Paper ML09

A Learned Approach to CDISC Specifications

Stephen M. Noga, Rho®, Chapel Hill, NC
Brandon Welch, Rho, Chapel Hill, NC
Jeff Abolafia, Rho, Chapel Hill, NC

Abstract

Ask statisticians and programmers to list their most enjoyable job tasks. You would have to search long and hard to find creating and entering SDTM or ADaM variable specifications on those lists. Yet those specifications and variable attributes, along with entering the accompanying documentation, are the backbone to the metadata necessary to describe the CDISC datasets. Additionally, the cost to produce these metadata is not cheap and increases as the amount of required metadata grows each year. The beauty of end-to-end, harmonized standards such as SDTM and ADaM is that variability in the metadata greatly decreases and computers become better at providing a data-driven solution. At Rho, we are working on a solution that revolves around the computer learning from our metadata repository. These metadata guide which specifications are appropriate for a dataset variable, and with the user's approval, the system will enter those values into the current project's metadata.

Synopsis

At Rho (a full-service CRO) designing, creating, and validating the SDTM and ADaM datasets are an everyday task for the statisticians and programmers. While creating and validating these datasets fall within their area of expertise, the design phase requires many tedious, manual tasks that do not require statistician or programmer skills. The manual tasks include: mapping the data (clinical -> SDTM -> ADaM), creating the specifications for each mapped variable (format, length, label, controlled terminology, origin, etc.), and data entry of all metadata. Because clinical trial data can be complex, the creation and validation of the SDTM and ADaM datasets take a considerable amount of statistician and programmer time. Additionally, the amount of metadata required for CDISC datasets continues to increase. As a result, the time and cost needed for these tasks has increased concomitantly. Despite using our metadata repository as a starting point, the trial becomes more expensive and time-consuming with the addition of the manual design phase tasks.

Rho believes it can greatly reduce the amount of manual work in the design phase based on three factors: use of data standards for a clinical trial is now the norm instead of exception; wealth of historical metadata in its metadata repository; and advancement of natural language processing. By eliminating the bulk of the subjective and menial tasks, Rho can create a better day-to-day work experience for its statisticians and programmers, produce more consistent metadata across sponsors, reduce the cost of producing the metadata, and deliver overall higher quality metadata and CDISC datasets. The system Rho is developing looks to accomplish all three of those goals.

When initiating this effort, data mining was easily performed via traditional programming methods (i.e. frequency calculation). However, the incorporation of the natural language algorithms added confidence to the results we saw in the initial phase of the project. This was especially true for the datasets in which the variable definitions were more subjective in nature. As the metadata repository grows on a daily basis, it increases our ability to more accurately predict the correct metadata for a variable.

In the following pages, we explain the steps taken thus far in developing an automated system for producing clinical study metadata and propose an end point to our process.

Approaches (old-school: rules)

As stated above, creating metadata for CDISC datasets is an expensive, necessary, yet not highly coveted task. In a fantasy world, we would be able to push a button and the computer would convert a study's clinical data to SDTM datasets, and then, into ADaM datasets. In reality, we started with the goal of reducing the manual effort by at least half by providing the means for manual effort to decrease as more study metadata was entered into the metadata repository. We began our automation project with the following goals:

1) Breaking down our metadata processes and products into individual steps and asking a number of questions: on average, how much time does a step take to complete; how much manual effort is involved in a step; how burdensome is it to complete a step; how unique is the step to each study
2) Examining the steps in terms of automation, brainstorming ways we could get the computer to do the manual work on a limited budget of time and money without having to hire a software development firm
3) Insuring that any improvements that would require coding and/or technology must integrate with our current metadata system
4) Choosing which converted metadata product would be our goal, SDTM (going from clinical data -> SDTM) or ADaM (going from SDTM -> ADaM)

The team concluded that the converted dataset variable definition would be the key to reducing manual effort. If an algorithmic prediction of the variable definition could exceed a defined percentage, then most metadata for that variable could be automatically generated. The team also decided that its efforts would concentrate on going from SDTM -> ADaM. As a CRO, the clinical data (data coming from data collection systems) which is the source of SDTM, comes in a variety of flavors and formats. On the other hand, the SDTM data we create or receive is relatively standard. We foresaw an easier path to initial success using more standardized input. If successful, the system could then be applied to the conversion of clinical data -> SDTM.

Rho has been storing study metadata in their metadata repository for over a decade providing a wealth of relevant data to use. The team limited the pool of studies to those that included conversions to ADaM in recent years. This allowed mining definitions that fit within the newer ADaM versions rather than including all previous versions of ADaM.

Mining

In general, datasets needed to support safety analyses are more uniform across studies and therapeutic areas than datasets required to support efficacy analyses. Therefore, safety datasets tend to have a lower degree of variability than their efficacy counterparts. For example, looking at the variables in dataset ADAE across multiple studies and therapeutic areas, the majority of variables in each ADAE dataset also exist in all other ADAE datasets. Consequently, safety datasets were the starting point for our mining expedition.

Looking at the initial dump of ADAE variable definition frequencies verified our assumption. For many of the variable names that began with ‘AE’, the definitions that had the highest frequency counts were straight pulls from like-named variables in SDTM datasets.

ADAE Analysis Variable	Analysis Variable Definition
AESOCCD	AE.AESOCCD
AESOC	AE.AESOC
AESHOSP	AE.AESHOSP
AESMIE	AE.AESMIE
AEDECOD	AE.AEDECOD

However, the definitions for these straight-pull variables, while similar, could come in any number of varieties.

ADAE Analysis Variable	Analysis Variable Definition
AEDECOD	AE.AEDECOD
	=AE.AEDECOD
	SDTM.AE.AEDECOD
	AE.AEDECOD if missing the set to 'Not Coded'
	RAW.ADAE.AEDECOD
	[Study A] ADAERCODE.PREFERREDNAME [Study B, Study C] ADAE.AEDECOD

As can be seen from the definitions above, sometimes what seemed like a straight-pull variable definition required more than just a dataset and variable name. Yet by applying rules created from these deviations, we were able to construct ‘buckets’ with which to categorize variable definitions. Examples of rules for inclusion in the straight-pull bucket would be: does the analysis variable name appear in the variable definition; no more than two ‘.’s in a definition; and length of the variable definition <= 26 characters. Placing a variable into a bucket was not done with certainty. Rather, the rules we created gave us a reasonable chance that a variable belonged to a specific category

type. These buckets would form the basis for computers suggesting the most likely definitions for an analysis dataset variable.

The straight-pull variable bucket showed that 31% of the ADaM datasets in our pool had straight-pull definitions that accounted for at least 40%-75% of their definitions. This group of datasets was either safety-related or in other areas such as inclusion/exclusion as opposed to efficacy. Further investigation of a dataset such as ADCM (41% of all definitions were straight-pull variables) found that 46% of variables found in most contributing studies had a greater than 65% chance of being a straight-pull variable.

When trying to find patterns in the variable definitions that were prevalent enough to warrant creating a bucket, we realized that almost all variable definitions grouped nicely into one of these categories: CONSTANT (e.g. 'AE', 13.1); COPY (straight copy from a same-named variable in a SDTM or clinical dataset); RENAME (very similar to COPY except that the ADaM variable name will not be the same as the input variable name); CONVERSION (convert the input variable in some fashion – character to numeric, numeric to character, apply date format, etc.); CODE (some custom programming logic needed). The straight-pull bucket held both COPY and RENAME definitions while the conversion bucket contained mostly CONVERSION definitions with a minimum of CODE definitions. The conversion bucket did not contain as many variables as the straight-pull bucket. Some variables that did get placed in it (such as AENDT and ASTDT) those variables were true conversions 80% of the time.

The above description of mining the Rho metadata is what programmers have always done - identify patterns and write code to achieve a result. It allowed us to place, with some degree of confidence, a variable in a category to make a more accurate prediction of what that variable definition should be, if possible. While our success rate in knowing what a definition should be for a variable varied by dataset and within the dataset itself, we felt we were heading in the right direction. However, there were still many gaps left to overcome especially with the variables in the large CODE category. This is the point where traditional programming approaches ceased to be effective and we explored whether machine learning programming could help us.

Approaches (new-school: Natural Language Processing (NLP))

Natural language processing techniques are not new, but their importance cannot be understated. This is especially true with the rise of machine learning and artificial intelligence (AI). For example, our cell phones and similar devices use NLP every day for tasks such as text-to-speech. There are a number of techniques available, but we are somewhat limited with our historical data. Machine learning algorithms require large amounts of data. Therefore, we are limited to more 'descriptive' NLP methods as opposed to those that require training/test data sets for prediction and/or classification.

The subset of studies which we used for the automation project totaled almost 100,000 observations with ADaM specifications. However, even with these data we can only use elementary NLP techniques. The hope is the user will have a choice of the top three definitions rather than starting from scratch. For variables with subjective definitions, we test cosine similarity and sentence scoring. We programmed all NLP algorithms in python 3.6.5.

Text pre-processing

Naturally each person writes specifications differently. In addition, definitions are variant across different sponsors and projects. Before employing our NLP techniques, we transform the specifications into text that more closely resembles the human language. This transformation helps us build a sound Term Frequency–Inverse Document Frequency (TF-IDF) matrix when we apply cosine similarity. The TF-IDF matrix is the data set we create to transform our definition text data to numeric data. In the simplest sense, this matrix assigns a value to each word in the variable definition. Each value represents the word's importance across the entire corpus of programming definitions. These numeric data are then fed into the NLP algorithm (cosine similarity). Before we construct the matrix, we use regular expressions and the inflect module to clean our data.

Example 1:

Some statisticians may use SAS code in their specifications. For example, here is an entry for defining the treatment emergent flag (TRTEMFL) in ADAE:

```
"Y" if index(AEINJTM,"After")>0
```

We apply back-referencing with the *re* module to transform a SAS function (INDEX) into human speech. In python,

```
import re
def prep(text):
  #--------------translate SAS functions to words----------------
  #index
  repl = re.sub(r'([iI][nN][dD][eE][xX])\((\w+),\s?"(\w+)"\)>?0?', r'\2 contains \3',
  text)
  #remove punctuation
  repl = re.sub(r'[^\w\s]', ' ', repl)
  #make lower case
  repl = repl.lower()
  #recode y to yes
  repl = re.sub(r'\sy\s', ' yes ', repl)
  #remove all multi-blanks
  repl =  re.sub(r'\s+', ' ', repl)
  return repl
```

Applying this function to our data frame gives us the text:

```
yes if aeinjtm contains after
```

Example 2:

We also need to convert any numbers in our definition to words. For example:

```
= 'Y' if ADSL.TRTSDT <= ASTDT <= ADSL.TRTEDT + 30 if ADAE.RELGR1N = 0
= 'Y' if ADSL.TRTSDT <= ASTDT if ADAE.RELGR1N = 1
```

We need 30 to be 'thirty'. We use the *inflect* module to achieve this. In python,

```
import inflect
p = inflect.engine()
def num2wrd(row):
    newrow = []
    for i in range(len(row)):
        if row[i].isdigit():
            row[i] = p.number_to_words(row[i])
    newrow = ' '.join(row)
    return newrow
```

Applying this function to our data frame, we get:

```
= 'Y' if ADSL.TRTSDT <= ASTDT <= ADSL.TRTEDT + thirty if ADAE.RELGR1N = zero
= 'Y' if ADSL.TRTSDT <= ASTDT if ADAE.RELGR1N = one
```

After further use of regular expressions, we get the final definition:

```
if trtsdt less than or equal to astdt less than or equal to trtedt plus thirty if
relgr1n equal to zero
equal to yes if trtsdt less than or equal to astdt if relgr1n equal to one
```

We are now ready to apply the NLP techniques.

Cosine Similarity

After we pre-process our data, we create a TF-IDF matrix. This matrix is what we use in our analysis. Cosine similarity is a technique for calculating the similarity between documents (programming definitions in our case). Mathematically, the sentence/definitions are represented by a vector in vector space. The cosine of the angle between the vectors gives us the measure of similarity.

For example, suppose we only have two definitions in our historical database (again using TRTEMFL as our test variable):

```
if ae_active.aedoseyn = 'Yes' then set to 'Y'. Otherwise set to missing

Y if ADAE.EPOCH='ON-TREATMENT' N otherwise
```

After preprocessing:

```
if aedoseyn equal to yes then set to yes otherwise set to missing

yes if epoch equal to on treatment no otherwise
```

TF-IDF Matrix:

Definition	aedoseyn	epoch	equal	if	missing	No	on	otherwise	…
if aedoseyn equal to yes then set to yes otherwise set to missing	0.2573	0	0.1831	0.1831	0.2573	0	0	0.1831	…
yes if epoch equal to on treatment no otherwise	0	0.3913	0.2784	0.2784	0	0.3912	0.3913	0.2784	…

Using these data, we calculate a measure of similarity between sentences. For this example, the cosine similarity is 0.36560003.

Another example helps illustrate what occurs when the definitions are more similar to one another.

```
 "Y" if [ADSL.TRTSDT < ADAE.AESDT <= ADSL.SPAEDT] or [AEONSET = 'ONSET AFTER FIRST DOSE
OF STUDY DRUG'] missing otherwise

'Y' if DERIVE.ADSL.TRTSDT <= AESDT or nmiss(AESDT) ='N' otherwise ='' if ANYAE='No'
```

After preprocessing:

```
if trtsdt less than aesdt less than or equal to spaedt or aeonset equal to onset after
first dose of study drug missing otherwise

if trtsdt less than or equal to aesdt or aesdt not missing equal to no otherwise equal
to missing if anyae equal to no
```

Abbev. Def.	aeonset	aesdt	after	anyae	dose	drug	equal	first	…
if trtsdt less than aesdt less than or equal to spaedt or aeonset equal…	0.2149	0.1529	0.2149	0	0.2149	0.2149	0.3058	0.2149	…
if trtsdt less than or equal to aesdt or aesdt not missing …	0	0.2503	0	0.1759	0	0	0.5006	0	…

Cosine similarity is 0.61228351. Note how this value is higher (more similar) that the previous example since these definitions are more similar.

As we compute the similarity values between all programming definitions in the historical metadata, we can rank those and consider the ones with the highest cosine similarity values. Using all historical data for ADAE.TRTEMFL, we show the three highest similarity values for the first three definitions, then sort by similarity values in descending order:

Abbrev. Def.	Similarity value1	Similarity value1	Similarity value1	Record (1st comparison)	Record (2nd comparison)	Record (3rd comparison)
`if astdt greater than or equal to trtsdt and trtsdt greater than …`	0.950081	0.865946	0.834122	28	50	45
`if astdt greater than or equal to trtsdt and trtsdt greater …`	0.950081	0.856613	0.854315	50	28	56
`if astdt greater than or equal to trtsdt and trtsdt greater than … missing`	0.931718	0.869696	0.865946	45	56	37
…						

The top row's highest similarity values across all compared definitions are 0.950081, 0.865946, 0.834122. The observations contributing these values are 28, 50, and 45 in the data set. The similarity values for the next two most similar definitions are also displayed. Given this ordered list, the user may choose any of these as a candidate for treatment emergence definition. Full text for record one:

"= 'Y' if DERIVE.ADAE.ASTDT >= DERIVE.ADSL.TRTSDT and DERIVE.ADSL.TRTSDT > .z
= 'Y' if DERIVE.ADAE.ASTDT is missing
= missing otherwise"

This is a sound definition for treatment emergence that theoretically applies across any study using CDISC standards.

Sentence Scoring

The second technique we test is a text summarization analysis using scores. It is achieved by assigning weights to words in the entire definition corpus for a particular variable. The weights are calculated by dividing the number of occurrences of each word by the maximum number of occurrences of any word. We remove all stop-words prior to calculating the frequencies. Stop-words are words that add no additional meaning (i.e. 'the', 'and', etc.) We then sum the weights by definition which provides a score. The sentences with the highest scores are considered most meaningful.

Example 3: using TRTEMFL as the variable from the examples above, assume our entire corpus consists of the following:

Definition 1:
`if aedoseyn equal to yes then set to yes otherwise set to missing`

Definition 2:

```
yes if epoch equal to on treatment no otherwise
```

The words in tabular form:

Word	Count	Weight
aedoseyn	1	1/3 = 0.3333
Epoch	1	1/3 = 0.3333
Equal	2	2/3 = 0.6667
Missing	1	1/3 = 0.3333
otherwise	2	2/3 = 0.6667
Set	2	2/3 = 0.6667
treatment	1	1/3 = 0.3333
Yes	3	3/3 = 1

Summing the weights, we get the following:

Definition 1:

if	aedoseyn	equal	to	yes	Then	set	to	yes	otherwise	Set	to	missing	Score = Sum of Weights
0	0.3333	0.6667	0	1	0	0.6667	0	1	0.6667	0.6667	0	0.3333	5.33

Definition 2:

yes	If	epoch	equal	To	on	treatment	no	otherwise	Score = Sum of Weights
1	0	0.3333	0.6667	0	0	0.3333	0	0.6667	3

Using this method and assuming this is our entire corpus, the first definition would be chosen as the better candidate for defining treatment emergence.

Using the entire database of ADAE.TRTEMFL definitions as input, the top five choices include:

Definition	Score
if trtsdt less than or equal to aesdt less than or equal to trtedt or trtsdt less than or equal to aesdt and trtedt is missing missing otherwise	12.7511
if astdt greater than or equal to trtsdt and trtsdt greater than missing equal to yes if astdt is missing equal to missing otherwise	11.0034
for vaccinees equal to yes if astdt greater than or equal to trtsdt equal to yes if astdt is missing for hhcs equal to missing	9.6728
if trtsdt less than or equal to aesdt or aesdt not missing equal to no otherwise equal to missing if anyae equal to no	9.6687
if trtsdt less than aesdt less than or equal to spaedt or aeonset equal to onset after first dose of study drug missing otherwise	9.2586

Note that the second definition was flagged as the top definition by cosine similarity.

Given these two approaches and examples, NLP techniques appear to enhance the traditional frequency approach. In addition to the techniques presented, we also plan to explore other analyses – possibly clustering techniques.

Next Steps

Identifying patterns, designing rules, and developing new techniques gave us confidence that it was possible to suggest the most likely definitions for variables contained in the most frequently programmed ADaM datasets. However, we needed to see if this approach had practical applications. We compared the suggested definition choices to actual definitions of ADaM datasets from studies recently programmed at Rho.

The suggested list of definitions for each variable was ordered from most likely to least likely. We then linked the programmed ADaM variable definition to the order position of the matched suggestion definition list and categorized the linkage position as 1 (most likely), 2 (second most likely), or 3+ (third or greater most likely linkage position).

Results from the Rules Approach vs. Actual Programmed Definition

- Study 1
  - 62% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 2
  - 60% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 3
  - 85% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot.

Results from the cosine similarity approach

- Study 1
  - 85% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 2
  - 62% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 3
  - 62% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot.

Results from the sentence scoring approach

- Study 1
  - 62% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 2
  - 70% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot
- Study 3
  - 62% of ADaM datasets had >= 50% of their definitions in the most likely suggested spot.

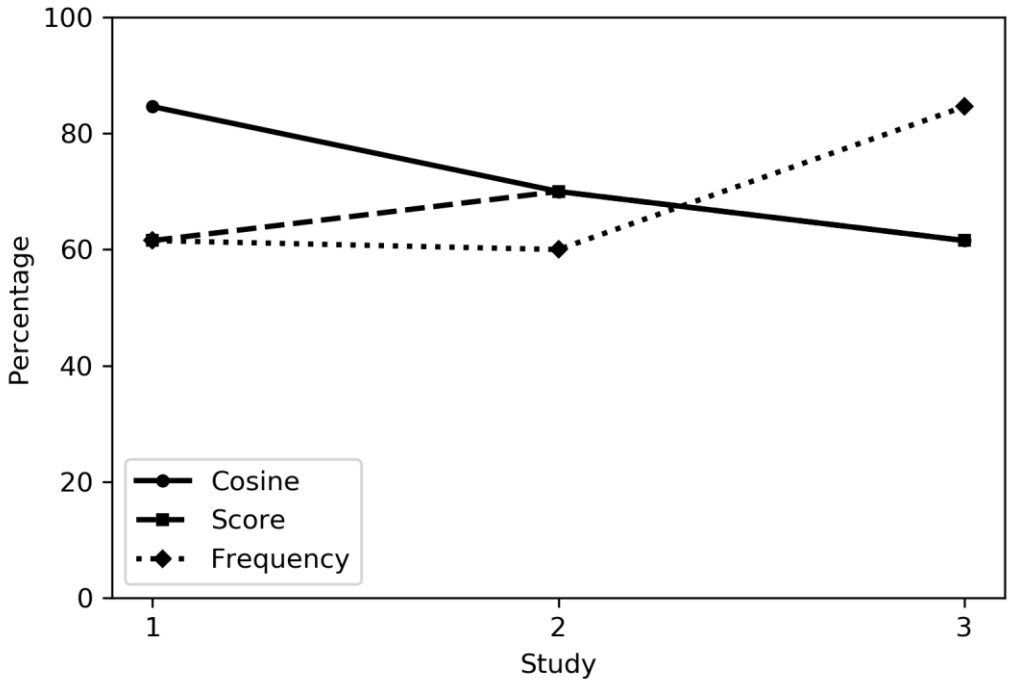
These results indicate that our automation project could succeed. The challenge for us now is multifaceted:

- We've demonstrated individual approaches can succeed, but now algorithms combining these approaches must be developed to provide the best suggested definition for a variable.
- Develop a strategy for incorporating user definition selection utilizing the suggestion algorithms
- Design, test, and build
  - Real-time processing of algorithms against current metadata for suggested definitions that is not frustratingly time-consuming
  - Redesign our interface for user definition choice that not only captures the selection, but also stores the associated metadata based on that definition

As we stated early on in this paper, we are not software developers. Maintaining a realistic approach to these next steps is paramount. Rho's current metadata system already allows a user to begin construction of a new study's metadata by selecting any or all metadata pieces from any study in its library. This can initially save time for a user. Any developments must integrate into this system. Currently, results from the above described approaches are available in a number of file formats (PDF, XLSX, etc.) and allow a user to search and copy results. This provides additional time savings but is not our long-term goal.

Conclusion

Taking full advantage of standards to reduce the mundane, manual efforts of dataset construction has been a goal at Rho for many years. Our company believes that repetitive tasks assigned to computers frees employees to undertake more intellectual endeavors. We were pleased that both traditional and newer programming techniques showed value in this automation project. Graphing the tabular results from above illustrates some challenges as we move forward with this project.



The cosine similarity approach holds a steady advantage over the traditional approach in studies 1 and 2. However, in study 3 the traditional approach performed markedly better than the two NLP techniques. These results illustrate that even though we have more advanced techniques at our disposal, we still cannot rely on a 1-size-fits-all model. The best solution is to apply the best technique for each situation. Study 3 was the oldest and had the most inconsistent programming definitions. If a programmer or statistician was faced with a situation like this, the user would have more than one tool option.

The NLP techniques tested are rudimentary compared to the plethora of other techniques available. Given we are novice data scientists ourselves, we are just scratching the surface of what is possible. For example, unsupervised clustering (such as k-means) techniques may offer more possibilities.

One of the dangers of embracing new technologies is ignoring the known benefits of currently available technology. Our experimentation with machine learning was done with approaches that looked to overcome the obvious pitfall of having too little data to provide meaningful results. We look forward to continuing our efforts on this project.

Contact Information

Steve Noga stephen_noga@rhoworld.com
Brandon Welch brandon_welch@rhoworld.com
Jeff Abolafia jeff_abolafia@rhoworld.com