

Metadata-based Auto-Programming Process

– Part 2: Map human language to SAS® code by Natural Language Processing (NLP)

PhUSE US Connect 2019
Paper ML05

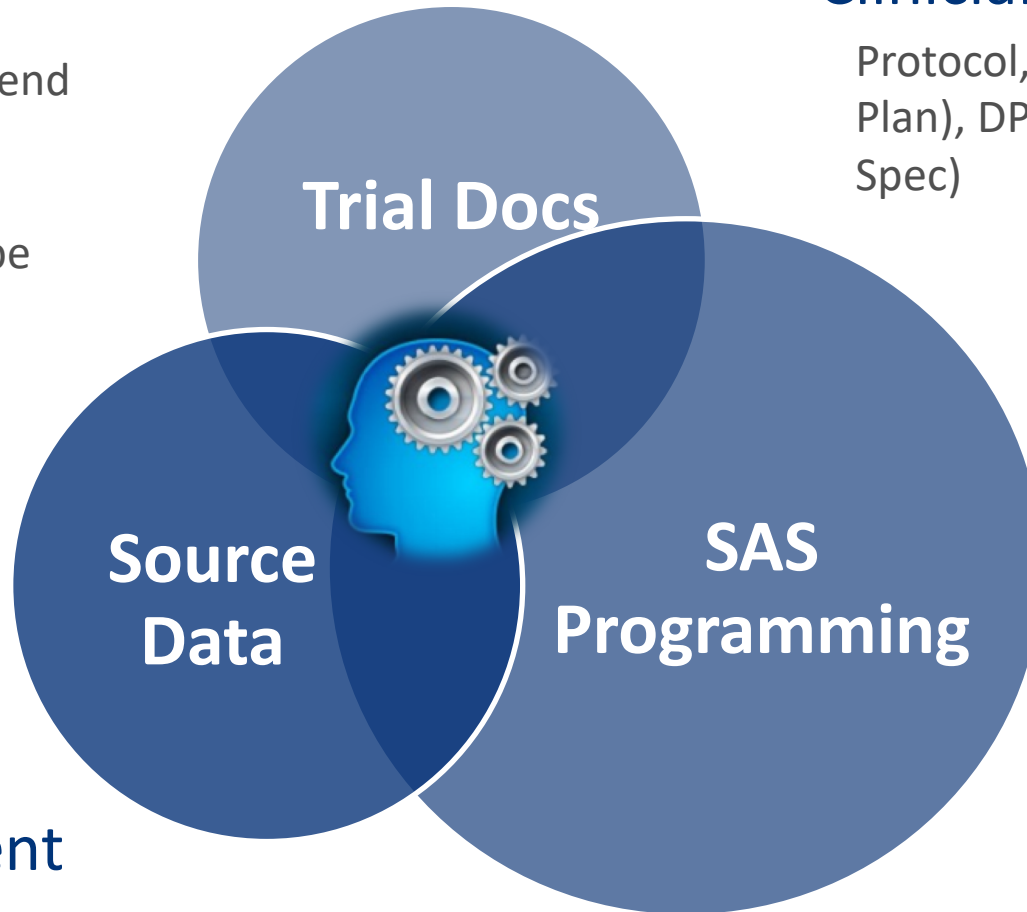
Sumesh Kalappurakal, Janssen R&D, New Jersey, US
Qian Zhao, Harry Chen, Janssen R&D, Shanghai, China

Agenda

- **Background**
- **Metadata-Driven Approach**
- **NLP**
 - ✓ **Why**
 - ✓ **Unsupervised approach**
 - ✓ **Supervised approach**
- **Conclusion**

CURRENT SITUATION - OUR WORK

Read/Comprehend
Compare
Translate
Copy/Paste/Type
...



Clinician, Statistician

Protocol, SAP(Standard Analysis Plan), DPS(Data Presentation Spec)

Programming

Macros
Open Codes
QC Codes
Analysis Data
Metadata excel file
(Specification file)
...

Data Management

Raw Data(SDTM)
CRF...

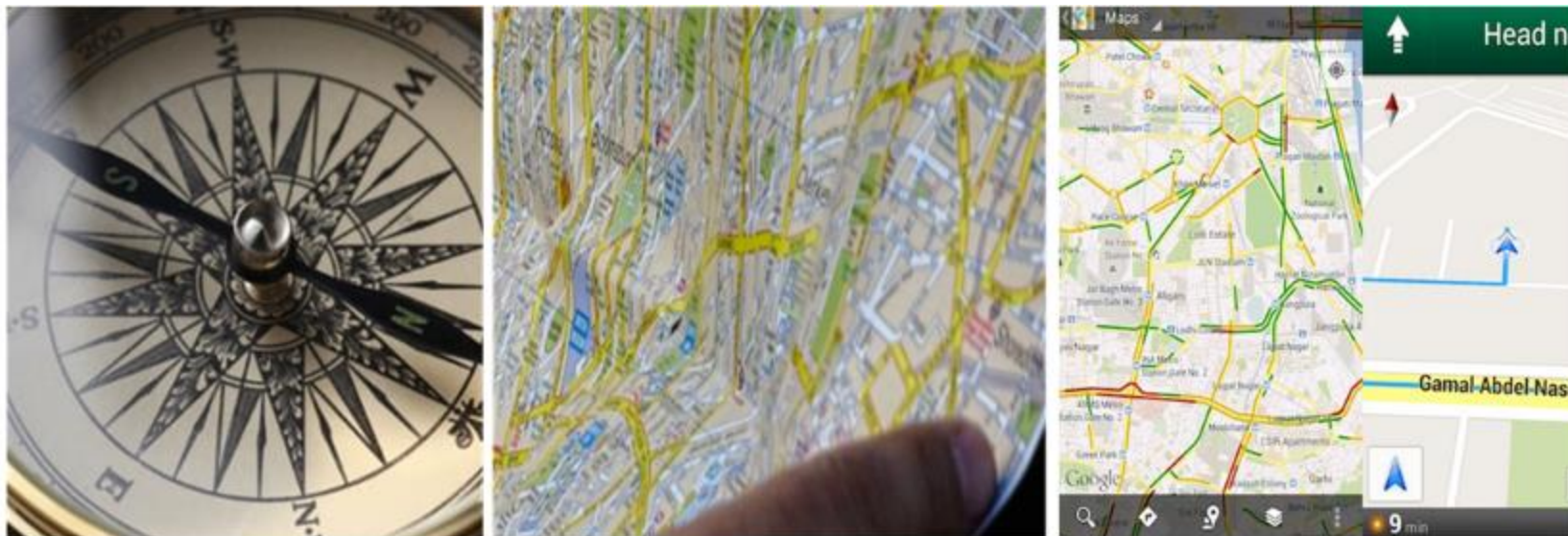
GENERAL PROCESS FOR DATA ANALYSIS

The screenshot displays the SAS Enterprise Guide interface. On the left, the 'Project Tree' shows a hierarchy of programs and data steps. The main window shows a complex data flow diagram with numerous steps connected by arrows. Overlaid on the bottom right is a large block of SAS code, which includes various data manipulation and statistical procedures. The code is partially obscured by large, semi-transparent text overlays: 'run;', 'end;', and 'by usubjid;'. At the bottom of the image, the Janssen logo is visible on the left, and the text 'PHARMACEUTICAL COMPANIES of Johnson & Johnson' is on the right. A small number '4' is in the bottom right corner.

Project Tree:

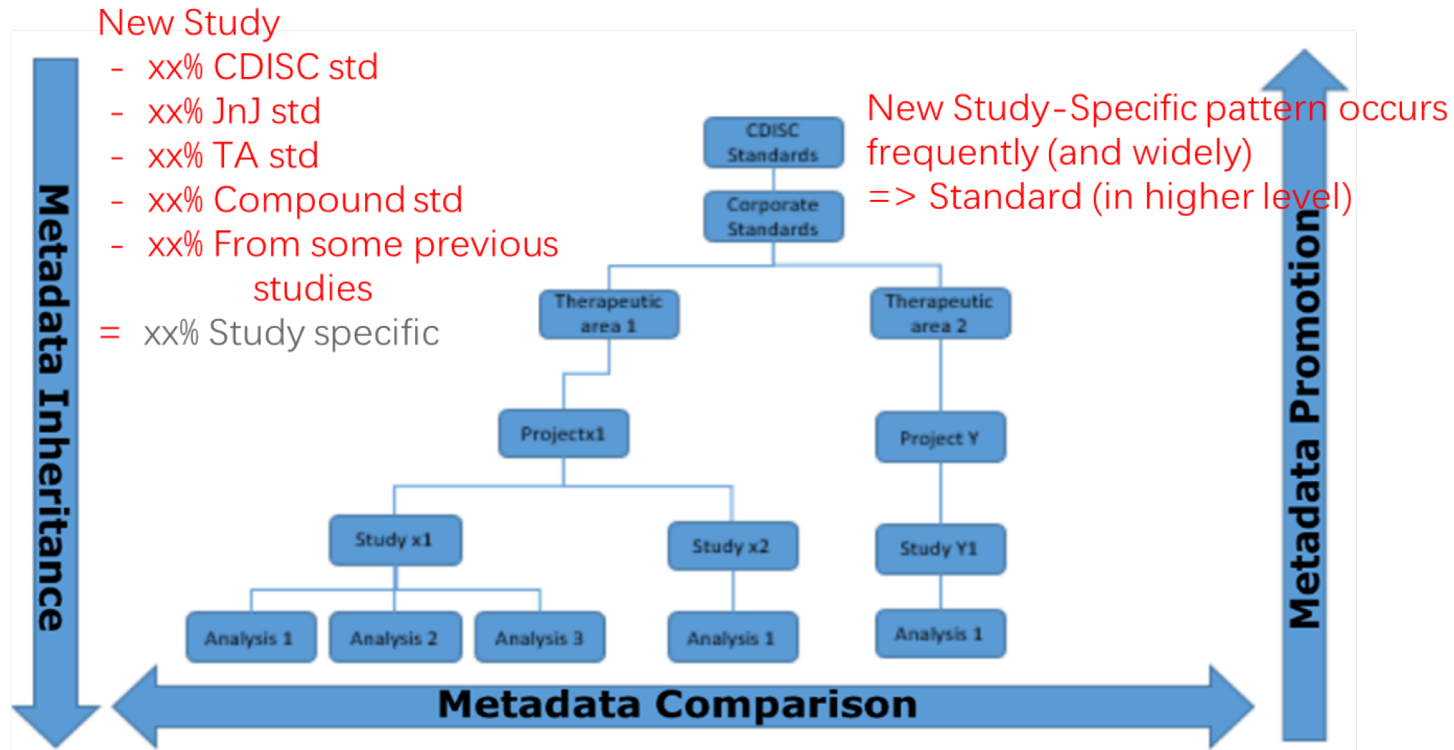
- Link to PROC COMPARE
- Link to PROC SQL
- Programs
 - PROC SQL
 - PROC SQL
 - PROC SQL
 - DATA Step
 - PROC FREQ
 - PROC SQL
 - PROC SQL
 - PROC IMPORT
 - DATA Step
 - PROC SQL
 - PROC SQL
 - PROC SORT
 - DATA Step
 - DATA Step
 - PROC SORT
 - PROC SQL
 - DATA Step
 - DATA Step
 - PROC FREQ
 - DATA Step
 - PROC SQL
 - DATA Step
 - DATA Step
 - PROC SORT
 - DATA Step
 - DATA Step
 - PROC SORT
 - DATA Step
 - DATA Step
 - PROC SQL
- WORK EX
 - Link to PROC SQL
 - Link to PROC SQL
 - WORK_ADDTM03
 - Link to PROC SQL
 - WORK_TRTSDT
 - Link to DATA Step
 - WORK_TRTSDTX
 - Link to DATA Step
 - WORK_ADLB04
 - Link to DATA Step
 - WORK_ADLB05
 - Link to DATA Step
 - WORK_ADLB06
 - Link to DATA Step
 - WORK_ADLB07
 - Link to DATA Step
 - WORK_ADLB08
 - Link to DATA Step
 - WORK_ADLB09
 - Link to DATA Step
 - WORK_ADLB10
 - Link to DATA Step
 - WORK_ADLB11
 - Link to DATA Step
 - WORK_ADLB12
 - Link to DATA Step
 - WORK_ADLB13
 - Link to DATA Step
 - WORK_ADLB14
 - Link to DATA Step
 - WORK_ADLB15
 - Link to DATA Step
 - WORK_ADLB16
 - Link to DATA Step
 - WORK_ADLB17
 - Link to DATA Step
 - WORK_ADLB18
 - Link to DATA Step
 - WORK_ADLB19
 - Link to DATA Step
 - WORK_ADLB20
 - Link to DATA Step
 - WORK_ADLB21
 - Link to DATA Step
 - WORK_ADLB22
 - Link to DATA Step
 - WORK_ADLB23
 - Link to DATA Step
 - WORK_ADLB24
 - Link to DATA Step
 - WORK_ADLB25
 - Link to DATA Step
 - WORK_ADLB26
 - Link to DATA Step
 - WORK_ADLB27
 - Link to DATA Step
 - WORK_ADLB28
 - Link to DATA Step
 - WORK_ADLB29
 - Link to DATA Step
 - WORK_ADLB30
 - Link to DATA Step
 - WORK_ADLB31
 - Link to DATA Step
 - WORK_ADLB32
 - Link to DATA Step
 - WORK_ADLB33
 - Link to DATA Step
 - WORK_ADLB34
 - Link to DATA Step
 - WORK_ADLB35
 - Link to DATA Step
 - WORK_ADLB36
 - Link to DATA Step
 - WORK_ADLB37
 - Link to DATA Step
 - WORK_ADLB38
 - Link to DATA Step
 - WORK_ADLB39
 - Link to DATA Step
 - WORK_ADLB40
 - Link to DATA Step
 - WORK_ADLB41
 - Link to DATA Step
 - WORK_ADLB42
 - Link to DATA Step
 - WORK_ADLB43
 - Link to DATA Step
 - WORK_ADLB44
 - Link to DATA Step
 - WORK_ADLB45
 - Link to DATA Step
 - WORK_ADLB46
 - Link to DATA Step
 - WORK_ADLB47
 - Link to DATA Step
 - WORK_ADLB48
 - Link to DATA Step
 - WORK_ADLB49
 - Link to DATA Step
 - WORK_ADLB50
 - Link to DATA Step
 - WORK_ADLB51
 - Link to DATA Step
 - WORK_ADLB52
 - Link to DATA Step
 - WORK_ADLB53
 - Link to DATA Step
 - WORK_ADLB54
 - Link to DATA Step
 - WORK_ADLB55
 - Link to DATA Step
 - WORK_ADLB56
 - Link to DATA Step
 - WORK_ADLB57
 - Link to DATA Step
 - WORK_ADLB58
 - Link to DATA Step
 - WORK_ADLB59
 - Link to DATA Step
 - WORK_ADLB60
 - Link to DATA Step
 - WORK_ADLB61
 - Link to DATA Step
 - WORK_ADLB62
 - Link to DATA Step
 - WORK_ADLB63
 - Link to DATA Step
 - WORK_ADLB64
 - Link to DATA Step
 - WORK_ADLB65
 - Link to DATA Step
 - WORK_ADLB66
 - Link to DATA Step
 - WORK_ADLB67
 - Link to DATA Step
 - WORK_ADLB68
 - Link to DATA Step
 - WORK_ADLB69
 - Link to DATA Step
 - WORK_ADLB70
 - Link to DATA Step
 - WORK_ADLB71
 - Link to DATA Step
 - WORK_ADLB72
 - Link to DATA Step
 - WORK_ADLB73
 - Link to DATA Step
 - WORK_ADLB74
 - Link to DATA Step
 - WORK_ADLB75
 - Link to DATA Step
 - WORK_ADLB76
 - Link to DATA Step
 - WORK_ADLB77
 - Link to DATA Step
 - WORK_ADLB78
 - Link to DATA Step
 - WORK_ADLB79
 - Link to DATA Step
 - WORK_ADLB80
 - Link to DATA Step
 - WORK_ADLB81
 - Link to DATA Step
 - WORK_ADLB82
 - Link to DATA Step
 - WORK_ADLB83
 - Link to DATA Step
 - WORK_ADLB84
 - Link to DATA Step
 - WORK_ADLB85
 - Link to DATA Step
 - WORK_ADLB86
 - Link to DATA Step
 - WORK_ADLB87
 - Link to DATA Step
 - WORK_ADLB88
 - Link to DATA Step
 - WORK_ADLB89
 - Link to DATA Step
 - WORK_ADLB90
 - Link to DATA Step
 - WORK_ADLB91
 - Link to DATA Step
 - WORK_ADLB92
 - Link to DATA Step
 - WORK_ADLB93
 - Link to DATA Step
 - WORK_ADLB94
 - Link to DATA Step
 - WORK_ADLB95
 - Link to DATA Step
 - WORK_ADLB96
 - Link to DATA Step
 - WORK_ADLB97
 - Link to DATA Step
 - WORK_ADLB98
 - Link to DATA Step
 - WORK_ADLB99
 - Link to DATA Step
 - WORK_ADLB100
 - Link to DATA Step

AUTOCODE PROJECT

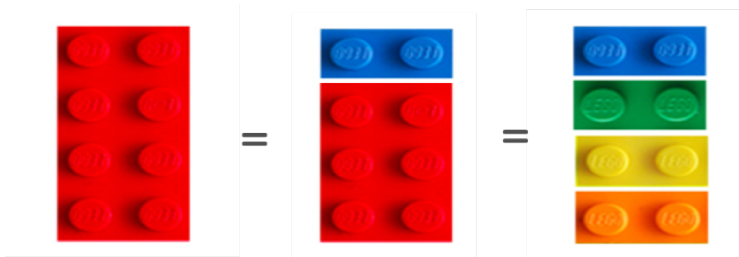


Exploration / Map / Auto-Navigation
Open-Code / Standard/ Auto-Code
Experience / Documentation/ System automation

METADATA DRIVEN APPROACH



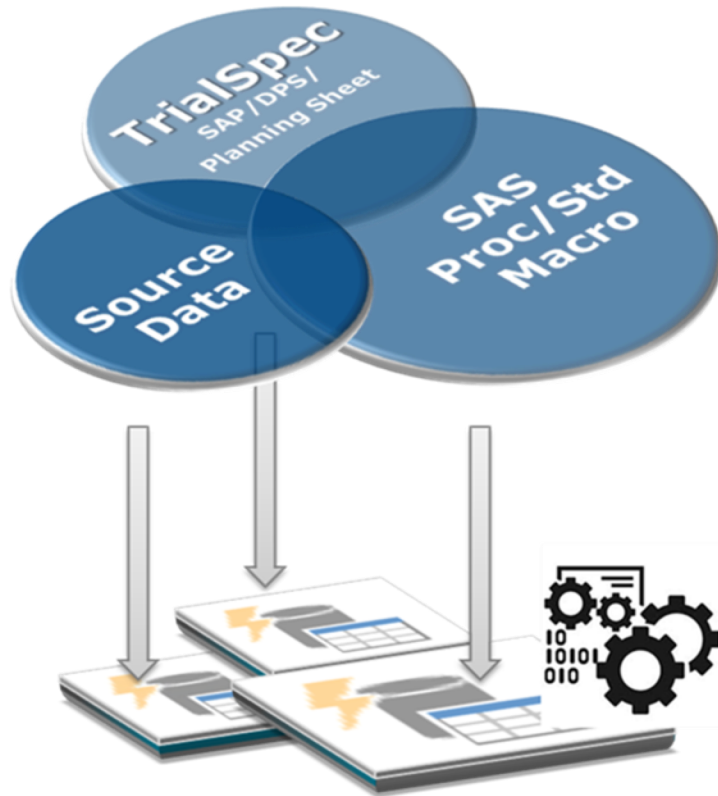
Source: PhUSE 2017 Paper SI02 Clinical Metadata – Metadata Management with a CDISC Mindset



- Exact match (rule-based system)
- Fuzzy match
- Smart match

METADATA DRIVEN APPROACH

- MODULES



- **Digitization & Structuralization**
Human readable
-> **Machine** readable



```
data c;  
  merge a b;  
  by usubjid;  
run;
```


WHY NLP (NATURAL LANGUAGE PROCESSING) ?

- ✓ Future trend
- ✓ Document orientated process
- ✓ Standard input and output
- ✓ Table data, general data manipulation
 - SAS/R Tidyverse

WHY NLP ?

Future Trend

- Strategic Planning Assumptions

By 2020, augmented analytics — a paradigm that includes **natural language** query and narration, augmented data preparation, automated advanced analytics and visual-based data discovery capabilities — will be a dominant driver of new purchases of business intelligence, analytics and data science and machine learning platforms and of embedded analytics.

...

By 2020, **natural-language** generation and artificial intelligence will be a standard feature of 90% of modern business intelligence platforms.

By 2020, 50% of analytical queries will be generated via search, **natural-language** processing or voice, or will be automatically generated.

...

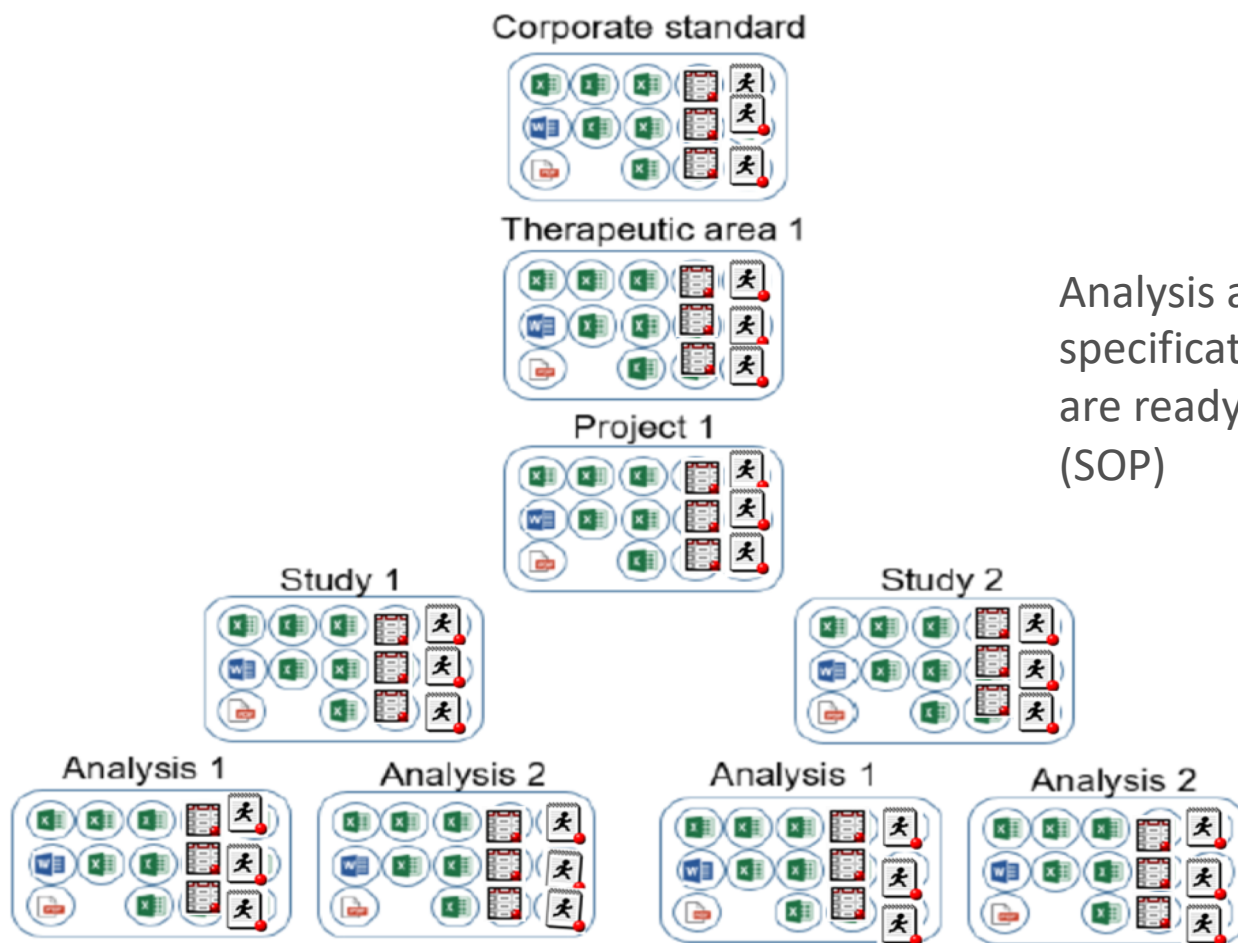
From <<https://www.gartner.com/doc/reprints?id=1-3TXXSLV&ct=170221&st=sb>>

- Satya Nadella, CEO of Microsoft, said artificial intelligence will be crucial for the coming age in which **speaking and texting will be “the new U.I.”** for software.

From <https://blogs.wsj.com/digits/2015/11/13/microsofts-satya-nadella-for-one-welcomes-our-new-ai-overlords/>

WHY NLP ?

Documents/Data/Codes



Analysis algorithm
specification documents
are ready before coding
(SOP)

Source: PhUSE 2017 Paper SI02 Clinical Metadata – Metadata Management with a CDISC Mindset

WHY NLP ?

Documents/Data/Codes

Standard input and output

Input Set - *SDTM* Data Standard

SDTM Variable Category	Count
⊕ GroupingQualifier	6
⊕ IdentifierVariable	7
⊕ RecordQualifier	54
⊕ ResultQualifier	3
⊕ SynonymQualifier	7
⊕ TimingVariable	24
⊕ TopicVariable	3
⊕ VariableQualifier	21
⊕ (blank)	3
Grand Total	128

Study Data Tabulation Model (SDTM)

Output Set - *ADaM* Data Standard – naming convention / structure

Count of Variable Name	Data Structure Name		
Variable Group	Basic Data Structure	Subject Level Analysis Dataset	Grand Total
Analysis Descriptor Variables	1		1
Analysis Parameter Variables	23		23
Analysis Visit Windowing Variables	6		6
Data Point Traceability Variables	3		3
Flag Variables	30		30
Lab Related Analysis Variables	8		8
Population Indicator(s)		7	7
Study Identifiers		6	6
Subject Demographics		6	6
Subject Identifier Variable	4		4
Time to Event Variables	3		3
Timing Variables	56		56
Treatment Variables	8	13	21
Trial Dates		31	31
Grand Total	142	63	205

Analysis Data Model (ADaM)

<https://www.cdisc.org>

WHY NLP ?

Documents/Data/Codes

SAS

One data structure: Table

Two data types: Char/Num

Implicit Loop in data step

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

Common Data manipulation Set: -
Proc/DataStep

Table

Sort

Subset/Split/Combine

Set/Merge/Join

Summary

Record(Row)

Select

NewRecord

Variable(Column)

Select

NewVar/NewValue

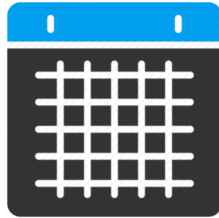
Transpose

[Source: SAS® Studio 3.71 with SAS® 9.4 and SAS® Viya® Programming Documentation/ SAS® Studio 3.71: Task Reference Guide](#)

WHY NLP ?

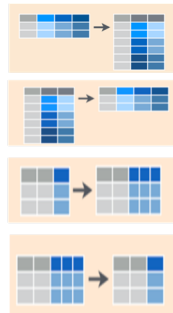
Documents/Data/Codes

R Tidyverse

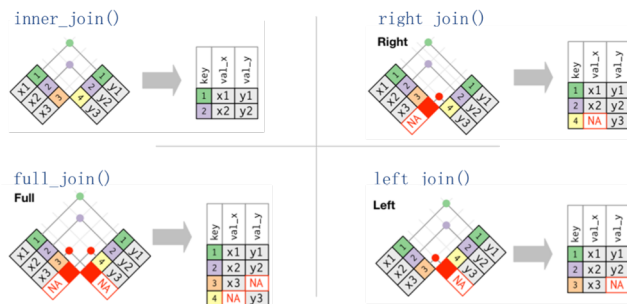


reshape data

- `gather()`
- `spread()`
- `separate()`
- `unite()`



combine data sets



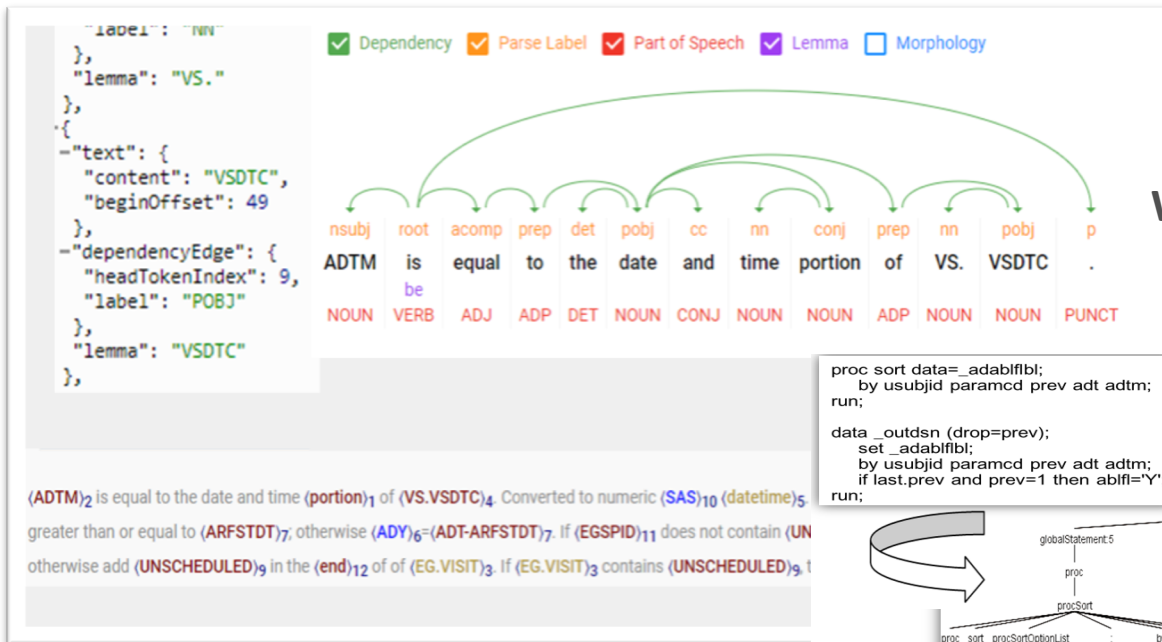
Data transformation

- `select()` relevant variables
- `filter()` out unnecessary observations
- `mutate()` new variables, or change existing ones
- `arrange()` in an easy-to-digest format
- `group_by()` groups the data according to some variable
- `summarize()` aggregates values in each group using a given function (`n()`, `mean()`, etc.)

Source: <https://jjrsvp.slack.com/> Channel: #tidyverse

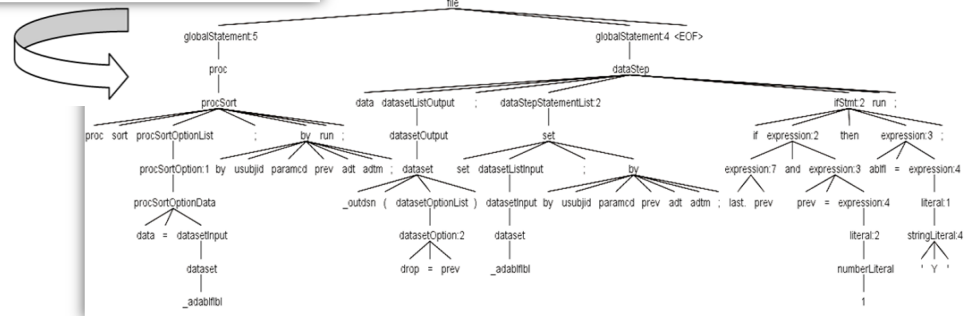
NLP (NATURAL LANGUAGE PROCESSING)

- Map human language to Code



Word, Phrase, Sentence, Paragraph

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



<http://www.antlr.org/>

Variable, Function, Data Step, PROC, Macro, Program

NLP (NATURAL LANGUAGE PROCESSING)

- Map human language to Code

ADaM Planning Sheet

DATASET	VARNAME	VARLABEL	ORIGCOM	DERVCOPY	SASMC
ADAE	AENTMF	Analysis End Time Imputation Flag	Set to 'M' if minute is missing and imputed. Set to 'H' if hour and minute are missing and imputed.	Derived	%ADAEDT
ADAE	ASTDY	Analysis Start Relative Day	ASTDY = ASTDT - ARFSTDT + 1 if ASTDT >= ARFSTDT, or ASTDT - ARFSTDT if ARFSTDT is after ASTDT. If the dates are missing ASTDY will be missing.	Derived	%ADAEDT
ADAE	AENDY	Analysis End Relative Day	AENDT-ARFSTDT+1 if on or after ARFSTDT, otherwise, AENDT-ARFSTDT.	Derived	%ADAEDT
ADAE	ADURN	AE Duration (N)	ADURN is the duration of the AE in days from onset date to AE stop date. AE.AEENDY - AE.AESTDY + 1. If using the numeric dates AENDT - ASTDT + 1, then check that the If ADSL.ARFSTDT=(ASTDT-(ADSL.TRTEDT + [xx]) days then TRTEMFL= 'Y'. See supplementaldefinitions.pdf for derivation of the flag in case of missing dates.	Derived	%ADADURN
ADAE	TRTEMFL	Treatment Emergent Analysis Flag		Derived	%ADTRTEMFL
ADAE	AESEVN	Severity/Intensity (N)	Code AE.AESEV to numeric value. Low intensity should correspond to low value.	Derived	%ADSEV

Code

```

*****
* Macro summary : Macro to create date, datetime of analysis reference date
* Output summary: Input dataset in INDSN with ARFSTDT and ARFSTDTM added to create the
* Variable list : ARFSTDT ARFSTDTM
*****/
%ADARFSTDT(in= _QTRTA
           ,keepvars = ARFSTDT ARFSTDTM
           ,out= _QADAEEDT
           ,debug=OFF);

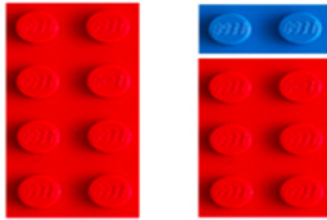
*****
* Macro summary : Macro to generate a duration variable (in minutes, hours, days, weeks,
* : years) from two input variables and the corresponding duration units v
* Output summary: input dataset in INDSN with variables added to create the output dates
* : OUTDSN depending on the sOUTVAR (duration) and sUNITVAR (duration unit
* Variable list : ADURN ADURNV
*****/
%ADADURN(in= _QADAEEDT
         ,out= _QADADURN
         ,stdt=astdt
         ,enddt=aendt
         ,outvar=adurn
         ,plusone=Y
         ,unit=Day
         ,unitvar=adurnv
         ,debug=OFF);

*****
* Macro summary : Macro to create Treatment Emergent Analysis Flag
* Output summary: input dataset in INDSN with TRTEMFL variable added to create the outpu
* : in OUTDSN
* Variable list : TRTEMFL
*****/
%ADTRTEMFL(in= _QADADURN
           ,invar=adayvar
           ,adrule = VERSION1
           ,out= _LOADTRTEMFL
           ,debug=OFF);

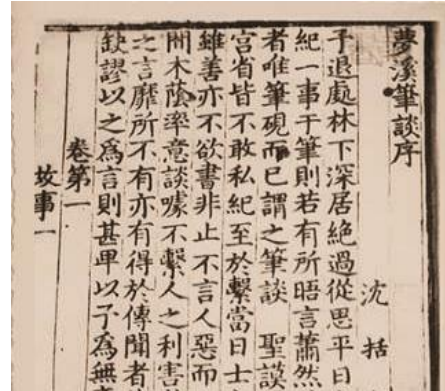
```

NLP - Unsupervised vs. Supervised

Similar sentence to similar code



More Reusable Patterns
– meaningful pieces



Woodblock printing

vs.



Movable type

flexible components to reproduce the elements of a document

NLP - Unsupervised approach - Similar sentence to similar code

- Fuzzy String Matching
- Word Embedding
- Word Mover's Distance(WMD)
- Sentence Embedding

Unsupervised approach

Fuzzy String Matching

Simple Ratio

```
>>> fuzz.ratio("this is a test", "this is a test!")
97
```

Partial Ratio

```
>>> fuzz.partial_ratio("this is a test", "this is a test!")
100
```

Token Sort Ratio

```
>>> fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
91
>>> fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
100
```

Token Set Ratio

```
>>> fuzz.token_sort_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")
84
>>> fuzz.token_set_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")
100
```

```
from fuzzywuzzy import fuzz

textA = "'Y' if index(ATRTRF, 'BEFORE/DURING') or ASTDT=AP01SDT"
textB = "if index(atrtrf,'BEFORE/DURING') or astdt =ap01sdt then ablfl='Y'"

fuzz.ratio(textA.upper(),textB.upper())/100

0.82
```

<https://www.neudesic.com/blog/fuzzywuzzy-using-python/>

<https://github.com/seatgeek/fuzzywuzzy>

Unsupervised approach

Word Embedding

	DOG	CAT	BANANA
DOG	1.00 ●	0.80 ●	0.24 ✖
CAT	0.80 ●	1.00 ●	0.28 ✖
BANANA	0.24 ✖	0.28 ✖	1.00 ●

- **similarity:** identical
- **similarity:** similar (higher is more similar)
- ✖ **similarity:** dissimilar (lower is less similar)

BANANA.VECTOR

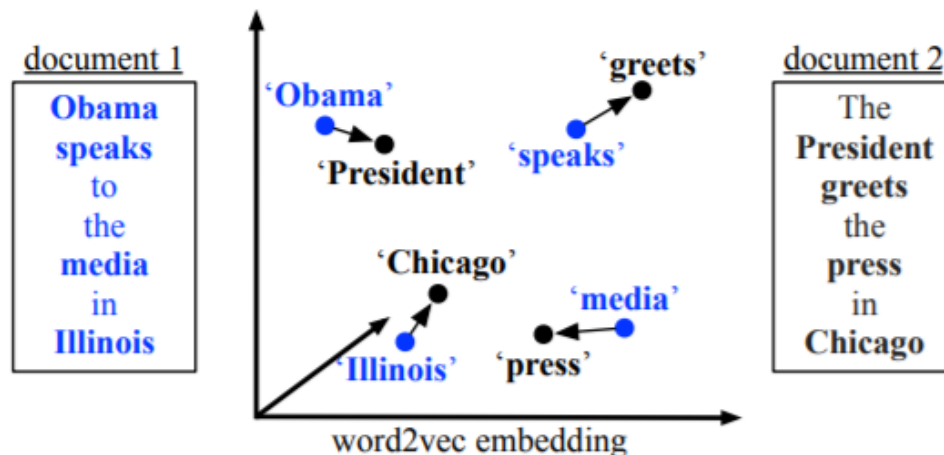
```
array([ 2.02280000e-01, -7.66180009e-02,  3.70319992e-01,
        3.28450017e-02, -4.19569999e-01,  7.20689967e-02,
       -3.74760002e-01,  5.74599989e-02, -1.24009997e-02,
        5.29489994e-01, -5.23800015e-01, -1.97710007e-01,
       -3.41470003e-01,  5.33169985e-01, -2.53309999e-02,
        1.73800007e-01,  1.67720005e-01,  8.39839995e-01])
```



<https://spacy.io/usage/vectors-similarity>

Unsupervised approach

Word Mover's Distance(WMD)



```
import textacy.similarity
import spacy
nlp = spacy.load('en_vectors_web_lg')

textA = "Defined as Closest Non-Missing Value prior to the first treatment date"
textB = "Defined as nearest non-Missing value before the first dosing date"

textacy.similarity.word_movers(nlp(textA), nlp(textB), metric='cosine')

0.8109184230589997
```

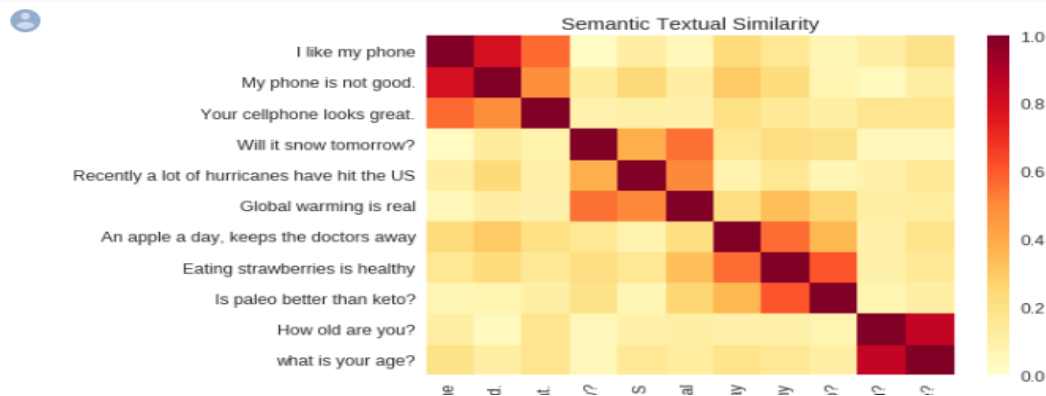
Unsupervised approach

Sentence Embedding

▼ Similarity Visualized

Here we show the similarity in a heat map. The final graph is a 9x9 matrix where each entry $[i, j]$ is colored based on sentence i and j .

```
[7] messages = [  
    # Smartphones  
    "I like my phone",  
    "My phone is not good.",  
    "Your cellphone looks great.",  
  
    # Weather  
    "Will it snow tomorrow?",  
    "Recently a lot of hurricanes have hit the US",  
    "Global warming is real",  
  
    # Food and health  
    "An apple a day, keeps the doctors away",  
    "Eating strawberries is healthy",  
    "Is paleo better than keto?",  
  
    # Asking about age  
    "How old are you?",  
    "what is your age?",  
]  
  
similarity_input_placeholder = tf.placeholder(tf.string, shape=(None))  
similarity_message_encodings = embed(similarity_input_placeholder)  
with tf.Session() as session:  
    session.run(tf.global_variables_initializer())  
    session.run(tf.tables_initializer())  
    run_and_plot(session, similarity_input_placeholder, messages,  
                  similarity_message_encodings)
```

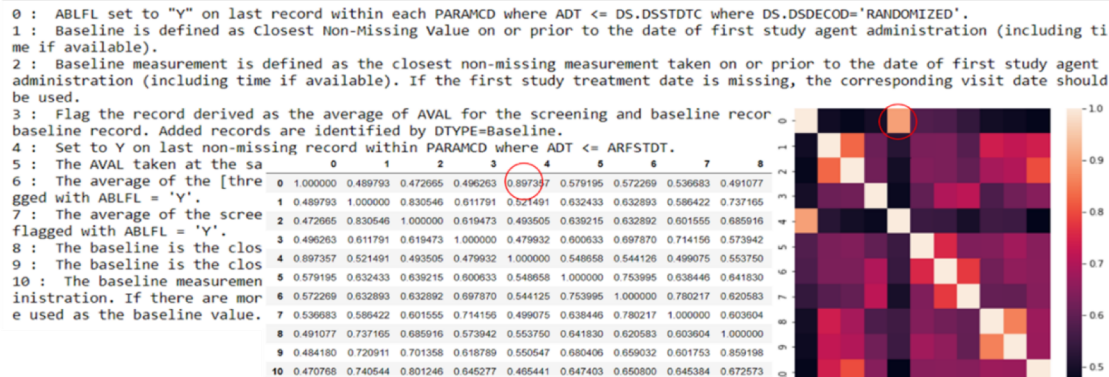


[Universal Sentence Encoder - Google Colab](#)

NLP - Unsupervised approach

Find similar sentence, copy & tweak correspondent codes

Text Similarity Matrix



0 : ABLFL set to "Y" on last record within each PARAMCD where ADT <= DS.DSSTDTC where DS.DSDECD='RANDOMIZED'.
 ==> 4: Set to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.



```

/*****
* Macro summary : Macro to create baseline flag : ABLFL variable.
* Output summary: input dataset in INDSN with ABLFL variable added to
*                 : OUTDSN.
* Variable list : ABLFL
*****/

%ADABLFL(indsn = xxx
        ,outdsn = xxx
        ,subset = %str(<ADT<=ARFSTDT)
        ,adrule = VERSION1);
  
```

Tweak the code if needed!

MORE REUSABLE PATTERNS – meaningful pieces

Baseline is defined/Set to Y /The baseline is/flagged /ABLFL = 'Y'

last /closest

non-missing assessment/value

prior to or on/<=

Dosing/Study Day 1/first dose date/first reference date/ARFSTDT

within each PARAMCD/of PSA

new record is created/Added records

screening and baseline records /values

average of

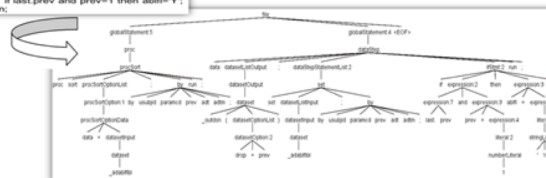
VARVER	VERDESC	ORIGCOM	SASMCRCR
1	Closest Non-Missing Value to	Baseline is defined as last non-missing assessment prior to dosing	%ADABLFL
		Set to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.	%ADABLFL
		The baseline is the closest non-missing value prior to or on Study Day 1	%ADABLFL
		ABLFL set to "Y" on the closest non-missing value prior to or on first dose date (or randomization date if subject is not treated) within each PARAMCD.	%ADABLFL
		Set to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.	%ADABLFL
		The baseline is the closest non-missing value of PSA prior to or on Study Day 1.	%ADABLFL
2	Average of all values prior	The average of the [three] replicate ECG parameters taken prior to or on reference date. The new record is created and flagged with ABLFL = 'Y'.	%ADABLFL
3	Average of screening and baseline	Flag the record derived as the average of AVAL for the screening and baseline records prior to first reference date as the baseline record. Added records are identified by DTYPE=Baseline.	%ADABLFL
		The average of the screening and baseline ECG values is used as the baseline for each subject. A new record is created and flagged with ABLFL = 'Y'.	%ADABLFL

MORE REUSABLE PATTERNS – meaningful pieces

Max of MACSEQ							
MACNAME		PARAM	EQU	PARAMVAL			
%DATA		assignment	=	%str(avisit='ANY POST-BASELINE VALUE';)		1	
				%str(avisitn=10;)		1	
				%str(indent=2;)	5		
		by	=	%str(by by_order1 by_order2;)		6	4
				%str(by paramn;)			1
		if	=	%str(if avisitn<=1 or aval=. or avisitn=9999)			1
				%str(if avisitn<=1 or aval=. then delete;)		1	
		indsn	=	a_in.adlb	2		1
				tbl	5	6	6
		outdsn	=	adlb	2	1	1
				tbl	5	6	6
		where	=	%str(where avisit ne " & anl01fl='Y' and index(avisit,'ENDPOINT') eq	2		
%PROC_SORT		by	=	%str(by anbr roworder by_order1 by_order2;)		5	3
				%str(by avisitn;)		2	
				%str(by paramn;)		3	
		indsn	=	a_in.adlb(where=(crit1^=" or crit2^=" or crit3^="))	3		
				adlb	2		
				tbl	5	3	
		outdsn	=	avisit(keep=avisit avisitn) nodupkey	2		
				param(keep=paramcd paramn) nodupkey	3		
%RMTGENTBL01		fileid	=	tsflab01	7		
				TSFLAB04		7	
				TSFLAB05			7
		grpvar	=	group1	7		
				labgrp		7	7

```
proc sort data=_adabtbl;
  by usubjcd paramcd prev adt adm;
run;

data _outdsn (drop=prev);
  set _adabtbl;
  by usubjcd paramcd prev adt adm;
  if last.prev and prev=1 then adt=Y;
run;
```

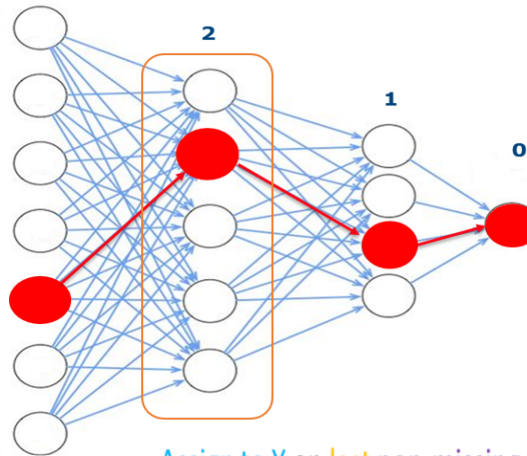


STUDY	PrgName	_RE
-Compound		
tsflab01	tsflab04	tsflab05
DATA	CENTI	DATA
DATA	CENTI	DATA
DATA	CENTI	DATA

PATTERN ALGORITHMS

Complex algorithm decomposed into pattern algorithms

Level 3	Level 2	Level 1	Level 0
DataProc	Pattern	ADMAName	ProductionCode
Sort	subset/where	%adabfl	ADXX.sas 3
Subset	equal to source value	%adadurn	
Summary	DTC to datetime	%adady	
addNew	assign	%adaecqnam	
split	seq/code/group/category	%adaedt	
combine	IF-THEN/ELSE logic	%adaetox	
merge/join	relative day, before/after	%adagegr	
transpose	first/last/closest/highest	%adanl01fl	
...	/lowest/worst	%adaocfl	
	min/max/average/total	%adarfsttd	
	formula calculation	%adatpt	
	non-missing/censored	%adavisfl	
	external mapping file	%adavisit	
		%adbsavisit	
		%adblvars	
		%adbmi	
		%adbsa	
		%adchg	
		%adclear	
		%adcmqnam	
		%adcmdt	
		%adcmfupfl	
		%adcmonttrfl	
		%adcmprefl	
		%adadurn	



Assign to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.

`%Assign(indsn=xxx,value='Y',by=PARAMCD,select=last non-missing,subset=%str(<ADT<=ARFSTDT) )`

`%SUBSET(<ADT<=ARFSTDT)`
`%SUBSET(not missing(AVAL))`
`%SORTBY(PARAMCD ADT)`
`%SUBSET(last.PARAMCD)`
`%ASSIGN(ABLFL='Y')`

NLP - Supervised approach

1) Split algorithm texts into pieces

- NLP tools (NLTK and spaCy) with manually reviews
- Train model to auto split

```
# Extract Keywords using noun chunks
text = str(Origcom)
doc = nlp(text)
keywords = Counter()
for chunk in doc.noun_chunks:
    keywords[chunk.lemma_] += 1

keywords.most_common(50)
```

```
[('the average', 3),
 ('reference date', 3),
 ('ablfl = ' y', 3),
 ('the baseline', 3),
 ('y', 2),
 ('the date', 2),
 ('first study agent administration', 2),
 ('time', 2),
 ('the screening', 2),
 ('the close non - miss value', 2),
 ('last record', 1),
 ('each paramcd', 1),
 ('where adt < = ds.dsstdtc', 1),
```

```
grammar = r"""
NP:
    {<.*>+}          # Chunk everything
    }((<WP|WDT|WP$|WRB|IN>)+){      # Chink sequence
    """
```

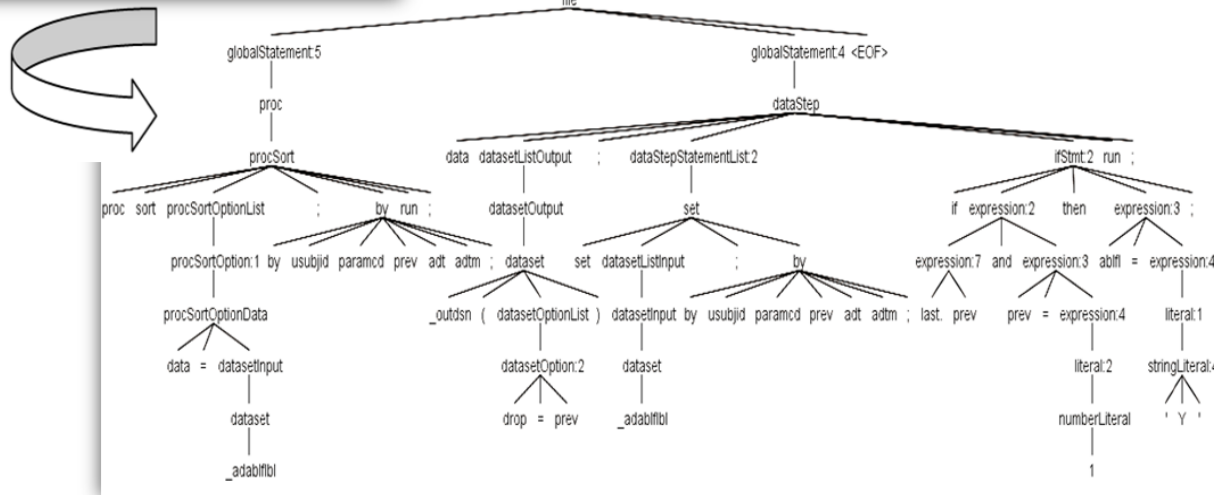
#WDT wh-determiner which
#WP wh-pronoun who, what
#WP\$ possessive wh-pronoun whose
#WRB wh-abverb where, when
#IN (Prepositions and Subordinating conjunctions)

Set to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.
=> ['set to y', 'last non-missing record', 'paramcd', 'adt < = arfsttdt .']

NLP - Supervised approach

2) Parse codes and decompose to smaller building blocks - ANTLR

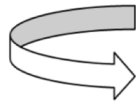
```
proc sort data=_adabflbl;  
  by usubjid paramcd prev adt adtm;  
run;  
  
data _outdsn (drop=prev);  
  set _adabflbl;  
  by usubjid paramcd prev adt adtm;  
  if last.prev and prev=1 then abfl='Y';  
run;
```



NLP - Supervised approach

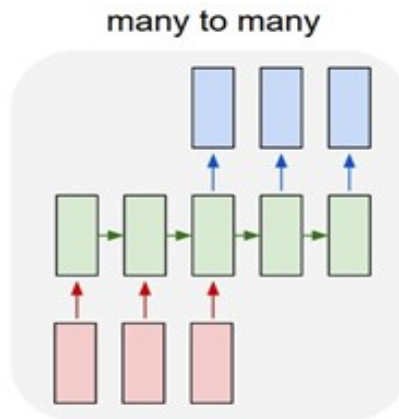
- 3) Map algorithm text pieces to code pieces
- training data

Assign to Y on last non-missing record within PARAMCD where ADT <= ARFSTDT.



```
%Assign(indsn=xxx,value='Y',by=PARAMCD,select=last non-missing ,subset=%str(<ADT<=ARFSTDT) )
```

- 4) Train an RNN model



Conclusion

Key Success Factors

- Data
- Standard
- Technology



Moore's Law provides exponential growth in...

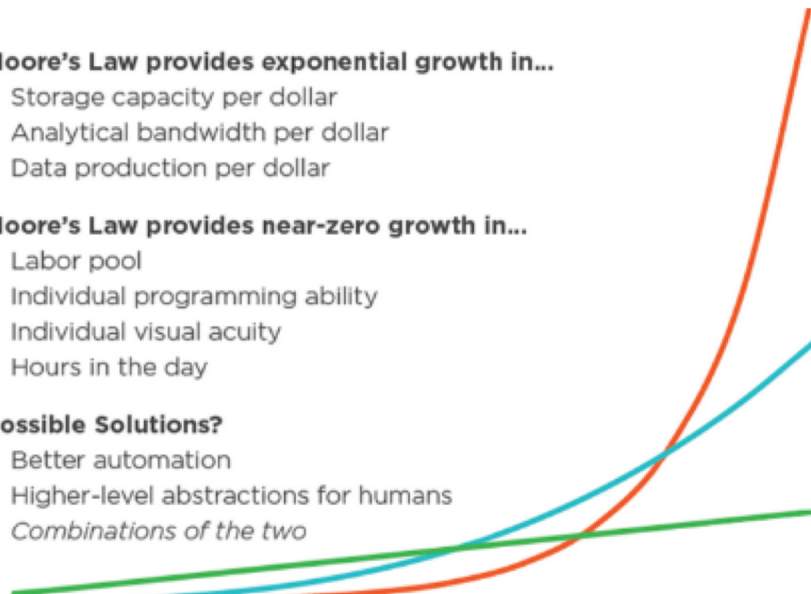
Storage capacity per dollar
Analytical bandwidth per dollar
Data production per dollar

Moore's Law provides near-zero growth in...

Labor pool
Individual programming ability
Individual visual acuity
Hours in the day

Possible Solutions?

Better automation
Higher-level abstractions for humans
Combinations of the two



SOURCE: <http://conferences.oreilly.com/strata/strata2014/public/schedule/detail/3295>

*Thank
you*





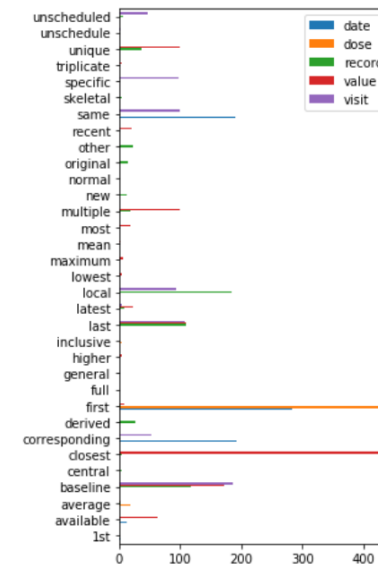
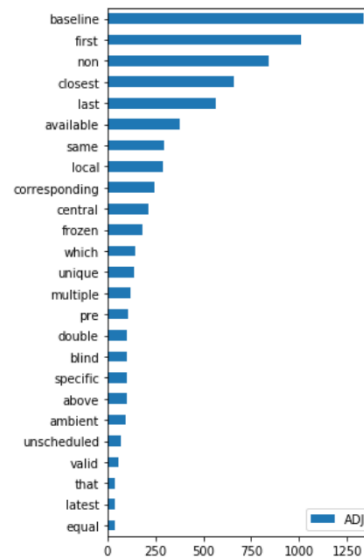
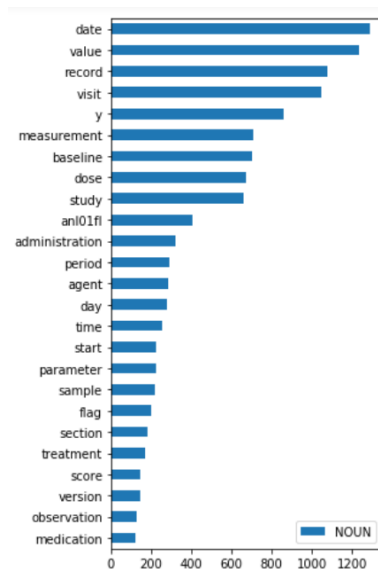
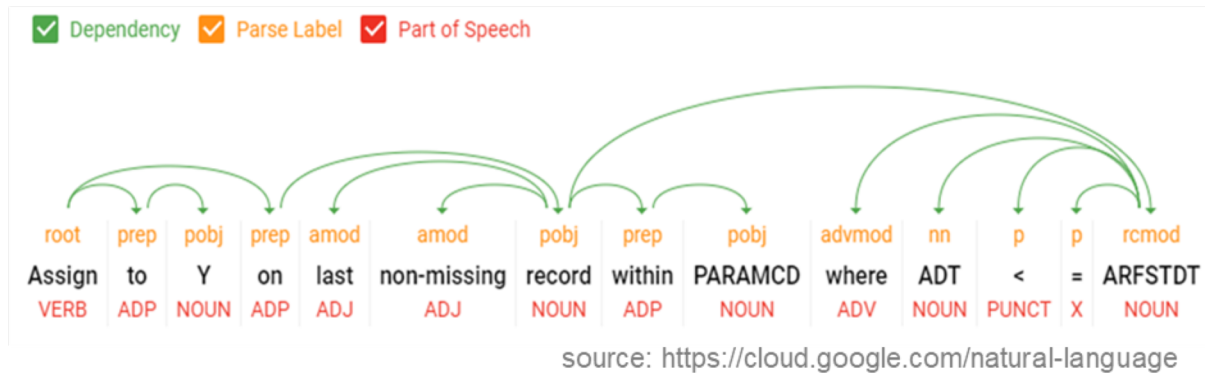
Sumesh Kalappurakal
Janssen R&D US
Phone: (908) -927-6715
e-mail: SKalappu@its.jnj.com

Qian Zhao
Janssen R&D China.
Phone: 86-21-3337-7624
e-mail: qzhao1@its.jnj.com

Harry Chen
Janssen R&D China.
Phone: 86-21-3337-8892
e-mail: hchen104@its.jnj.com

backup

NLP - NATURAL LANGUAGE PROCESSING



	Max of MACSEQ						_STUDY	Prg
							-Compound	
							tsflab04	
	MACSEQ	MACNAME	PA	PARAM	EQUAL	PARAMVAL	DATACENTER	
PrgName	1	%DATA	1	outdsn	=	adlb	1	
tsflab02			2	if	=	%str(if avisitn<=1 or aval=, then delete;)	1	
tsflab03			3	assignment	=	%str(avisitn=10;)	1	
tsflab04			4	assignment	=	%str(avisit='ANY POST-BASELINE VALUE;')	1	
tsflab05	2	%PROC_SORT	1	indsn	=	adlb	2	
tsflab06								
tsflab07								
tsflab08								
tsflab09								
tsicm01								
tsider01								
tsider02								
tsids01								
tsidv01								
tsienr01								
tsiexs01								
tsiexs02								
tsira01								
tsfae14								
tsfae17								
tsflab01								
tsflab02								
tsflab03								
tsflab04								
tsflab05								
tsflab06								
tsicmst01								
tsidem06								
tsidemst01								
tsids01								
tsidv01								
tsidv02								
tsienr101								

	Max of MACSEQ						_STUDY	PrgName	_RE
							-Compound		
							tsflab01	tsflab04	tsflab05
	MACNAME	PARAM	EQU	PARAMVAL			DATACENTER1	DATACENTER2	DATACENTER3
%DATA	assignment	=		%str(avisit='ANY POST-BASELINE VALUE;')				1	
				%str(avisitn=10;)				1	
				%str(indent=2;)			5		
by		=		%str(by by_order1 by_order2;)				6	4
				%str(by paramn;)					1
if		=		%str(if avisitn<=1 or aval=, or avisitn=9999)					1
				%str(if avisitn<=1 or aval=, then delete;)				1	
indsn		=		a_in.adlb			2		1
				tbl			5	6	6
outdsn		=		adlb			2	1	1
				tbl			5	6	6
where		=		%str(where avisit ne '' & anl01fl='Y' and index(avisit,'ENDPOINT') eq			2		
%PROC_SORT	by	=		%str(by anbr roworder by_order1 by_order2;)				5	3
				%str(by avisitn;)				2	
				%str(by paramn;)				3	
	indsn	=		a_in.adlb(where=(crit1^=" or crit2^=" or crit3^="))				3	
				adlb				2	
				tbl				5	3
outdsn		=		avisit(keep=avisit avisitn) nodupkey				2	
				param(keep=paramcd paramn) nodupkey				3	
%RMTGENTBL01	fileid	=		tsflab01			7		
				TSFLAB04				7	
				TSFLAB05					7
	grpvar	=		group1			7		
				labgrp				7	7

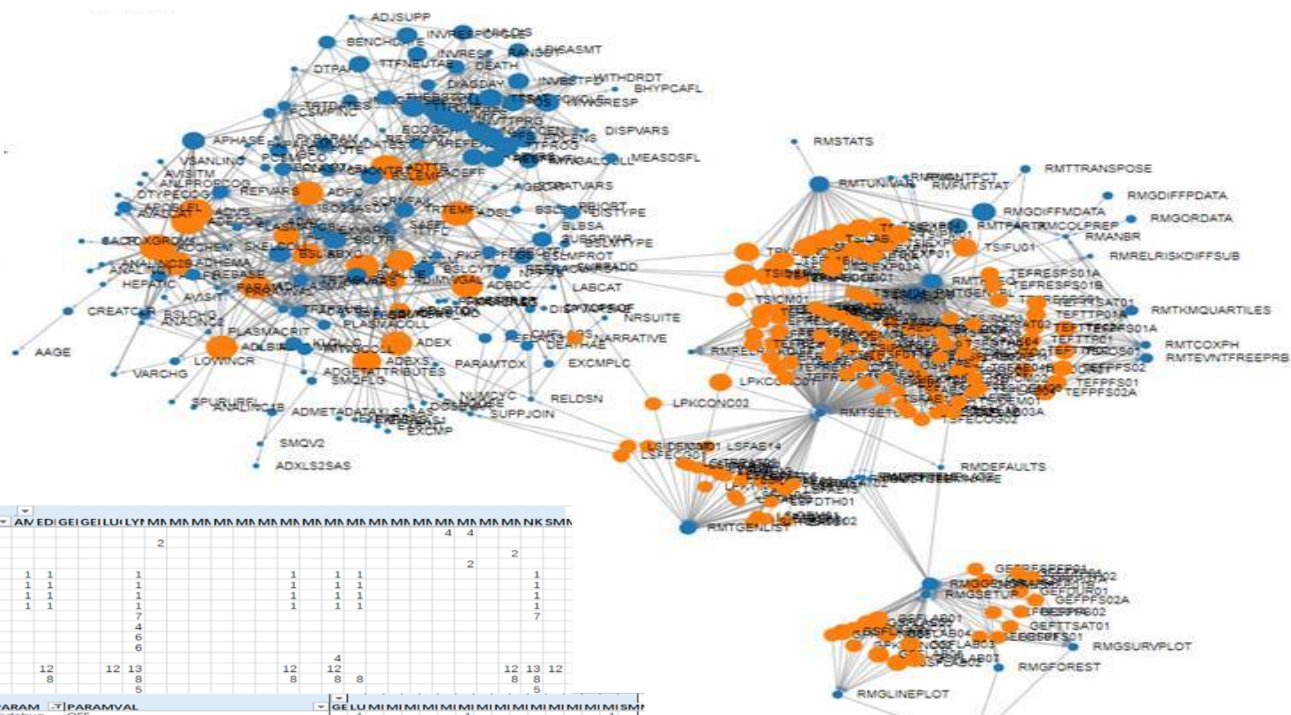
MODULES

- 1. PDE (PowerDataExplorer)

The screenshot displays the PowerDataExplorer (PDE) interface, which is used for managing and analyzing clinical trial data. The interface is divided into several sections:

- Top Section:** Contains various filters and selection tools for different data elements.
 - studyID:** A list of study identifiers (e.g., 12001, 2001, 2001-BASE ONLY).
 - UserExclud:** A dropdown menu for selecting users to exclude.
 - Core:** A section for selecting core data elements (e.g., Eip, Perm).
 - Observation:** A section for selecting observation types (e.g., All Classes, Interventions).
 - Role:** A section for selecting roles (e.g., Grouping Qualifier, Identifier).
 - MEMNAME:** A section for selecting memory names (e.g., EG, EX, FA, HD, IE, IS, LB).
 - Level:** A section for selecting levels (e.g., -298, -297, -73, -35, -31, -18, -16, -15, -11, -2, 3, 5, 6).
 - Variable_Na:** A section for selecting variable names (e.g., ADJ, CAT, DOMAIN, DOSE, DOSFRM, DOSFRQ, DOSU, ENDTC, ENDY, EPOCH, GRPID, LOT).
- Table View:** A large table displaying data for a specific study (e.g., 011). The table has columns for MEMNAME, NAME, LABEL, TYP, LEN, FOR, YAP, Control, N, Ch, NM, N, NS, studyID, and various data points (e.g., ADEFF, ADLB, ADLBC, ADLBCM, ADLBM, ADVS, ADVSM). The table is filtered to show data for the study 011.
- Types:** A section for selecting data types (e.g., APHASE*APHASE*APERIOD*AVISITN*AVISIT*AVISITDY*ATPTN*ATPT*PARCATY*PARAMCD*PARAM, #BODSYS*--DECOD*#TERM*#REL*#OUT, #DTC, #ENDTC*#ENRF*#ENRTP*#STDTC*#STRF*#STRTP, ARM*ACTARM*TRTSEQP*TRTSEQA*SEX).
- TypesMatched:** A section for selecting matched types (e.g., APHASE*AVISITN*AVISIT, APHASE*AVISITN*AVISIT, APHASE*AVISITN*AVISIT, APHASE*PARAMCD*PA, PARAMCD*PARAM, ADJOTC).

- 2. PME (PowerMacroExplorer)

[illegible]

Demo



1_datastep.gif



2_proc.gif



3_macro.gif



4_combined.gif



5_PME.gif

MODULES

- 2. PME (PowerMacroExplorer) parse data steps + procs + macros

ANTLR: Complex SAS codes with datasteps, macro calls and procs can be structuralized into XML and converted into database.

```
proc format;
  value primary
    1="Primary efficacy outcome"
    4="... Symptomatic lower extremity DVT";
run;

*****
* Create dummy treatment group
*****

*_testComplex dataset;
data data;
  set set;
  if if > 0;
  and = 1;
  where where='Y' and and;
run;

%*_testComplex macro call;
%complexMacCall(dsn =
  text= a b

- <stepStatement>
- <procFormat>
- <formatName>
  value primary
- <expressions>
  <expression1>1</expression1>
  <operator>=</operator>
  <expression2>"Primary efficacy outcome"</expression2>
- </expressions>
- <expressions>
  <expression1>4</expression1>
  <operator>=</operator>
  <expression2>"...Symptomatic lower extremity DVT"</expression2>
- </expressions>
- </formatName>
</procFormat>
</stepStatement>
- <stepStatement>
  <stmtComment>***** Create dummy treatment group *****</stmtComment>
  <stmtComment>*_testComplex dataset;</stmtComment>
</stepStatement>
- <stepStatement>
  <dataSet>
    <outdsn>data</outdsn>
    <indsn>set</indsn>
    <ifstmt>if if > 0;</ifstmt>
    <assignment>
      <idx>and</idx>
      <operator>=</operator>
      <expression>1</expression>
    </assignment>
    <wherestmt>where where='Y' and and;</wherestmt>
  </dataSet>
</stepStatement>
</stepStatement>
  <macroComment>%*_testComplex macro call;</macroComment>
</stepStatement>
- <stepStatement>
  <macroCall>
    <MACRONAME>%complexMacCall</MACRONAME>
    <macexpressions>
      <param>dsn</param>
      <operator>=</operator>
      <paramval>tbi</paramval>
    </macexpressions>
    <macexpressions>
      <param>text</param>
      <operator>=</operator>
      <paramval>a b c %str(,) %str(=) 'quoted,String' %str(!!!%str(%')xxx a=b=c ,,,, 'quoted,String' %str(unblanced parenthesis)
    </macexpressions>
  </macroCall>
```