translate from text specifications to codes, and natural language generation (NLG) to translate from codes back to text specifications when needed. This approach will be demonstrated with an Rshiny app for SDTM mapping process. We also explored the ChatGPT solution and discussed about the potential of NLP for other Biometrics tasks in this framework.

INTRODUCTION

The current data flow of traditional clinical trials involves a convoluted and non-linear process that requires a great deal of manual effort and often leads to rework and inefficiencies. However, the future state of clinical trials involves a more streamlined and interoperable approach. This will be made possible through the standardization of data and vocabulary, as well as through leveraging the insights gained from AI and machine learning analysis. By embracing these advancements, the clinical trial process can become more efficient and effective, ultimately leading to better outcomes for patients.

In clinical research, digitalization of trial artifacts, including documentation, programming codes, data, output tables, and complex processes, is critical for achieving FAIRness (Findable, Accessible, Interoperable, and Reusable). By converting these artifacts into metadata and systematically managing them, clinical data analysis can be conducted more efficiently, leading to better decision-making.

Natural Language Processing (NLP) is a field of AI that aims to bridge the gap between human communication and computer understanding. It is essential for processing large amounts of textual data and resolving ambiguity in language to structure unstructured data sources. NLP has two major components: Natural language understanding (NLU) and Natural language generation (NLG). NLU helps computers understand human language by parsing text and extracting meaning, while NLG generates human-like text. This technology can be used for various applications, such as translating text to code and vice versa. Both rule-based systems and machine learning can be used to implement NLU and NLG, depending on the complexity of the language task. Rule-based systems are effective for highly structured and constrained language, while machine learning is preferred for more complex and varied language tasks.



BUSINESS PROBLEM

Developing clinical data analysis algorithms is a complex process that typically requires pre-planning in human-readable text. However, strictly adhering to standard algorithm descriptions can be challenging for human users. While Graphical User Interfaces (GUIs) can provide an alternative means of building flexible algorithms, they are often difficult to use. As a result, algorithms written in free text are commonly used in the industry, leading to diverse mapping specifications across therapeutic areas and companies. Translating human-readable text to machine-readable code is a time-consuming process, and ongoing updates are necessary to accommodate changes in source data and the debugging process. In daily work, synchronizing specifications and code can present a challenging task for human users.

Define-XML Version 2 [Variable Level Metadata]

Demographics (DM) [Location: dm.xml]

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
STUDYID	Study Identifier	1	text	40		Protocol	
DOMAIN	Domain Abbreviation		text	2		Assigned	Two-character abbreviation for the domain most relevant to the observation. The Domain abbreviation is also used as a prefix for variables to ensure uniqueness when datasets are merged.
USUBJID	Unique Subject Identifier	2	text	20		Derived	Concatenation of STUDYID and SUBJID
SUBJID	Subject Identifier for the Study		text	3		CRF Page 1	
RFSTDTC	Subject Reference Start Date/Time		datetime		ISO8601	Derived	RFSTDTC = First date/time of study drug treatment
RFENDTC	Subject Reference End Date/Time		datetime		ISO8601	Derived	RFENDTC = Date of Completion or date of Withdrawal. Null for screen failures
RFXSTDTC	Date/Time of First Study Treatment		datetime		ISO8601	Derived	RFXSTDTC = Date/time of first exposure to protocol specified treatment from CRF
RFXENDTC	Date/Time of Last Study Treatment		datetime		ISO8601	Derived	RFXENDTC = Date/time of last exposure to protocol specified treatment from CRF

To address these challenges, leveraging AI and NLP technologies can make a significant difference in the clinical data analysis process. These technologies can help automate the translation of human-readable text to machine-readable code, reducing the time required for updates and synchronization. Additionally, AI and NLP can help enhance the flexibility and user-friendliness of algorithms, resulting in more effective clinical data analysis.

CASE SHARING

As a case sharing exercise, we present a proposed solution for the SDTM Mapping case, which involves translating Text Specifications to Code using AI and NLP technologies. Our proposed solution includes SAS/R code generation, and algorithm text generation. By automating the translation of Text Specifications to Code, our solution can significantly reduce the time required for updates and synchronization, while also enhancing the user-friendliness and flexibility of algorithms. Through this case sharing exercise, we aim to demonstrate the potential of AI and NLP technologies in facilitating the translation of human-readable text to machine-readable code, revolutionizing clinical data analysis, and enabling more effective and efficient decision-making in the field.

OBJECT

The SDTM mapping file is typically composed of multiple sheets in an Excel file, which contain mapping algorithms at different levels, including data, variable, and value levels. Meanwhile, the target SAS/R codes consist of several sections, such as the header, macro variable definition, variable attributes definition, data join, and variable and value level derivation. To facilitate the generation of SDTM data from the Excel mapping spec, an automated process is designed to analyze the Excel file and produce the corresponding SAS/R code for execution.

How to translate the Text Spec into codes for SDTM Code Generation?

LBORRES = For RAW.LB records:

```

IF RAW.LB MODULE="LB3" and RAW.LB.LBVIDP not missing then set to upcase(value(RAW.LB.LBVIDP)); else if
RAW.LB MODULE="LB3" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB4"
and RAW.LB.LBVIDP not missing then set to upcase(value(RAW.LB.LBVIDP)); else if RAW.LB MODULE="LB4" and
RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB6" and RAW.LB.LBVIDP not
missing then set to upcase(value(RAW.LB.LBVIDP)); else if RAW.LB MODULE="LB6" and RAW.LB.LBORRES not missing then
set to RAW.LB.LBORRES; else set to RAW.LB.LBORRES.

```

C3 Course Development Worksheet							
1. Course Description: This is a pre-graduate course in CS&IT 2. Course Objectives: This course is designed to provide students with a solid foundation in the fundamentals of computer science and information technology. The course covers the following topics: 3. Course Prerequisites: Students must have completed the following courses before enrolling in this course: 4. Course Materials: The following textbooks and materials are required for this course: 5. Course Schedule: The course is scheduled to run from September to December, with a total of 16 weeks of instruction. 6. Course Evaluation: Students will be evaluated based on their performance in the following areas: 7. Course Contact Information: For more information about this course, please contact the course coordinator at [email address].							
Variable	Variable Label	Type	Length	Associated Variable or Formula	Output	Class	Associated Solution
CS0001	Course Number	C	10	CS0001	Course Number	CS0001	Course Number
CS0002	Course Title	C	10	CS0002	Course Title	CS0002	Course Title
CS0003	Course Description	C	10	CS0003	Course Description	CS0003	Course Description
CS0004	Course Prerequisites	C	10	CS0004	Course Prerequisites	CS0004	Course Prerequisites
CS0005	Course Materials	C	10	CS0005	Course Materials	CS0005	Course Materials
CS0006	Course Schedule	C	10	CS0006	Course Schedule	CS0006	Course Schedule
CS0007	Course Evaluation	C	10	CS0007	Course Evaluation	CS0007	Course Evaluation
CS0008	Course Contact Information	C	10	CS0008	Course Contact Information	CS0008	Course Contact Information
CS0009	Course Objectives	C	10	CS0009	Course Objectives	CS0009	Course Objectives
CS0010	Course Prerequisites	C	10	CS0010	Course Prerequisites	CS0010	Course Prerequisites
CS0011	Course Materials	C	10	CS0011	Course Materials	CS0011	Course Materials
CS0012	Course Schedule	C	10	CS0012	Course Schedule	CS0012	Course Schedule
CS0013	Course Evaluation	C	10	CS0013	Course Evaluation	CS0013	Course Evaluation
CS0014	Course Contact Information	C	10	CS0014	Course Contact Information	CS0014	Course Contact Information
CS0015	Course Objectives	C	10	CS0015	Course Objectives	CS0015	Course Objectives
CS0016	Course Prerequisites	C	10	CS0016	Course Prerequisites	CS0016	Course Prerequisites
CS0017	Course Materials	C	10	CS0017	Course Materials	CS0017	Course Materials
CS0018	Course Schedule	C	10	CS0018	Course Schedule	CS0018	Course Schedule
CS0019	Course Evaluation	C	10	CS0019	Course Evaluation	CS0019	Course Evaluation
CS0020	Course Contact Information	C	10	CS0020	Course Contact Information	CS0020	Course Contact Information
CS0021	Course Objectives	C	10	CS0021	Course Objectives	CS0021	Course Objectives
CS0022	Course Prerequisites	C	10	CS0022	Course Prerequisites	CS0022	Course Prerequisites
CS0023	Course Materials	C	10	CS0023	Course Materials	CS0023	Course Materials
CS0024	Course Schedule	C	10	CS0024	Course Schedule	CS0024	Course Schedule
CS0025	Course Evaluation	C	10	CS0025	Course Evaluation	CS0025	Course Evaluation
CS0026	Course Contact Information	C	10	CS0026	Course Contact Information	CS0026	Course Contact Information
CS0027	Course Objectives	C	10	CS0027	Course Objectives	CS0027	Course Objectives
CS0028	Course Prerequisites	C	10	CS0028	Course Prerequisites	CS0028	Course Prerequisites
CS0029	Course Materials	C	10	CS0029	Course Materials	CS0029	Course Materials
CS0030	Course Schedule	C	10	CS0030	Course Schedule	CS0030	Course Schedule
CS0031	Course Evaluation	C	10	CS0031	Course Evaluation	CS0031	Course Evaluation
CS0032	Course Contact Information	C	10	CS0032	Course Contact Information	CS0032	Course Contact Information
CS0033	Course Objectives	C	10	CS0033	Course Objectives	CS0033	Course Objectives
CS0034	Course Prerequisites	C	10	CS0034	Course Prerequisites	CS0034	Course Prerequisites
CS0035	Course Materials	C	10	CS0035	Course Materials	CS0035	Course Materials
CS0036	Course Schedule	C	10	CS0036	Course Schedule	CS0036	Course Schedule
CS0037	Course Evaluation	C	10	CS0037	Course Evaluation	CS0037	Course Evaluation
CS0038	Course Contact Information	C	10	CS0038	Course Contact Information	CS0038	Course Contact Information
CS0039	Course Objectives	C	10	CS0039	Course Objectives	CS0039	Course Objectives
CS0040	Course Prerequisites	C	10	CS0040	Course Prerequisites	CS0040	Course Prerequisites
CS0041	Course Materials	C	10	CS0041	Course Materials	CS0041	Course Materials
CS0042	Course Schedule	C	10	CS0042	Course Schedule	CS0042	Course Schedule
CS0043	Course Evaluation	C	10	CS0043	Course Evaluation	CS0043	Course Evaluation
CS0044	Course Contact Information	C	10	CS0044	Course Contact Information	CS0044	Course Contact Information
CS0045	Course Objectives	C	10	CS0045	Course Objectives	CS0045	Course Objectives
CS0046	Course Prerequisites	C	10	CS0046	Course Prerequisites	CS0046	Course Prerequisites
CS0047	Course Materials	C	10	CS0047	Course Materials	CS0047	Course Materials
CS0048	Course Schedule</						

[illegible]

SAS

[illegible]

PROCESS

In the context of Text to Code generation, the following five steps can be identified:

1. Pre-processing: In some cases, long texts may need to be split into shorter sentences to facilitate subsequent processing.
2. Named Entity Recognition (NER): Variables and formulas can be masked using NER to identify their presence and position within the text.
3. Normalization: The text can be semantically matched to a standard algorithm using embedded vectors to ensure consistency and facilitate parsing.
4. Parsing: The normalized algorithm can be parsed into an Abstract Syntax Tree (AST) for more structured processing.
5. Translation: Finally, the AST can be translated into code that can be executed by a computer.

When generating Text from Code, the following two steps can be followed:

1. Code Parsing: The input code can be parsed into an Abstract Syntax Tree (AST) for more structured processing.
2. Translation: The AST can be translated into text, or into other programming languages like R, Python or SQL if needed.

In the above steps, Named Entity Recognition (NER) and Normalization rely on machine learning techniques, while the other steps are based on rule-based algorithms.

NER (Named Entity Recognition)

General example:

On the 15th of September DATE , Tim Cook PERSON announced that Apple ORG wants to acquire ABC Group ORG from New York GPE for 1 billion dollars MONEY

Domain-specific:

```

If SMOKSTAT VAR in ('CURRENT', 'FORMER') FUNC then 'SMOKER' QUOT ; else if
SMOKSTAT VAR eq 'NEVER' QUOT then 'NON-SMOKER' QUOT .

```

Normalization (Spec smart match)

Flexible Spec	Variable masked	Standard Spec	
inputSpec	Input Spec normalized	Standard Spec matched	Similarity
0 Baseline flag	BASLINE FLAG	DERIVE BASLINE FLAG	90.0%
1 Convert date (VISDAT/VSDAT) to ISO8601 date.	CONVERT DATE (\$\$/\$\$) TO ISO8601 DATE	CONVERT DATE (\$\$/\$\$) TO ISO8601 DATE FORMAT	95.0%
2 Set to RAW.AE.AENO	SET TO \$\$	SET TO \$\$	100.0%
3 Convert To Char	CONVERT TO CHAR	CONVERT TO CHARACTER VERSION	90.0%
4 Convert To SAESEV.Format	CONVERT TO \$\$\$.FORMAT	CONVERT TO \$XX .FORMAT	95.0%
5 Sort by variable STUDYID USUBJID AESTDTC AEDECOD. Restart the numbering at each USUBJID with 1	SORT BY VARIABLE \$\$ \$ \$ \$ \$ \$ \$. RESTART THE NUMBERING AT EACH \$ \$ WITH 1	SORT BY VARIABLE \$ \$ \$ \$ \$ \$. RESTART THE NUMBERING AT EACH \$ \$ WITH 1 (ORDER SEQUENTIALLY)	95.0%
6 Concatenate (RAW.AE.AESTDAT and RAW.AE.AESTTIM) by '/'	CONCATENATE (\$\$ AND \$\$) BY @@	CONCATENATE \$ \$ \$ BY @@	95.0%
7 Calculation of AESTDY = (Number version of date of AESTDTC - Numeric version of date of DM.RFSTDTC). Note1: Add 1 if greater than or equal to 0.	CALCULATION OF \$\$ = (NUMBER VERSION OF DATE OF \$\$ - NUMERIC VERSION OF DATE OF \$\$). NOTE1: ADD 1 IF GREATER THAN OR EQUAL TO 0.	CALCULATION OF \$\$ = (NUMERIC VERSION OF \$\$ - NUMERIC VERSION OF \$\$). NOTE: ADD 1 IF CALCULATION IS GREATER THAN OR EQUAL TO 0	89.6%

SOLUTION

Here is the process of how we convert the text algorithms into SAS code, taking variable LBORRES derivation for example:

Variable Name	Variable Label	Type	Length	Controlled Terms or Format	Origin	Conversion Definition
LBORRES	Result or Finding in Original Unit					For RAW.PREG records: If RAW.PREG MODULE="PREG" then set to upcase(vvalue(RAW.PREG.PREGTSTU)); else if RAW.PREG MODULE="PREG1" then set to upcase(vvalue(RAW.PREG.PREGTSTS)); For RAW.LB records: If RAW.LB MODULE="LB3" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB3" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB4" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB4" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB5" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB5" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB6" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB6" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else set to RAW.LB.LBORRES; For RAW.CLAB records: Set to upcase of RAW.CLAB.LBORRES.
LBORRES					RAW.PREG	For RAW.PREG records: If RAW.PREG MODULE="PREG" then set to upcase(vvalue(RAW.PREG.PREGTSTU)); else if RAW.PREG MODULE="PREG1" then set to upcase(vvalue(RAW.PREG.PREGTSTS));
LBORRES					RAW.LB	For RAW.LB records: If RAW.LB MODULE="LB3" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB3" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB4" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB4" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB5" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB5" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB6" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB6" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else set to RAW.LB.LBORRES;
LBORRES					RAW.CLAB	For RAW.CLAB records: Set to upcase of RAW.CLAB.LBORRES.

1. Pre-process

LBORRES = For RAW.LB records:

If RAW.LB MODULE="LB3" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB3" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB4" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB4" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB5" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB5" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB6" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB6" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else set to RAW.LB.LBORRES;

2. NER

3. Normalization

If RAW.LB MODULE="LB3" FORMULA and RAW.LB.LBVDIP VAR not missing then set to upcase(vvalue(RAW.LB.LBVDIP)) FUNC; else if RAW.LB MODULE="LB3" FORMULA and RAW.LB.LBORRES VAR not missing then set to RAW.LB.LBORRES VAR; else if RAW.LB MODULE="LB4" FORMULA and RAW.LB.LBVDIP VAR not missing then set to upcase(vvalue(RAW.LB.LBVDIP)) FUNC; else if RAW.LB MODULE="LB4" FORMULA and RAW.LB.LBORRES VAR not missing then set to RAW.LB.LBORRES VAR; else if RAW.LB MODULE="LB5" FORMULA and RAW.LB.LBVDIP VAR not missing then set to upcase(vvalue(RAW.LB.LBVDIP)) FUNC; else if RAW.LB MODULE="LB5" FORMULA and RAW.LB.LBORRES VAR not missing then set to RAW.LB.LBORRES VAR; else if RAW.LB MODULE="LB6" FORMULA and RAW.LB.LBVDIP VAR not missing then set to upcase(vvalue(RAW.LB.LBVDIP)) FUNC; else if RAW.LB MODULE="LB6" FORMULA and RAW.LB.LBORRES VAR not missing then set to RAW.LB.LBORRES VAR; else set to RAW.LB.LBORRES VAR.

Spec (Human Language + SAS code piece)

AST (Abstract Syntax Tree)

4. Parse

5. Translation

```
/* For RAW.LB records: If RAW.LB MODULE="LB3" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB3" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB4" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB4" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB5" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB5" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB MODULE="LB6" and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB MODULE="LB6" and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else set to RAW.LB.LBORRES; */
```

SAS programs (or R/Python/SQL)

Achieving full automation in the process can be challenging, and it is important to involve humans in the review and decision-making process for new scenarios, while also adding new patterns to the knowledge base as needed. For example, if the derivation for a variable such as AEACN is unrecognizable and cannot be parsed into SAS codes, the system will automatically flag this new scenario, requiring a programmer to write ad hoc open code to handle it. If this pattern occurs frequently, we can add it to the system library for future re-use.

Variable	Conversion Definition	sas	flag
STUDYID	" 00008"	STUDYID=" 00008"	Y
USUBJID	Concatenate STUDY and SUBJECT separating by '/'. (e.g., 00008/ 001).	USUBJID=cat(strip(STUDY), '/', strip(SUBJECT))	Y
AESEQ	Sort by variable STUDYID USUBJID AESTDTC AEDECOD. Restart the numbering at each	%sdtm_seq(indata=&syslast, domain=&sdtm_domain.);	Y
AESPID	Set to "AE-" left(trim(put(RAW.AE.AENO, best.))) if RAW.AE.AENO is not missing.	if ^ missing(AENO) then do; AESPID = "AE-" left(trim(put(AENO, best.))); end;	Y
AETERM	Set to RAW.AE.AETERM.	*AETERM=AETERM; *copy/keep;	Y
AELLT	Set to RAW.AE.LLT_NAME	AELLT=LLT_NAME	Y
AECAT	Assgin RAW.AE.AECAT as format if RAW.AE.AECATOCC=1	if AECATOCC=1 then do; AECAT = ifc(AECAT = "1", "ANAPHYLAXIS AND OTHER SERIOUS HYPERSENSITIVITY REACTIONS", ifc(AECAT = "2", "INFUSION-RELATED REACTIONS"	Y
AEACN	Set to RAW.AE.AEACN with below conversion. Note: 0 = DOSE NOT CHANGED, 3 = DRUG INTERRUPTED, 4 = DRUG WITHDRAWN, 98 = NOT APPLICABLE	AEACN= AEACN with below conversion. Note: 0 = DOSE NOT CHANGED, 3 = DRUG INTERRUPTED, 4 = DRUG WITHDRAWN, 98 = NOT APPLICABLE	N
AEREL	Assign RAW.AE.AEREL as format	AEREL =ifc(AEREL = "0", "NOT RELATED", ifc(AEREL = "1", "RELATED",	Y
AESCONG	When AESER = "Y": Assign RAW.SERAE.AESCONG as format.	if AESER="Y" then do; AESCONG = ifc(AESCONG = "0", "N", ifc(AESCONG = "1", "Y", ifc(AESCONG = "C49487", "N", ifc(AESCONG = "C49488", "Y", AESCONG)))) ; end;	Y
AECONTRT	Set to substr (vvalue (RAW.AE.AETRT), 1, 1)	AECONTRT=substr(vvalue(AETRT), 1, 1)	Y

It is crucial to note that when dealing with highly complex study-specific derivations, the chances of success for NLP analysis and code auto generation are limited, and the opportunity for reusability may also be limited. Therefore, in such cases, it may be necessary to manually perform the derivation using user-defined complex algorithms written in open code.

Taking all the scenarios into consideration, we divide the derivations into three types, including Type I (simple derivation like copy/drop/rename or function call), Type II (standard derivation involving macro calls like standard mapping or user-defined pattern), and Type III (complex derivation involving user-defined complex algorithms that typically requires manual coding). NLP-powered automation can be useful for Types I and II, while Type III is best left to human intelligence.

• Type I (from Spec) – simple derivation

- Copy/Drop/Rename
- Assign - function call

• Type II (from Spec) – standard derivation (macro call)

- Standard Mapping macro call (top down)
- User-defined Pattern macro call, some chance to reuse (bottom up)

• Type III (from open code) – complex derivation

- User-defined complex algorithm, little chance to reuse
- Do it in open code manually
- Rerun applicable with updated Code & Spec

```

206 _domain= 'AE';
207 _module= 'AE SERAE';
208 _semdt=put( aedis. INT.);
209 _semdt= %dtc(INVAR_DAT= ae.semdat , INVAR_TTM= ae.semtia , OUTVAR_DTC= ae._semdtc);
210 _seout=upcase(put( _seout, vformat(_seout)));
211 _secl=put( _secl, $AESELF.);
212 _seer=upcase(put( _seer, vformat(_seer)));
213 _seev=upcase(put( _seev, vformat(_seev)));
214 _sestdt= %dtc(INVAR_DAT= ae.sestdat , INVAR_TTM= ae.semtia , OUTVAR_DTC= ae._sestdtc);
215 _seater=seater;
216 _sehltd=hltdcode;
217 _sehltd=hltdname;
218 _sehltd=hltdcode;
219 _sehltd=hltdname;
220 _ip=ip;
221 run;
222
223 *****
224 * Execute standalone macros
225 *****
226 %dtc(INVAR_REFDTC=templib._dm.rfstdtc , INVAR_DTC=templib._ae.sestdtc , OUTVAR_DT=templib._ae.sestdy);
227 %dtc(INVAR_REFDTC=templib._dm.rfstdtc , INVAR_DTC=templib._ae.sestdtc , OUTVAR_DT=templib._ae.sestdy);
228 %seq(DATASET=LAYOUT=STUDYID USUBJID AESTDTC AEDECOD, OUTVAR_SEQ=templib._ae.seseq);
229
230 *****
231 * Variable TRSPRC
232 *****
233 *****
234 *****
235 *****
236 *****
237 *****
238 *****
239 *****
240 *****
241 *****
242 *****
243 *****
244 *****
245 *****
246 *****
247 *****
248 *****
249 *****
250 *****
251 *****
252 *****
253 *****
254 *****
255 *****
256 *****
257 *****
258 *****
259 *****
260 *****
261 *****
262 *****
263 *****
264 *****
265 *****
266 *****
267 *****
268 *****
269 *****
270 *****
271 *****
272 *****
273 *****
274 *****
275 *****
276 *****
277 *****
278 *****
279 *****
280 *****
281 *****
282 *****
283 *****
284 *****
285 *****
286 *****
287 *****
288 *****
289 *****
290 *****
291 *****
292 *****
293 *****
294 *****
295 *****
296 *****
297 *****
298 *****
299 *****
300 *****
301 *****
302 *****
303 *****
304 *****
305 *****
306 *****
307 *****
308 *****
309 *****
310 *****
311 *****
312 *****
313 *****
314 *****
315 *****
316 *****
317 *****
318 *****
319 *****
320 *****
321 *****
322 *****
323 *****
324 *****
325 *****
326 *****
327 *****
328 *****
329 *****
330 *****
331 *****
332 *****
333 *****
334 *****
335 *****
336 *****
337 *****
338 *****
339 *****
340 *****
341 *****
342 *****
343 *****
344 *****
345 *****
346 *****
347 *****
348 *****
349 *****
350 *****
351 *****
352 *****
353 *****
354 *****
355 *****
356 *****
357 *****
358 *****
359 *****
360 *****
361 *****
362 *****
363 *****
364 *****
365 *****
366 *****
367 *****
368 *****
369 *****
370 *****
371 *****
372 *****
373 *****
374 *****
375 *****
376 *****
377 *****
378 *****
379 *****
380 *****
381 *****
382 *****
383 *****
384 *****
385 *****
386 *****
387 *****
388 *****
389 *****
390 *****
391 *****
392 *****
393 *****
394 *****
395 *****
396 *****
397 *****
398 *****
399 *****
400 *****
401 *****
402 *****
403 *****
404 *****
405 *****
406 *****
407 *****
408 *****
409 *****
410 *****
411 *****
412 *****
413 *****
414 *****
415 *****
416 *****
417 *****
418 *****
419 *****
420 *****
421 *****
422 *****
423 *****
424 *****
425 *****
426 *****
427 *****
428 *****
429 *****
430 *****
431 *****
432 *****
433 *****
434 *****
435 *****
436 *****
437 *****
438 *****
439 *****
440 *****
441 *****
442 *****
443 *****
444 *****
445 *****
446 *****
447 *****
448 *****
449 *****
450 *****
451 *****
452 *****
453 *****
454 *****
455 *****
456 *****
457 *****
458 *****
459 *****
460 *****
461 *****
462 *****
463 *****
464 *****
465 *****
466 *****
467 *****
468 *****
469 *****
470 *****
471 *****
472 *****
473 *****
474 *****
475 *****
476 *****
477 *****
478 *****
479 *****
480 *****
481 *****
482 *****
483 *****
484 *****
485 *****
486 *****
487 *****
488 *****
489 *****
490 *****
491 *****
492 *****
493 *****
494 *****
495 *****
496 *****
497 *****
498 *****
499 *****
500 *****
501 *****
502 *****
503 *****
504 *****
505 *****
506 *****
507 *****
508 *****
509 *****
510 *****
511 *****
512 *****
513 *****
514 *****
515 *****
516 *****
517 *****
518 *****
519 *****
520 *****
521 *****
522 *****
523 *****
524 *****
525 *****
526 *****
527 *****
528 *****
529 *****
530 *****
531 *****
532 *****
533 *****
534 *****
535 *****
536 *****
537 *****
538 *****
539 *****
540 *****
541 *****
542 *****
543 *****
544 *****
545 *****
546 *****
547 *****
548 *****
549 *****
550 *****
551 *****
552 *****
553 *****
554 *****
555 *****
556 *****
557 *****
558 *****
559 *****
560 *****
561 *****
562 *****
563 *****
564 *****
565 *****
566 *****
567 *****
568 *****
569 *****
570 *****
571 *****
572 *****
573 *****
574 *****
575 *****
576 *****
577 *****
578 *****
579 *****
580 *****
581 *****
582 *****
583 *****
584 *****
585 *****
586 *****
587 *****
588 *****
589 *****
590 *****
591 *****
592 *****
593 *****
594 *****
595 *****
596 *****
597 *****
598 *****
599 *****
600 *****
601 *****
602 *****
603 *****
604 *****
605 *****
606 *****
607 *****
608 *****
609 *****
610 *****
611 *****
612 *****
613 *****
614 *****
615 *****
616 *****
617 *****
618 *****
619 *****
620 *****
621 *****
622 *****
623 *****
624 *****
625 *****
626 *****
627 *****
628 *****
629 *****
630 *****
631 *****
632 *****
633 *****
634 *****
635 *****
636 *****
637 *****
638 *****
639 *****
640 *****
641 *****
642 *****
643 *****
644 *****
645 *****
646 *****
647 *****
648 *****
649 *****
650 *****
651 *****
652 *****
653 *****
654 *****
655 *****
656 *****
657 *****
658 *****
659 *****
660 *****
661 *****
662 *****
663 *****
664 *****
665 *****
666 *****
667 *****
668 *****
669 *****
670 *****
671 *****
672 *****
673 *****
674 *****
675 *****
676 *****
677 *****
678 *****
679 *****
680 *****
681 *****
682 *****
683 *****
684 *****
685 *****
686 *****
687 *****
688 *****
689 *****
690 *****
691 *****
692 *****
693 *****
694 *****
695 *****
696 *****
697 *****
698 *****
699 *****
700 *****
701 *****
702 *****
703 *****
704 *****
705 *****
706 *****
707 *****
708 *****
709 *****
710 *****
711 *****
712 *****
713 *****
714 *****
715 *****
716 *****
717 *****
718 *****
719 *****
720 *****
721 *****
722 *****
723 *****
724 *****
725 *****
726 *****
727 *****
728 *****
729 *****
730 *****
731 *****
732 *****
733 *****
734 *****
735 *****
736 *****
737 *****
738 *****
739 *****
740 *****
741 *****
742 *****
743 *****
744 *****
745 *****
746 *****
747 *****
748 *****
749 *****
750 *****
751 *****
752 *****
753 *****
754 *****
755 *****
756 *****
757 *****
758 *****
759 *****
760 *****
761 *****
762 *****
763 *****
764 *****
765 *****
766 *****
767 *****
768 *****
769 *****
770 *****
771 *****
772 *****
773 *****
774 *****
775 *****
776 *****
777 *****
778 *****
779 *****
780 *****
781 *****
782 *****
783 *****
784 *****
785 *****
786 *****
787 *****
788 *****
789 *****
790 *****
791 *****
792 *****
793 *****
794 *****
795 *****
796 *****
797 *****
798 *****
799 *****
800 *****
801 *****
802 *****
803 *****
804 *****
805 *****
806 *****
807 *****
808 *****
809 *****
810 *****
811 *****
812 *****
813 *****
814 *****
815 *****
816 *****
817 *****
818 *****
819 *****
820 *****
821 *****
822 *****
823 *****
824 *****
825 *****
826 *****
827 *****
828 *****
829 *****
830 *****
831 *****
832 *****
833 *****
834 *****
835 *****
836 *****
837 *****
838 *****
839 *****
840 *****
841 *****
842 *****
843 *****
844 *****
845 *****
846 *****
847 *****
848 *****
849 *****
850 *****
851 *****
852 *****
853 *****
854 *****
855 *****
856 *****
857 *****
858 *****
859 *****
860 *****
861 *****
862 *****
863 *****
864 *****
865 *****
866 *****
867 *****
868 *****
869 *****
870 *****
871 *****
872 *****
873 *****
874 *****
875 *****
876 *****
877 *****
878 *****
879 *****
880 *****
881 *****
882 *****
883 *****
884 *****
885 *****
886 *****
887 *****
888 *****
889 *****
890 *****
891 *****
892 *****
893 *****
894 *****
895 *****
896 *****
897 *****
898 *****
899 *****
900 *****
901 *****
902 *****
903 *****
904 *****
905 *****
906 *****
907 *****
908 *****
909 *****
910 *****
911 *****
912 *****
913 *****
914 *****
915 *****
916 *****
917 *****
918 *****
919 *****
920 *****
921 *****
922 *****
923 *****
924 *****
925 *****
926 *****
927 *****
928 *****
929 *****
930 *****
931 *****
932 *****
933 *****
934 *****
935 *****
936 *****
937 *****
938 *****
939 *****
940 *****
941 *****
942 *****
943 *****
944 *****
945 *****
946 *****
947 *****
948 *****
949 *****
950 *****
951 *****
952 *****
953 *****
954 *****
955 *****
956 *****
957 *****
958 *****
959 *****
960 *****
961 *****
962 *****
963 *****
964 *****
965 *****
966 *****
967 *****
968 *****
969 *****
970 *****
971 *****
972 *****
973 *****
974 *****
975 *****
976 *****
977 *****
978 *****
979 *****
980 *****
981 *****
982 *****
983 *****
984 *****
985 *****
986 *****
987 *****
988 *****
989 *****
990 *****
991 *****
992 *****
993 *****
994 *****
995 *****
996 *****
997 *****
998 *****
999 *****
1000 *****

```


INTEGRATED PROCESS

The R Shiny App is developed to provide a comprehensive solution for generating SAS code from an Excel Spec, allowing users to upload the file, select the target domain, and automatically generate the SAS code. If any Type III algorithms require manual coding, users can insert the open code and upload the revised code to the system. Additionally, the Excel Spec can also be updated if necessary. The final code is regenerated by combining the user's code with the system's automatically generated code. This process can be repeated multiple times if necessary, with Type I and II always coming from the Excel Spec and Type III being added by the user.

Here is an example of auto-generated code. Type III part can be updated in the 'Todo placeholder'.

The image shows a SAS code snippet with several annotations in green boxes:

- Program header**: Lines 1-14, including program name, type, purpose, author, date, and version information.
- System macro/variable**: Lines 15-20, defining macros like %sysmacro and %sysvariable.
- Attributes, good format**: Lines 21-24, defining attributes like STUDYID, AESTDY, and AESTDT.
- Rename, Where**: Lines 25-30, using the RENAME and WHERE statements to filter data.
- Codelist**: Lines 31-35, defining codelist values for AESTDY and AESTDT.
- Data Join/Combine**: Lines 36-40, using the JOIN statement to combine data.
- Grouping process**: Lines 41-45, using the GROUP BY statement to group data.
- Variable level**: Lines 46-50, defining variable levels for AESTDY and AESTDT.
- Data level**: Lines 51-55, defining data levels for AESTDY and AESTDT.
- Todo placeholder**: Lines 56-60, a placeholder for user-defined code.

CODE TO ALGORITHM TEXT

As introduced above, when we have the code and want to get the algorithm text, it is easy to use rule-based system to parse the code into an Abstract Syntax Tree (AST) based on which can be translated into algorithm text, or into other programming languages like R, Python or SQL if needed.

SAS function to Algorithm Text Smart Translator

① Paste SAS functions here

One SAS function, can be nested function

```
USUBJID=cat(strip(STUDY),"/",strip(SUBJECT));
```

② Convert into Algorithm Text

Convert

Test

```
USUBJID=Concatenate STUDY and SUBJECT separated by "/"
```

Abstract syntax tree

```
start
None
None
equation
variable USUBJID
=
element
cat
(
element
element
element
strip
(
variable STUDY
)
,
quotedstring /
,
element
strip
(
variable SUBJECT
)
)
```

① Paste SAS functions here

One SAS function, can be nested function

```
ifc(RAW.LB.MODULE="LB3" and "missing(RAW.LB.LBVDIP)",upcase(value(RAW.LB.LBVDIP)),ifc(RAW.LB.MODULE="LB3" and "missing(RAW.LB.LBORRES)",RAW.LB.LBORRES;ifc(RAW.LB.MODULE="LB4" and "missing(RAW.LB.LBORRES)",RAW.LB.LBORRES;ifc(RAW.LB.MODULE="LB6" and "missing(RAW.LB.LBORRES)",upcase(value(RAW.LB.LBVDIP)),ifc(RAW.LB.MODULE="LB6" and "missing(RAW.LB.LBORRES)",RAW.LB.LBORRES))))))
```

② Convert into Algorithm Text

Convert

Test

```
if RAW.LB.MODULE="LB3" and not missing(RAW.LB.LBVDIP) then set to upcase(value(RAW.LB.LBVDIP));
else if RAW.LB.MODULE="LB3" and not missing(RAW.LB.LBORRES) then set to RAW.LB.LBORRES;
else if RAW.LB.MODULE="LB4" and not missing(RAW.LB.LBVDIP) then set to upcase(value(RAW.LB.LBVDIP));
else if RAW.LB.MODULE="LB4" and not missing(RAW.LB.LBORRES) then set to RAW.LB.LBORRES;
else if RAW.LB.MODULE="LB6" and not missing(RAW.LB.LBVDIP) then set to upcase(value(RAW.LB.LBVDIP));
else if RAW.LB.MODULE="LB6" and not missing(RAW.LB.LBORRES) then set to RAW.LB.LBORRES;
else RAW.LB.LBORRES
```

Abstract syntax tree

```
start
None
None
element
ifc
(
element
equation
variable RAW.LB.MODULE
=
quotedstring LB3
and
element
element
missing
(
variable RAW.LB.LBVDIP
)
,
element
element
upcase
,
```

SAS function to R Smart Translator

① Paste SAS functions here

One SAS function, can be nested function

```
USUBJID=cat(strip(STUDY),';',strip(SUBJECT));
```

② Convert into R

Convert

R code

```
USUBJID=paste0(trimws(STUDY,which="both"),';',trimws(SUBJECT,which="both"))
```

Abstract syntax tree

```
start
None
```

① Paste SAS functions here

One SAS function, can be nested function

```
LBORRES = ifc(RAW.LB.MODULE='LB3' and 'missing(RAW.LB.LBVDIP)', upcase(vvalue(RAW.LB.LBVDIP)), ifc(RAW.LB.MODULE='LB3' and 'missing(RAW.LB.LBORRES)', RAW.LB.LBORRES, ifc(RAW.LB.MODULE='LB4' and 'missing(RAW.LB.LBVDIP)', upcase(vvalue(RAW.LB.LBVDIP)), ifc(RAW.LB.MODULE='LB4' and 'missing(RAW.LB.LBORRES)', RAW.LB.LBORRES, ifc(RAW.LB.MODULE='LB6' and 'missing(RAW.LB.LBVDIP)', upcase(vvalue(RAW.LB.LBVDIP)), ifc(RAW.LB.MODULE='LB6' and 'missing(RAW.LB.LBORRES)', RAW.LB.LBORRES, RAW.LB.LBORRES))))))
```

② Convert into R

Convert

R code

```
LBORRES = ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB3' and is.na(RAW.LB.LBVDIP), toupper(RAW.LB.LBVDIP), ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB3' and is.na(RAW.LB.LBORRES), RAW.LB.LBORRES, ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB4' and is.na(RAW.LB.LBVDIP), toupper(RAW.LB.LBVDIP), ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB4' and is.na(RAW.LB.LBORRES), RAW.LB.LBORRES, ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB6' and is.na(RAW.LB.LBVDIP), toupper(RAW.LB.LBVDIP), ifelse(is.na(RAW.LB.MODULE) & RAW.LB.MODULE == 'LB6' and is.na(RAW.LB.LBORRES), RAW.LB.LBORRES, RAW.LB.LBORRES))))))
```

Abstract syntax tree

```
start
None
```

DISCUSSION

CHATGPT SOLUTION TESTING

As a powerful system excels in both NLP and code analysis & generation, ChatGPT has recently made many breakthroughs. So we did similar testing for the task at hand. We found that R code outperforms SAS code due to its access to more open-source training data, and translating code into R before translating it into SAS can improve the quality. Achieving accurate results may require multiple rounds of refinement and adjustment, while controlling the quality of its results can be challenging due to ChatGPT's black box nature. However, ChatGPT is capable of effectively processing very complex text, surpassing the capabilities of many self-developed systems. It's essential to consider the protection of company-restricted or confidential data when using ChatGPT.

write SAS codes for the algorithms below:

```
LBORRES =
IF RAW.LB.MODULE='LB3' and RAW.LB.LBVDIP not missing then set to
upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB.MODULE='LB3' and RAW.LB.LBORRES not
missing then set to RAW.LB.LBORRES; else if RAW.LB.MODULE='LB4' and RAW.LB.LBVDIP not
missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if RAW.LB.MODULE='LB4' and
RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB.MODULE='LB6'
and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)); else if
RAW.LB.MODULE='LB6' and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else
set to RAW.LB.LBORRES;
```

Here is the SAS code for the algorithm you provided:

```
data output_dataset;
set input_dataset;
LBORRES =
if RAW.LB.MODULE='LB3' and RAW.LB.LBVDIP not missing then upcase(vvalue(RAW.LB.LBVDIP))
else if RAW.LB.MODULE='LB3' and RAW.LB.LBORRES not missing then RAW.LB.LBORRES
else if RAW.LB.MODULE='LB4' and RAW.LB.LBVDIP not missing then upcase(vvalue(RAW.LB.LBVDIP))
else if RAW.LB.MODULE='LB4' and RAW.LB.LBORRES not missing then RAW.LB.LBORRES
else if RAW.LB.MODULE='LB6' and RAW.LB.LBVDIP not missing then upcase(vvalue(RAW.LB.LBVDIP))
else if RAW.LB.MODULE='LB6' and RAW.LB.LBORRES not missing then RAW.LB.LBORRES
else RAW.LB.LBORRES;
run;
```

Note that you should replace "input_dataset" and "output_dataset" with the actual names of your input and output datasets, respectively.

write the R codes

Here is the R code for the algorithm you provided:

```
R
output_dataset <- input_dataset %>%
mutate(LBORRES = ifelse(RAW.LB.MODULE == 'LB3' & !is.na(RAW.LB.LBVDIP),
  toupper(RAW.LB.LBVDIP),
  ifelse(RAW.LB.MODULE == 'LB3' & !is.na(RAW.LB.LBORRES),
    RAW.LB.LBORRES,
    ifelse(RAW.LB.MODULE == 'LB4' & !is.na(RAW.LB.LBVDIP),
      toupper(RAW.LB.LBVDIP),
      ifelse(RAW.LB.MODULE == 'LB4' & !is.na(RAW.LB.LBORRES),
        RAW.LB.LBORRES,
        ifelse(RAW.LB.MODULE == 'LB6' & !is.na(RAW.LB.LBVDIP),
          toupper(RAW.LB.LBVDIP),
          ifelse(RAW.LB.MODULE == 'LB6' & !is.na(RAW.LB.LBORRES),
            RAW.LB.LBORRES,
            RAW.LB.LBORRES)))))))
```

Note that you should replace "input_dataset" and "output_dataset" with the actual names of your input and output data frames, respectively.

translate the R codes into SAS

Here is the SAS code for the R code you provided:

```
SAS
data output_dataset;
set input_dataset;
LBORRES = ifc(RAW.LB.MODULE='LB3' & not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB3' & not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE='LB4' & not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB4' & not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE='LB6' & not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB6' & not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  RAW.LB.LBORRES)))));
run;
```

Note that you should replace "input_dataset" and "output_dataset" with the actual names of your input and output datasets, respectively. Also, SAS's equivalent to R's "ifelse" function is "ifc". The "not missing" function is used to check whether a variable is missing or not in SAS. The "vvalue" function is used to resolve the value of a macro variable within a data step. Finally, the "toupper" function is used to convert a string to upper case in SAS.

please note LBORRES is a character variable, it will not work, please rewrite the SAS code.

You're correct. Since "ifc" only works with numeric variables, we need to use a different approach to handle character variables in SAS. Here is the updated SAS code:

```
SAS
data output_dataset;
set input_dataset;
if RAW.LB.MODULE='LB3' and not missing(RAW.LB.LBVDIP) then LBORRES = upcase(vvalue(RAW.LB.LBVDIP));
else if RAW.LB.MODULE='LB3' and not missing(RAW.LB.LBORRES) then LBORRES = RAW.LB.LBORRES;
else if RAW.LB.MODULE='LB4' and not missing(RAW.LB.LBVDIP) then LBORRES = upcase(vvalue(RAW.LB.LBVDIP));
else if RAW.LB.MODULE='LB4' and not missing(RAW.LB.LBORRES) then LBORRES = RAW.LB.LBORRES;
else if RAW.LB.MODULE='LB6' and not missing(RAW.LB.LBVDIP) then LBORRES = upcase(vvalue(RAW.LB.LBVDIP));
else if RAW.LB.MODULE='LB6' and not missing(RAW.LB.LBORRES) then LBORRES = RAW.LB.LBORRES;
else LBORRES = RAW.LB.LBORRES;
run;
```

This code uses the "if-then-else" statement to handle character variables. The "upcase" function is used to convert a string to upper case in SAS. The "vvalue" function is used to resolve the value of a macro variable within a data step. Finally, the "not missing" function is used to check whether a variable is missing or not in SAS.

can you use ifc function?

Yes, you can use the "ifc" function in SAS to handle character variables in a more concise way. The "ifc" function is similar to "if", but it works with character variables instead of numeric variables. Here is the SAS code that uses the "ifc" function:

```
SAS
data output_dataset;
set input_dataset;
LBORRES = ifc(RAW.LB.MODULE='LB3' and not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB3' and not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE='LB4' and not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB4' and not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE='LB6' and not missing(RAW.LB.LBVDIP), upcase(vvalue(RAW.LB.LBVDIP)),
  ifc(RAW.LB.MODULE='LB6' and not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  RAW.LB.LBORRES)))));
run;
```

The "ifc" function works by evaluating a series of conditions and returning the corresponding value for the first condition that evaluates to true. If none of the conditions are true, the function returns the default value (in this case, "RAW.LB.LBORRES"). The "upcase" function is used to convert a string to upper case in SAS, and the "vvalue" function is used to resolve the value of a macro variable within a data step. Finally, the "not missing" function is used to check whether a variable is missing or not in SAS.

OTHER TASKS

In addition to its great benefits for the SDTM mapping process, the NLP solution can also be applied to other tasks such as ADaM, TLG, and CSR generation, provided that standard algorithm libraries and knowledge databases are available to support. Based on our observations of the progress in our industry's work, we would like to share our thoughts.

- ADaM: This task involves long, complex sentences and can be challenging for non-standard specifications. Standard macros/functions and knowledge databases are required for support.

2. Automate Scientific Document such as CSR Generation using ML/NLP.
https://www.lexjansen.com/phuse-us/2021/ml/PRE_ML01.pdf
3. ChatGPT: Optimizing Language Models for Dialogue.
<https://openai.com/blog/chatgpt>
4. Metadata-based Auto-Programming Process.
<https://www.lexjansen.com/phuse-us/2018/si/SI10.pdf>
5. Metadata-based Auto-Programming Process – Part 2: Map human language to SAS® code by Natural Language Processing (NLP).
<https://www.lexjansen.com/phuse-us/2019/ml/ML05.pdf>
6. IDAR virtual AI assistant to automate statistical report generation.
https://www.lexjansen.com/phuse-us/2020/ml/ML01_ppt.pdf

ACKNOWLEDGMENTS

We would like to express our gratitude to Meng Chen for her generous support in sponsoring this project. Special thanks to Kevin Huang, Xinya Wang, Zhanjian Dong, Kang Zhao, Fan Zhang, and Zhilong Qi for their significant contributions to designing, testing and providing feedback on the SDTM mapping process. We would also like to acknowledge the valuable contributions of our data scientist interns, Wenkai Xiang, Jessica Chen, and Hengjie Cui, for their exceptional programming and development of the R Shiny app.

CONTACT INFORMATION

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

Hao Chen, Yiwen Wang
AstraZeneca R&D China Biometrics
88 Xizang North Road, Jing'an District
Shanghai, China, 201210
Email: hao.chen15@astrazeneca.com
Email: yiwen.wang1@astrazeneca.com

Jaimin Patel
AstraZeneca R&D US Biometrics
Wilmington, DE 19803 United States of America
Email: jaimin.patel2@astrazeneca.com

Brand and product names are trademarks of their respective companies.