

Balance between Human and Machine

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

Hao Chen
Jaimin Patel
Yiwen Wang



Agenda

Background

Business Problem

Case Sharing

Discussion

Summary



Background



TRANSITIONING TO FUTURE OF CLINICAL TRIALS

Study published by the [Massachusetts Institute of Technology \(MIT\)](#) has found that only 13.8% of drugs successfully pass clinical trials company can expect to pay between \$161 million to \$2 billion for any drug to complete the entire clinical trials. This Artificial Intelligence has the potential to Significantly reduce clinical cycle time and cost while helping to improve the outcome of clinical development.

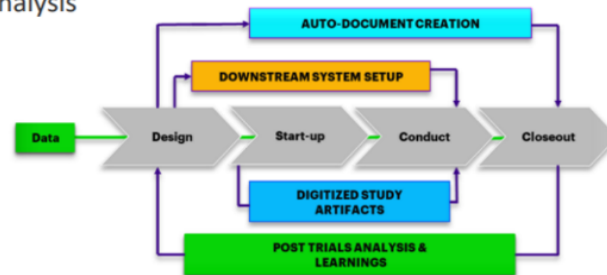
Traditional clinical trail data flow

A Non-linear maze of manual effort, rework
Inefficiency



The future state vision

Streamline interoperability of trial artifacts enabled through the standardization of data/vocabulary learnings from AI & ML analysis



FUTURE STATE BENEFITS



Reduced manual
Effort & rework.



Increased
interoperability
Across Multiple
systems



Improved Trial
Cycle time.



Optimized Use of Insights
To Improve the Next Trial



Reduced Errors in
Trial artifacts.

From Unstructured Data to Structured Data – AI Solution

NLP(Natural Language Processing)

The screenshot displays a JSON object representing NLP data, a dependency parse tree, and a corresponding sentence. The JSON object includes fields like "label", "lemma", "text", "content", "beginOffset", "dependencyEdge", "headTokenIndex", and "label". The dependency parse tree shows the relationship between tokens in the sentence "ADTM is equal to the date and time portion of VS. VSDTC". The tokens are labeled with parts of speech (POS) and lemmas. The sentence is: "(ADTM)₁₂ is equal to the date and time (portion)₁ of (VS.VSDTC)₁₄. Converted to numeric (SAS)₁₀ (datetime)₅. (ADY)₆-(ADT)₈-(ARFSTD)₇ plus 1 if (ADT)₈ is greater than or equal to (ARFSTD)₇; otherwise (ADY)₆-(ADT-ARFSTD)₇. If (EGSPID)₁₁ does not contain (UNCHED)₁₃, then equal to (EG)₁₅(VISIT)₃, otherwise add (UNCHEDULED)₉ in the (end)₁₂ of (EG.VISIT)₃. If (EG.VISIT)₃ contains (UNCHEDULED)₉, then (AVISIT)₁₄ will end with (UNCHEDULED)₉.

5

<https://cloud.google.com/natural-language/>

Unstructured Data

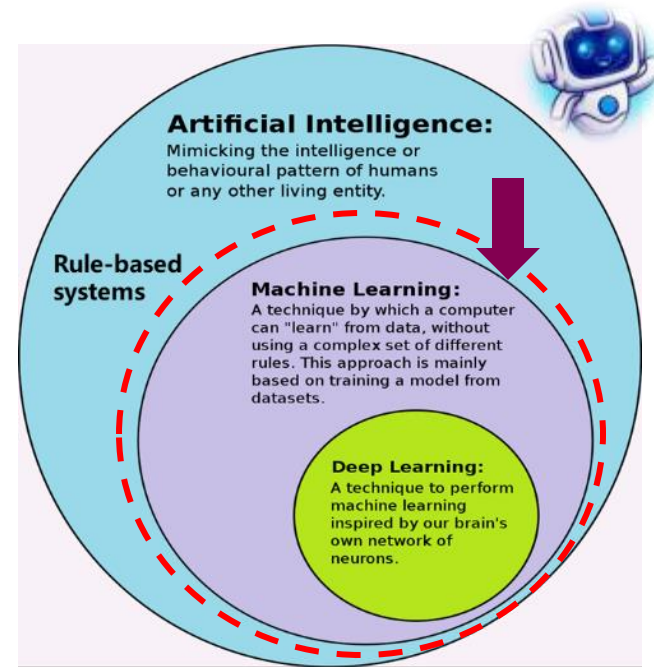
The university has 5600 students. Shaun (ID Number: 160801), 18 years old Communication study. Linh with ID number 160802, majoring in Accounting and is 20 years old. Ahmed from Psychology study program, 19 years old, ID number 160803.

Semi-Structured Data

```
<University>
<ID Number="160801">
  <Name="Shaun">
    <Age="18">
      <Program="Communication">
        <ID Number="160802">
          <Name="Linh">
            <Age="20">
              <Program="Accounting">
                .....
      </University>
```

Structured Data

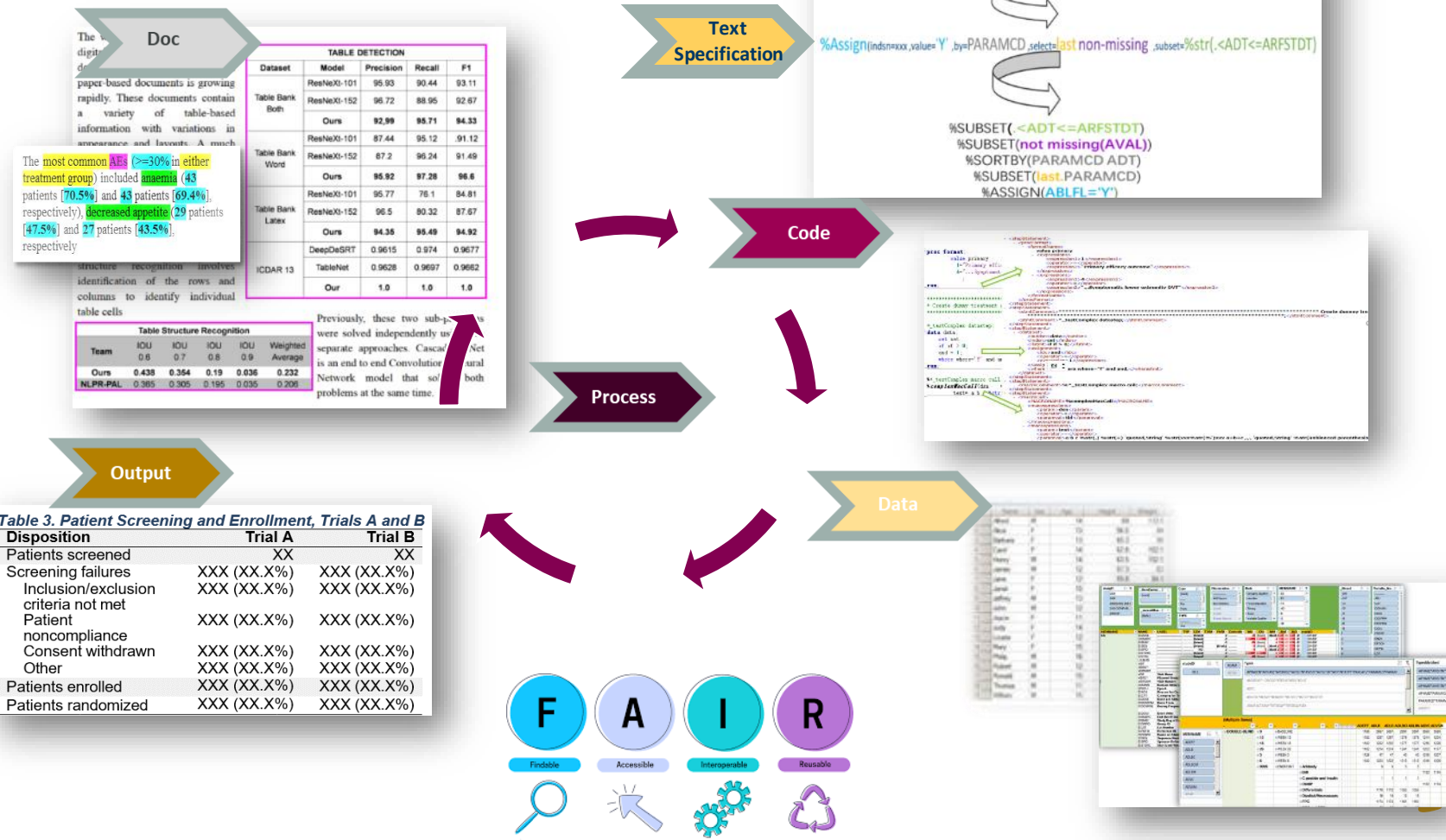
ID	Name	Age	Program
160801	Shaun	18	Communication
160802	Linh	20	Accounting
160803	Ahmed	19	Psychology



How deep learning is a subset of machine learning and how machine learning is a subset of artificial intelligence (AI) https://en.wikipedia.org/wiki/Deep_learning



Digitalization for the Trial Artifacts





Business Problem



Business Problem

- Clinical data analysis algorithms are complex and require pre-planning in human-readable text
- People cannot always follow strict standard algorithm descriptions
- GUI(Graphical User Interface) is not user-friendly to build flexible algorithms
- Algorithm in 'free' text are commonly used in the industry
- Diverse Mapping Specification across TA/Companies
- Translation of human-readable text to machine-readable code is time-consuming
- Ongoing updates are required due to changes in source data and debugging process
- Synchronizing Specifications and code is challenging for human

Define-XML Version 2 [Variable Level Metadata]

Demographics (DM) [Location: [dm.xpt](#)]

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
RFXSTDC	Date/Time of First Study Treatment		datetime		ISO8601	Derived	RFXSTDC = 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
RFICDTC	Date/Time of Informed Consent		date		ISO8601	CRF Page 1	
RFPSTDC	Date/Time of End of Participation		date		ISO8601	Derived	RFPSTDC = Date/time when subject ended participant or follow-up in a trial (last date/time of contact)
DTHDTC	Date/Time of Death		date		ISO8601	CRF Page 907	From AE page; when outcome is Death then Y then date/time, else Null
DTHFL	Subject Death Flag		text	1	["Y" = "Yes"] <No Yes Response Subset (Y)>	CRF Page 907	From AE page; when outcome is Death then Y, else Null



SDTM Mapping Code Generation

SDTM specs and programming codes consistency

- Updates should be taken in both files

SDTM programming codes not fully reusable in most cases

- Reference programming code needs to be checked line by line before implementation

Manual creation not the best solution

- Time consuming
- Pron to error
- Intensive QC required for simple and standardized variable generation



Standard Derivations in clinical laboratory data

- Laboratory toxicity grading has been very important part of safety reporting since the FDA began accepting electronic data. Remaining up to date with the various CTCAE versions could be a challenge when you have different source laboratory raw data each time.

Grade 1	Grade 2	Grade 3	Grade 4
>ULN - 2.5 x ULN if baseline was normal; 2.0 - 2.5 x baseline if baseline was abnormal	>2.5 - 5.0 x ULN if baseline was normal; >2.5 - 5.0 x baseline if baseline was abnormal	>5.0 - 20.0 x ULN if baseline was normal; >5.0 - 20.0 x baseline if baseline was abnormal	>20.0 x ULN if baseline was normal; >20.0 x baseline if baseline was abnormal



if LLN <= BASE (LBSTRESN when LBBLF="Y") <= ULN then derive LBTOXGRN as per following.

if ULN < LB.LBORRES <= ULN*2.5 then LBTOXGRN = 1.

else if ULN*2.5 < LB.LBORRES <= ULN*5 then LBTOXGRN = 2.

else if ULN*5 < LB.LBORRES <= ULN*20 then LBTOXGRN = 3.

else if LB.LBORRES > ULN*20 then LBTOXGRN = 4.

or else if BASE < LLN or BASE > ULN then derive LBTOXGRN as per following.

if BASE*2 < LB.LBORRES <= BASE*2.5 then LBTOXGRN = 1.

else if BASE*2.5 < LB.LBORRES <= BASE*5 then LBTOXGRN = 2.

else if BASE*5 < LB.LBORRES <= BASE*20 then LBTOXGRN = 3.

else if LB.LBORRES > BASE*20 then LBTOXGRN = 4.



```

*Calculate toxicity grade ;
proc fcomp outlib=work.funcs.toxgrd ;
    function GradeTox(lbcd $, lln, uln, bsln, value,dir $);
        nml=(missing(lln)=0);
        *Non-missing normal range lower limit;
        nmu=(missing(uln)=0);
        *Non-missing normal range upper limit;
        /*-----*/
        /* Tox Grades
        /*-----*/

    *Low direction;
    if dir='L' then do;
        if lbcd='NEUT' then
            grade=4*( . < value < 0.5)
            +3*(0.5 <= value < 1.0)
            +2*(1.0 <= value < 1.5)
            +1*(1.5 <= value < llm);
        else if lbcd='ALB' then
            grade=3*( . < value < 20)
            +2*(20 <= value < 30)
            +1*(30 <= value < llm);
        else if lbcd='COCA' then
            grade=4*( . < value < 1.5)
            +3*((1.5 <= value < 1.75) )
            +2*((1.75 <= value < 2.0) )
            +1*((2. <= value < llm) );
        return (grade);
    end;

    *High direction;
    else if dir='H' then do;
        if lbcd='HGB' then
            grade=3*((uln+40<value) or (bsln>uln and bsln+40<value))
            +2*((uln+20<value<=uln+40) or (bsln>uln and value>bsln))
            +1*((uln<value<=uln+20) or (bsln>uln and bsln<=value<=bsln+20));
        else if lbcd='INR' then
            grade=3*( (value>2.5*uln))
            +2*((1.5*uln<value<=2.5*uln))
            +1*((1*uln<value<=1.5*uln));
    end;
end;

```



Standard algorithm example from CDISC 360



The diagram shows a curved arrow pointing from the 'Possible SAS Code' column of the table to the 'Expression Builder' window, indicating the process of translating the SAS code into a GUI-based expression.

ID	TYPE	DESC	KEYWORD	Parameter	Definition of KEYWORD	Possible SAS Code
ALL.USUBJID		Concatenation of STUDYID and SUBID	Concatenate	dot/STUDYID/SUBID	when the system identifies the keyword is Concatenate, it will look into parameter field Observation_Number (I)	USUBJID = catx('-', STUDYID, SUBID);
V5.VSPID		Current observation Number (N)	Observation_Number		when the system identifies the keyword is Observation_Number, it will convert the ISODTC (source date):	VSPID = put(n, best.);
V5.VSDTC		Convert assessment date (VISDAT/VSDAT) to ISO8601 date format. Calculation: VSDTC = RFSSTDC+1 if VSDTC is on or after RFSSTDC, VSDTC - RFSSTDC if VSDTC precedes RFSSTDC	ISODTC	VISDAT/VSDAT	when the system identifies the keyword is ISODTC, it will look into parameters field for Relative_Day (assessment Date/Reference Date):	VSDTC = put(VSDAT, e8601da.);
V5.VSDY	Computation		Relative_Day	VSDTC/RFSSTDC	when the system identifies the keyword is Standard_Height_Conversion (original value/original unit/standard unit):	VSDY = input(VSDTC, e8601da.);
V5.VSSTRESN.item1	Computation	Convert VSORRES to standard result.	Standard_Height_Conversion	VSORRES/VSORRESU/VSSTRESU	when the system identifies the keyword is Sequence (reference variable to start counting):	VSDY = input(VSDTC, e8601da.);
V5.VSSEQ	Computation	Sequential number identifying records within each USUBJID.	Sequence	USUBJID	When the system identifies the keyword is EPOCH (Dataset/MergingKey/Date):	VSDY = input(VSDTC, e8601da.);
V5.SE.SE_EPOCH		Assign EPOCH based on the SE domain	EPOCH	V52/USUBJID/VSDTC	When the system identifies the keyword is BASEFL (Dataset/Sorting/Test):	VSDY = input(VSDTC, e8601da.);
V5.V5.V5BASEFL		Derive Baseline flag	BASEFL	V52/USUBJID, VSPOS, V5TESTCD, VSDTC/V5TESTCD	when the system identifies the keyword is BASEFL, it will look into parameter field for 3 ISODTC (source date):	VSDY = input(VSDTC, e8601da.);
V6.DMDATC		Convert to ISO8601 date	ISODTC	DMDAT	when the system identifies the keyword is ISODTC, it will look into parameters field for ISODTC (source date):	VSDY = input(VSDTC, e8601da.);
V6.RFSDTC		Convert to ISO8601 date	ISODTC	DSSTDAT	when the system identifies the keyword is ISODTC, it will look into parameters field for ISODTC (source date):	VSDY = input(VSDTC, e8601da.);
V6.RPKSDTC		Convert to ISO8601 date	ISODTC	DSSTDAT	when the system identifies the keyword is ISODTC, it will look into parameters field for ISODTC (source date):	VSDY = input(VSDTC, e8601da.);

GUI(Graphical User Interface) example for algorithm build

Expression Builder - [CustomExpression]

ROUND(Pi() * POWER(Radius, 2) + Decimal)

+ - * / % = < > AND OR NOT

Functions

fx Text
fx Aggregate
fx Date-Time
fx Logical
fx Math
fx Other

Operators
Constants
Fields

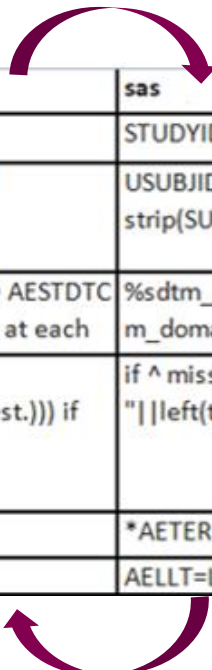
Sum
Average
Min
Max
Count
First
Last
Now
Today
AddDays
AddHours
AddMilliseconds
AddMinutes

Accurate algorithm generated, However

- Not user-friendly
- Hard to apply for projects across different therapeutic area/company
- Not adopted in daily work



Can we translate 'free' text specification to code, and vice visa?



The diagram illustrates a cycle between 'free' text specification and code. A curved arrow at the top points from the 'Conversion Definition' column to the 'sas' column, representing the translation from free text to code. A curved arrow at the bottom points from the 'sas' column back to the 'Conversion Definition' column, representing the translation from code back to free text.

Variable	Conversion Definition	sas
STUDYID	"00008"	STUDYID="00008"
USUBJID	Concatenate STUDY and SUBJECT separating by '/'. (e.g., 00008/001).	USUBJID=cat(strip(STUDY), '/', strip(SUBJECT))
AESEQ	Sort by variable STUDYID USUBJID AESTDTC AEDECOD. Restart the numbering at each	%sdm_seq(indata=&syslast, domain=&sdtm_domain.);
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;
AETERM	Set to RAW.AE.AETERM.	*AETERM=AETERM; *copy/keep;
AELLT	Set to RAW.AE.LLT_NAME	AELLT=LLT_NAME





Case Sharing

SDTM Mapping Text Spec to Code Translation

- SAS Code Generation
- Algorithm Text Generation
- R Code Generation



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

```

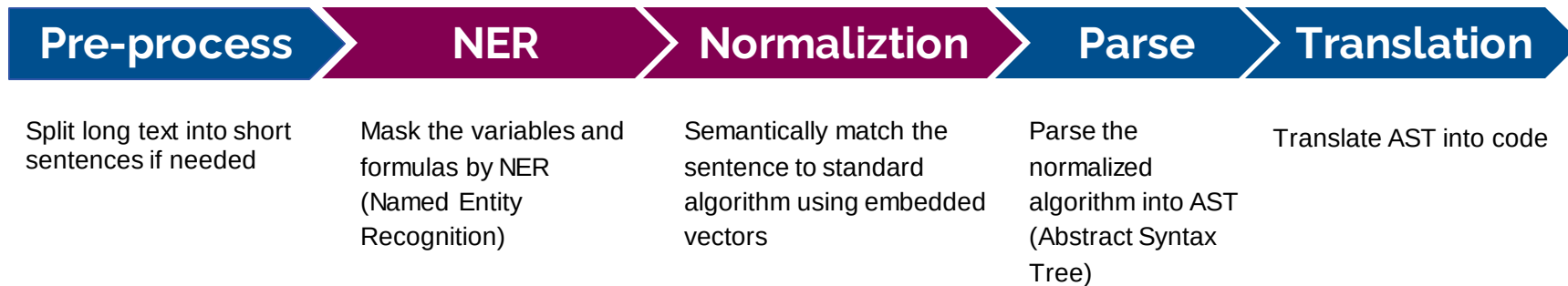
if RAW.LB.MODULE='LB3' and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)), else if
RAW.LB.MODULE='LB3' and RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES, else if RAW.LB.MODULE='LB4'
and RAW.LB.LBVDIP not missing then set to upcase(vvalue(RAW.LB.LBVDIP)), else if RAW.LB.MODULE='LB4'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

```

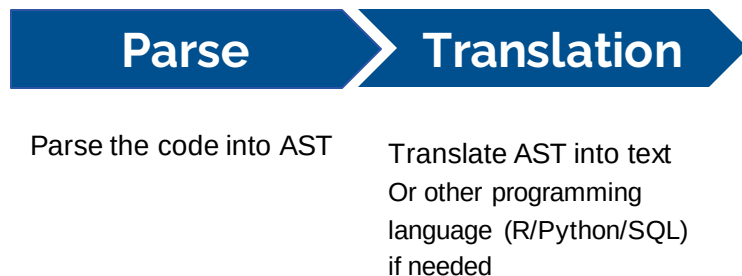
[illegible][illegible]

Process

Text to Code



Code to Text



- Machine Learning
- Rule-based



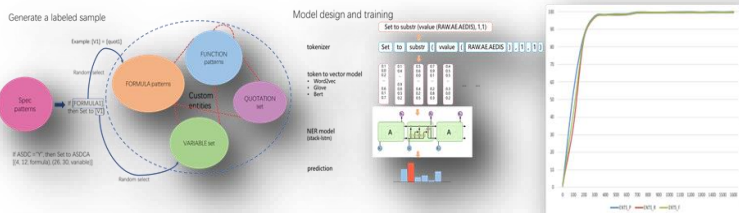
Process

NER

Use NER model to recognize variables, formulas, functions and quotes. Then mask with special strings

Normalization

Semantically match sentences with standard algorithms based on the embedding vectors



"Lion is the king of the jungle."

"The tiger hunts in this forest."

"Everybody loves New York."

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	BASILINE FLAG	DERIVE BASELINE 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 \$\$\$.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). Note: 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

Variable Name	Variable Label	Type	Length	Conversion 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='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						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						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='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						For RAW.CLAB records: Set to upcase of RAW.CLAB.LBORRES.

Solution

Spec (Human Language + SAS code piece)

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='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='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.

4. Parse

```
?value: numeric  
equation1  
equation2  
equation3  
equationdte  
equationdttim  
concat  
concatnote  
set  
setnote  
calculate  
sort  
convertformat  
where  
when  
setto  
equationelif
```

```
equation1: name "=" if conditions then action2 else action2  
equation2: name "=" variable if conditions else action2  
equation3: name "=" (equation3sub)+  
equation3sub: (variable|assignasformat) if conditions  
equationelif: name "=" if conditions then action2 (equationelifsub)* (equationelisesub)*  
equationelifsub: ? if condition then action2
```

```
equationelif  
name  
if  
conditions  
conditionand  
conditionvar  
variable  
LB3  
condition  
missing  
variable  
then  
action2  
string  
equationelifsub  
else  
if  
condition  
string  
RAW.LB.LBORRES not missing  
then  
action2  
string  
RAW.LB.LBORRES  
equationelifsub  
else  
if  
condition  
string  
not missing  
then  
action2  
string  
equationelifsub  
else  
if  
condition  
string  
RAW.LB.LBORRES not missing  
then  
action2  
string  
equationelifsub  
else  
if  
condition  
string  
RAW.LB.LBORRES  
.....
```

AST (Abstract Syntax Tree)

```
/* For RAW.LB records: If RAW.LB.MODULE='LB3' and RAW.LB.LBVDIP not mi  
RAW.LB.LBORRES, else if RAW.LB.MODULE='LB4' and RAW.LB.LBVDIP not missi  
RAW.LB.LBORRES, else if RAW.LB.MODULE='LB6' and RAW.LB.LBVDIP not missi  
RAW.LB.LBORRES, else set to RAW.LB.LBORRES. */  
LBORRES= ifc(MODULE='LB3' and ^missing(LBVDIP), upcase(vvalue(LBVDIP)),  
ifc(MODULE='LB3' and ^missing(LBORRES), LBORRES,  
ifc(MODULE='LB4' and ^missing(LBVDIP), upcase(vvalue(LBVDIP)),  
ifc(MODULE='LB4' and ^missing(LBORRES), LBORRES,  
ifc(MODULE='LB6' and ^missing(LBVDIP), upcase(vvalue(LBVDIP)),  
ifc(MODULE='LB6' and ^missing(LBORRES), LBORRES,  
LBORRES ))))));
```

SAS programs (or R/Python/SQL)

5. Translation



Solution

- Achieving 100% automation can be a challenging goal
- Involve humans in the review and decision-making process for new scenarios (automatically flagged)
- Additional patterns can be added to the knowledge base as needed to improve the process



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
AESQ	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	Assign RAW.AE.AECAT as format if RAW.AE.AECATOC=1	if AECATOC=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



Mapping Code Auto Generation

– Focusing on Type I & II

• Derivation Types

• 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 _aewd=put( aedis, $NY.);
209 _aendtc=%dtc(INVAR_DAT= ae.aeendat , INVAR_TIM= ae.aentim , OUTVAR_DTC= ae._aendtc );
210 _aeout=upcase(putc( aeout,vformat(aeout)));
211 _aerel=put( aerel, $AERELF.);
212 _aeser=upcase(putc( aeser,vformat(aeser)));
213 _aesev=upcase(putc( aesevmax,vformat(aesevmax)));
214 _aestdtc=%dtc(INVAR_DAT= ae.aestdat , INVAR_TIM= ae.aesttim , OUTVAR_DTC= ae._aestdtc );
215 _aeterm=aeterm;
216 _aehlgtcd=hltcode;
217 _aehlgt=hltname;
218 _aehltd=hlt_code;
219 _aehl=hlt_name;
220 _ip=ip;
221 run;

222
223
224 *****
225 * Execute standalone macros
226 *****
227 %dp(INVAR_REFDTC=&_templib._dm.rfstdtc , INVAR_DTC=&_templib._ae.aendtc , OUTVAR_DY=&_templib._ae.aeendy ); ;
228 %dp(INVAR_REFDTC=&_templib._dm.rfstdtc , INVAR_DTC=&_templib._ae.aestdtc , OUTVAR_DY=&_templib._ae.aesty ); ;
229 %seq(DATASET=LAYOUT=STUDYID USUBJID AESTDTC AEDECOD, OUTVAR_SEQ=&_templib._ae.aeseq); ;

4 ****for RAW.PREC records:****if RAW.PREC.MODULE='PRE...****;
5 /* variable LBSPEC [code] */
6 VISTNUM=visit;
7 %do i=1 to (INVAR_DAT=amp dal, INVAR_TIM=, OUTVAR_DTC=andm domain.dtc );
8 if vvalue(asm_auv)="" then LBSPEC="NOT DONE";
9 if upcase(module)="PREC" then
10 do;
11 LBSPEC=upcase(vvalue(PREGTSTD));
12 LBSPEC="URINE";
13 end;
14 if upcase(module)="PREG1" then
15 do;
16 LBSPEC=upcase(vvalue(PREGTSTD));
17 LBSPEC="SERUM";
18 end;
19 LBSPEC="HCG";
20 LBSPEC="Chorio gonadotropin Beta";
21 LBSPEC="PREGNANCY TEST";
22 /* LBSPEC= */
23 /* variable LBSPEC [code] nan nan nan
24 run;
```



Code Generated

```

* Program name ..... : AE.as
* Type ..... : SDTM*
* Purpose ..... : Creation of sdtm.AE and sdtm.supplAE
* Author ..... : XXXXX
* Datetime created ..... : 07/13/2023, 12:53:15.*
* Input datasets ..... : [NOTE: Complete as needed from the mapping specification]....
* Extract Rule ..... : [where RAW.AE.AEANY="1"]
* Macros used ..... : %setup, %localsetup*
* Output files ..... : sdtm.AE, sdtm.supplAE*
* Revision history ..... :
*
* Date ..... Author ..... Ref ..... Revision
*
*****
** Setup calls .....
%autoexec;

%let sdtm_domain = AE;

%let AESort_Order = STUDYID USUBJID AESTDTC AEDECOD;

%let kname = STUDYID DOMAIN USUBJID AESEQ AESPID AETERM AELLT AELLTCD AEDECOD AEPTCD AEHLT AEHLTCD
AESDTH AESHOSP AESLIFE AESMIE AECONTRT EPOCH AESTDTC AEENDTC AESTDY AEENDY;

%preprocess (domain=AE);

***** RAW AE *****
data RAW_AE;

  attrib
    STUDYID label='Study Identifier'.....length=$21
    ...
    AEREL label='Causality'.....length=$200
    AEOUT label='Outcome of Adverse Event'.....length=$200

set
  RAW.AE (rename=(AESER = _AESER ). where=(AEANY="1"));

  /* 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;

  /* Set to RAW.AE.SOC_CODE. */
  AESOCCD = SOC_CODE;

  /* Assign RAW.AE.AESEVMAX as format. */
  AESEV=
    ifc(AESEVMAX = "1", "MILD",
    ifc(AESEVMAX = "2", "MODERATE",
    ifc(AESEVMAX = "3", "SEVERE",
    AESEVMAX))) ;

  /* Assign RAW.AE.AESCD as format. */

```

```

3311  data temp;
3312  merge RAW_AE(in=a) RAW_SERAE;
3313  by SUBJECT AENO;
3314  if a
3315  ;
3316  run;
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```

Data Join/Combine

Grouping process

Todo placeholder

Variable level

Data level



~/ShinyApps/online_sdtm_tool0707/sdtm_tool_s1/sdtm_tool_0816 - Shiny - Google Chrome

rstudio.scp.astrazeneca.net/s/aef2cc391f1230bee297a/?view=shiny_foreground

https://rstudio.scp.astrazeneca.net/s/aef2cc391f1230bee297a/p/48ecb705/ Open in Browser Publish

Mapping Spec File Code Generated Generation Process

Upload Mapping Spec File

Browse... No file selected

Name of Sheet (Case-Sensitive)

☐ use template SAS code?

Integrated Process DEMO



Text to Code live demo

Text to Code Demo

Enter some text:

```
grade =if >ULN - 2.5 x ULN    when baseline    was normal or 2.0 -2.5 x baseline when    baseline
was abnormal then set to 'Grade 1'; if    >2.5 -    5.0 x ULN and baseline was normal or >2.5 - 5.0 x
baseline and baseline was abnormal    then    set to 'Grade 2'; if >5.0 - 20.0 x ULN and baseline was
normal or >5.0 -20.0 x baseline    and baseline was abnormal    then    set to 'Grade 3';
if >20.0 x ULN and baseline was normal or >20.0 x baseline and baseline was abnormal    then    set
to 'Grade 4'; else set to ''
```

Parse Text

NER

NER2Func

Normalize

Text to Code Demo

Enter some text:

```
grade =if >ULN - 2.5 x ULN    when baseline    was normal or 2.0 -2.5 x baseline when    baseline
was abnormal then set to 'Grade 1'; if    >2.5 -    5.0 x ULN and baseline was normal or >2.5 - 5.0 x
baseline and baseline was abnormal    then    set to 'Grade 2'; if >5.0 - 20.0 x ULN and baseline was
normal or >5.0 -20.0 x baseline    and baseline was abnormal    then    set to 'Grade 3';
if >20.0 x ULN and baseline was normal or >20.0 x baseline and baseline was abnormal    then    set
to 'Grade 4'; else set to ''
```

Parse Text

NER

NER2Func

Normalize

Normalized:

```
grade =if range(>ULN , 2.5 *ULN) and
isnormal(baseline) or range(2.0 *baseline , 2.5
*baseline) and isabnormal(baseline) then set to
'Grade 1'; if range(>2.5 *ULN , 5.0 *ULN) and
isnormal(baseline) or range(>2.5 *baseline , 5.0
*baseline) and isabnormal(baseline) then set to
'Grade 2'; if range(>5.0 *ULN , 20.0 *ULN) and
isnormal(baseline) or range(>5.0 *baseline ,
20.0 *baseline) and isabnormal(baseline) then set
to 'Grade 3'; if range(>20.0 *ULN) and
isnormal(baseline) or range(>20.0 *baseline) and
isabnormal(baseline) then set to 'Grade 4'; else
set to ''
```

SAS code:

```
grade= ifc(range(>ULN , 2.5 *ULN) and
isnormal(baseline) or range(2.0 *baseline , 2.5
*baseline) and isabnormal(baseline), 'Grade 1',
ifc(range(>2.5 *ULN , 5.0 *ULN) and
isnormal(baseline) or range(>2.5 *baseline , 5.0
*baseline) and isabnormal(baseline), 'Grade 2',
ifc(range(>5.0 *ULN , 20.0 *ULN) and
isnormal(baseline) or range(>5.0 *baseline , 20.0
*baseline) and isabnormal(baseline), 'Grade 3',
ifc(range(>20.0 *ULN) and
isnormal(baseline) or range(>20.0 *baseline) and
isabnormal(baseline), 'Grade 4',
''));
```

Parsed Result:

```
equation1if
name grade
if if
conditionsonr
conditionsand
condition
string range(>ULN , 2.5 *ULN)
condition
string isnormal(baseline)
conditionsand
condition
string range(2.0 *baseline , 2
condition
string isabnormal(baseline)
then then
action2
string set to 'Grade 1'
equation1ifsub
elseif
if if
conditionsonr
conditionsand
condition
string range(>2.5 *ULN
condition
string isnormal(baseli
conditionsand
condition
string range(>2.5 *bas
condition
string isabnormal(base
then then
action2
string 'Grade 2';
equation1ifsub
elseif
if if
conditionsonr
conditionsand
condition
string range(>5.0 *UL
condition
string isnormal(baseli
conditionsand
condition
string range(>5.0 *bas
condition
string isabnormal(base
then then
```



Code to Algorithm Text

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

Text

```
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(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 Algorithm Text

Convert

Text

```
if RAW.LB.MODULE='LB3' and not missing(RAW.LB.LBVDIP) then set to upcase(vvalue(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(vvalue(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(vvalue(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
            /
          )
    )
  )
```



Code to Other Programming Language(e.g., R)

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

Abstract syntax tree

```
start  
None
```





Discussion

- ChatGPT solution testing
- Other Tasks AI/NLP may help





ChatGPT solution testing

write SAS codes for the algorithms below:
LBORRES =
if RAW.LB.MODULE="L83" and RAW.LB.LVDIP not missing then set to
uppercase(value(RAW.LB.LVDIP)); else if RAW.LB.MODULE="L83" and RAW.LB.LBORRES not
missing then set to RAW.LB.LBORRES; else if RAW.LB.MODULE="L84" and RAW.LB.LVDIP not
missing then set to uppercase(value(RAW.LB.LVDIP)); else if RAW.LB.MODULE="L84" and
RAW.LB.LBORRES not missing then set to RAW.LB.LBORRES; else if RAW.LB.MODULE="L86"
and RAW.LB.LVDIP not missing then set to uppercase(value(RAW.LB.LVDIP)); else if
RAW.LB.MODULE="L86" 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="L83" and RAW.LB.LVDIP not missing then uppercase(value(RAW.LB.LVDIP))
  else if RAW.LB.MODULE="L83" and RAW.LB.LBORRES not missing then RAW.LB.LBORRES
  else if RAW.LB.MODULE="L84" and RAW.LB.LVDIP not missing then uppercase(value(RAW.LB.LVDIP))
  else if RAW.LB.MODULE="L84" and RAW.LB.LBORRES not missing then RAW.LB.LBORRES
  else if RAW.LB.MODULE="L86" and RAW.LB.LVDIP not missing then uppercase(value(RAW.LB.LVDIP))
  else if RAW.LB.MODULE="L86" 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 == "L83" & !is.na(RAW.LB.LVDIP),
    toupper(RAW.LB.LVDIP),
    ifelse(RAW.LB.MODULE == "L83" & !is.na(RAW.LB.LBORRES),
      RAW.LB.LBORRES,
      ifelse(RAW.LB.MODULE == "L84" & !is.na(RAW.LB.LVDIP),
        toupper(RAW.LB.LVDIP),
        ifelse(RAW.LB.MODULE == "L84" & !is.na(RAW.LB.LBORRES),
          RAW.LB.LBORRES,
          ifelse(RAW.LB.MODULE == "L86" & !is.na(RAW.LB.LVDIP),
            toupper(RAW.LB.LVDIP),
            ifelse(RAW.LB.MODULE == "L86" & !is.na(RAW.LB.LBORRES),
              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 = if(RAW.LB.MODULE="L83" & not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  if(RAW.LB.MODULE="L83" & not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  if(RAW.LB.MODULE="L84" & not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  if(RAW.LB.MODULE="L84" & not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  if(RAW.LB.MODULE="L86" & not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  if(RAW.LB.MODULE="L86" & 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 "value" 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="L83" and not missing(RAW.LB.LVDIP) then LBORRES = uppercase(value(RAW.LB.LVDIP));
else if RAW.LB.MODULE="L83" and not missing(RAW.LB.LBORRES) then LBORRES = RAW.LB.LBORRES;
else if RAW.LB.MODULE="L84" and not missing(RAW.LB.LVDIP) then LBORRES = uppercase(value(RAW.LB.LVDIP));
else if RAW.LB.MODULE="L84" and not missing(RAW.LB.LBORRES) then LBORRES = RAW.LB.LBORRES;
else if RAW.LB.MODULE="L86" and not missing(RAW.LB.LVDIP) then LBORRES = uppercase(value(RAW.LB.LVDIP));
else if RAW.LB.MODULE="L86" 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 "uppercase"
function is used to convert a string to upper case in SAS. The "value" 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 "ifn", 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="L83" and not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  ifc(RAW.LB.MODULE="L83" and not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE="L84" and not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  ifc(RAW.LB.MODULE="L84" and not missing(RAW.LB.LBORRES), RAW.LB.LBORRES,
  ifc(RAW.LB.MODULE="L86" and not missing(RAW.LB.LVDIP), uppercase(value(RAW.LB.LVDIP)),
  ifc(RAW.LB.MODULE="L86" 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 "uppercase"
function is used to convert a string to upper case in SAS, and the "value" 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.

- Versatile system excels in both text and code analysis & generation
- R code outperforms SAS code due to its access to more open-source training data
- Translating code into R before translating it into SAS can improve the quality
- Achieving accurate results may require multiple rounds of refinement and adjustment
- Challenging to control the quality of its results due to its black box nature
- Despite these limitations, ChatGPT can effectively process very complex text, surpassing the capabilities of many self-developed systems
- Important to consider the protection of company-restricted or confidential data when using ChatGPT



Other Tasks AI/NLP may help

ADaM

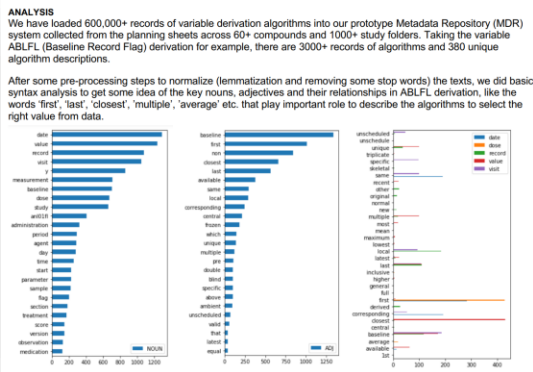
Long sentence and complex logic;
Pretty difficult for non-standard Specs;
Need more support from standard macro/functions and knowledge database;

TLG

Short and relatively Standard sentence;
Need support from standards macro/functions;
Need Table layout analysis from mockup table;

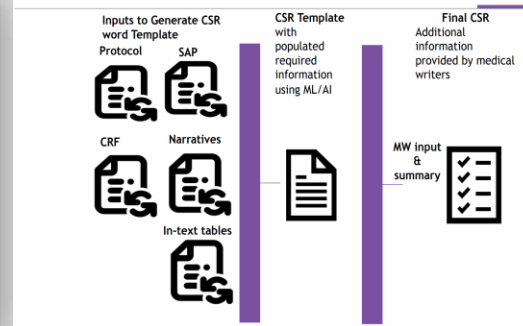
CSR

Good at key information extraction, similarity calculation, and text summary;
Predict and identify key information in CSRs and link to different documents and sections



TLG title NLU - title **auto** analyzed to **structured** data – map to code

Process



[Metadata-based Auto-Programming Process – Part 2: Map human language to SAS® code by Natural Language Processing \(NLP\)](#)

[JDAR virtual AI assistant to automate statistical report generation](#)

[Automate Scientific Document such as CSR Generation using ML/NLP](#)



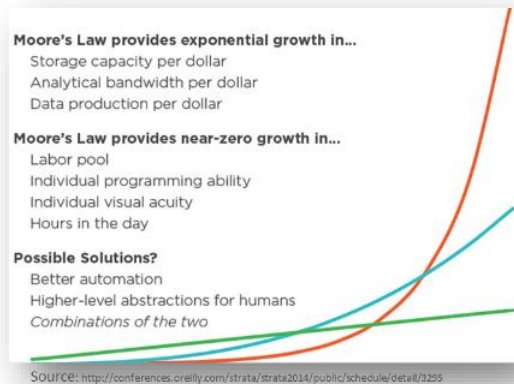


Summary



Summary

- Reduce the requirements on Spec, applicable for Spec across different TAs and companies
- Reduce manual synchronization work between Spec and Code
- Cultivate user's mindset in Standard & Automation
- Accumulate algorithm knowledgebase
- Framework applicable for other programming language (R/Python/SQL)
- Benefit for both upstream and downstream work(e.g., Mapping recommendation, Dependency analysis, Define.xml)
- AL/ML accelerate/simplify/automate the tasks for conversion between the unstructured data and structured data, based on which rule-based system can build automatic systems
- Combination of Machine and Human(AI+HI) revolutionize the way we work, learn and solve problems





Hao Chen

hao.chen15@astrazeneca.com

Yiwen Wang

yiwen.wang1@astrazeneca.com

AstraZeneca R&D China, Biometrics

Jaimin Patel

jaimin.patel2@astrazeneca.com

AstraZeneca US, Biometrics





Programming: Kevin Huang, Xinya Wang, Zhanjian Dong, Kang Zhao, Fan

Zhang, Zhilong Qi

Data Scientist Intern: Wenkai Xiang, Jessica Chen, Hengjie Cui

Thank
you

