PharmaSUG 2023 - Paper SD-232

Tired of Manual Language Translation? Give it a REST!

Shawn Hopkins and Matthew Ness, Seagen Inc.

ABSTRACT

Working in a global environment or partnership settings, we are bound to encounter situations where we need to translate content from one language to another in order to analyze clinical trial data. With respect to SAS data sets with content in a foreign language, the FDA requires a verified English translation, which is often performed by a third party. However, this may take considerable time and could squeeze programming development too much to meet upcoming timelines. With that in mind, is there a way to get a "good-enough" data set translation in place quickly and efficiently, so programming can start development long before the official translation enters the scene?

In this paper we'll share an innovative, fully automated approach to informal translation of SAS data sets received in Extended Unix Code simplified Chinese characters (EUC-CN) into UTF-8 encoding in English. Using Seagen's Microsoft Azure subscription with Text Translation service (free to try), we translated both the variable labels and the content of character variables in each data set using the cloud-based REST API via PROC HTTP. This setup can immediately support translations to and from over 100 different languages, and it has been tremendously helpful to get an early start on analysis programming to comfortably meet our timelines well before third-party translations are available.

Keywords: Azure Cognitive Services, Web API, Text Translation, PROC HTTP

INTRODUCTION

A recent partnership with an organization in Asia resulted in the receipt of SAS data sets containing multi-byte encoded character variables in simplified Chinese. Ideally, a third party would provide English translated data sets, however, that process would likely take months and our programming team needed to work with the data immediately.

This paper describes how we created English translated data sets by using SAS in combination with Microsoft Azure Translation service. We will also describe challenges related to data set encoding and how we overcame them.

ENCODING CHALLENGES

The data sets we received from our partner contained Chinese characters and were encoded in UTF-8, which is a variable width, multibyte encoding. Unfortunately, session encoding in our SAS environment is set to WLATIN1, which is a single-byte character set (SBCS) that only supports ASCII/ANSI characters correctly. Therefore, users could not access the data programmatically nor could they view the contents of the data sets.

We were unable to modify the production server encoding as that would require re-validation of the entire SAS system. As an interim solution, IT created several virtual machines with a local installation of PC SAS configured to use UTF-8 encoding and language support for simplified Chinese characters. Even though this was only a development environment it was sufficient to enable us to work with the data in an unofficial capacity.

AZURE COGNITIVE SERVICES

Depending on a vendor to perform the official translation of the content required a significant amount of time. To get a version of the data that we could work with quickly, IT and Statistical Programming collaborated on a strategy using Azure Cognitive Services' Translator API.

Within Microsoft Azure's AI + Machine Learning Services, there is a Cognitive Services category, under which users can create a text translation service. Since patient data would be exposed, we implemented a containerized translation service in the company's Azure Cloud environment rather than utilizing a public Web API.

This cloud-based REST API can be called directly from SAS using PROC HTTP. The GET endpoint returns a JSON response that lists all the available languages, featuring over 100 supported languages. The POST endpoint accepts a JSON body as a collection of key-value pairs and returns a JSON response with values translated into the destination language.

Once the API is created, you will need the URL and some required information in the headers of the request. If you do not have access to the Azure portal, the administrator will need to provide you with the subscription key and region.

- Ocp-Apim-Subscription-Key: the translation service key from Azure portal
- Ocp-Apim-Subscription-Region: the region where the resource was created.
- Content-Type: **application/json** or **charset=UTF-8**.

An API client application like Postman makes it simple to test the implementation of the web services like Azure Translator.

The root location URL for translation service is specific to the organization and will need to be provided by the Azure administrator. There are several endpoints supported by the API with many options (see API Guide for comprehensive documentation). The two used for this solution were the languages and translate endpoints.

Action	Endpoint	Response	Example
GET	languages	Returns a list of supported languages for translation	/languages?api-version=3.0&scope=translation
POST	translate	Returns the translation of a text collection specified in the request body	/translate?api-version=3.0&to=en&from=Zh-hans

Table 1. Azure Cognitive Services Translator Endpoints

AVAILABLE LANGUAGES

The languages endpoint returns the list of supported languages available for translation. At the time of this paper, there were 111 available. See a table displaying available languages and an example of how to use that languages endpoint in GitHub.

TESTING THE SERVICE

To confirm the translation service is implemented and your authentication is working, test the endpoints using a client application. Postman is the easiest, but if it's not installed, you can use the cURL utility available at the command prompt.

1. Create a JSON file with a single key-value pair inside of an array. Save the file locally as helloworld.json.

   ```
   Example: [{"text":"this is only a test"}]
   ```

2. Open the CMD prompt from the same folder, and invoke the cURL utility and pass the required headers and parameters:

   ```
   CURL "your api root/translate?api-version=3.0&from=en&to=fr" -H
   "Content-type:application/json" -H "Ocp-Apim-Subscription-Key:your key"
   -H "Ocp-Apim-Subscription-Region:your region" -X POST -d
   @helloworld.json
   ```

3. If the request was successful, the translated value and the TO language code are returned as a JSON array in the response.

```
[{"translations":[{"text":"Esto es sólo una prueba","to":"es"}]}]
```

TRANSLATING A SAS DATA SET

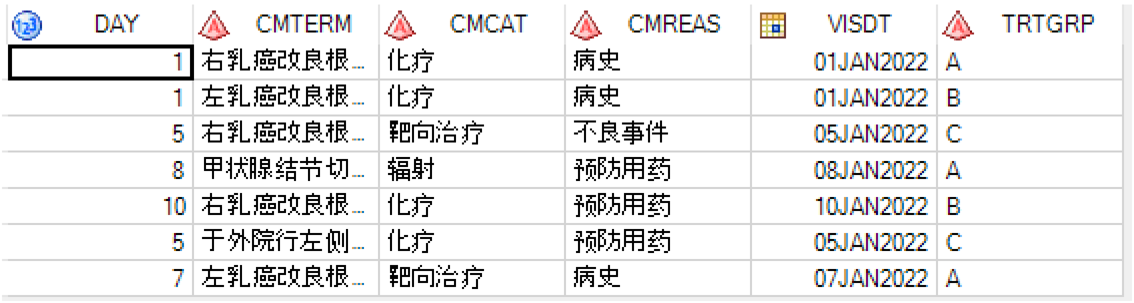
Using the same logic from previous example, we can call the API directly from SAS using PROC HTTP.

First assign the API root and required headers in macro variables. For demonstration, the arguments are assigned directly in the SAS program, securing these values is out of scope for this paper. The code for the following program is available in its entirety in GitHub.

```
%let api=[your API root];
%let key=[your subscription key];
%let region=[your Azure region e.g. westus2];
```

INPUT DATA SET

The source data sets contained a mix of character values in Simplified Chinese encoding and a variety of numeric variables including integers, floating point, and date values. The translation needed to convert the character values with the original variable names and retain the numeric values without modification.



DAY	CMTERM	CMCAT	CMREAS	VISDT	TRTGRP
1	右乳癌改良根...	化疗	病史	01JAN2022	A
1	左乳癌改良根...	化疗	病史	01JAN2022	B
5	右乳癌改良根...	靶向治疗	不良事件	05JAN2022	C
8	甲状腺结节切...	辐射	预防用药	08JAN2022	A
10	右乳癌改良根...	化疗	预防用药	10JAN2022	B
5	于外院行左侧...	化疗	预防用药	05JAN2022	C
7	左乳癌改良根...	靶向治疗	病史	07JAN2022	A

Table 2: Source Data for Translation

It is critical that the SAS environment encoding is UTF-8 and supports the multi-byte character set of the source language to allow the data set to be read.

CAPTURE THE METADATA FROM THE INPUT DATA SET

The metadata from the input data set should be captured so the translation can be automated, specifically a list of all the character variables, numeric variables, and the counts of each. This will enable the reconstruction of the data with the translated character variables and the unchanged numeric variables:

```
%*---------------------------------------------------------------------*
| Sort character variables first by name. Used to rename response data
| from translation API
*---------------------------------------------------------------------*;
proc contents data=&dsn. out=&dsn._md(keep=name type length)
              nodetails noprint;
run;
%*---------------------------------------------------------------------*
| Create mvar with all variables in sequence to maintain the original
| order of the variables within the dataset
*---------------------------------------------------------------------*;
proc sql noprint;
  select name into :original_order separated by ' '
```

```
    from &dsn._md
    order by varnum;
  quit;

  %*-----------------------------------------------------------------------*
  | Capture metadata for character and numeric variables in macro variables
  *-----------------------------------------------------------------------*;
  proc sort data=&dsn._md;
    by descending type  name;
  run;  data _null_;
    length cvars nvars $5000;
    retain cvars nvars;
    set &dsn._md end=eof;
    by descending type name;
    %*--------------------------------------------------------------------*
    | Build a list of character variables
    *--------------------------------------------------------------------*;
    if type=2 then do;
      cvars = catx(" ", cvars, name);
      ncvars+1;
    end;
    %*--------------------------------------------------------------------*
    | Build a list of numeric variables
    *--------------------------------------------------------------------*;
    else do;
      nvars = catx(" ",strip(nvars),strip(name));
      nnvars+1;
    end;
    if eof then do;
      call symputx("cvars",cvars);
      call symputx("nvars",nvars);
      call symputx("ncvars",ncvars);
      call symputx("nnvars",nnvars);
    end;
  run;
```

BUILD THE JSON FILE

A JSON file is created from the character variables in the input data set using the CVARS macro variable to write values to the JSON file, for every observation.

Create the JSON file in the WORK folder to dispose of the file upon successful disconnection from the workspace session.

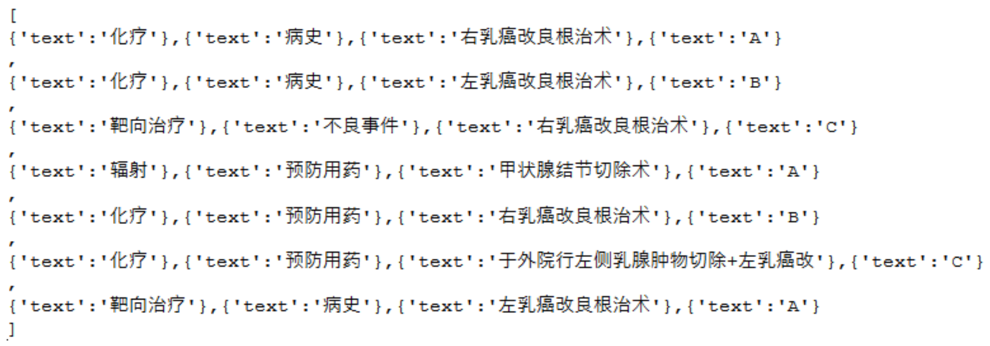
The DO loop writes the key-value pairs in the expected format as a JSON array. Since the number of character variables is known, it's easy to transpose back into observations once the text is translated.

```
  filename jbody "%sysfunc(pathname(work))\jbody.json" encoding="UTF-8";

  data _null_;
    length _line $2000;
    set &dsn.(keep=&cvars.) end=eof;
    file jbody;
    array _v[*] &cvars.;
    if _n_=1 then put '[';
    call missing(line);
    do i = 1 to dim(_v);
```

```
      if ^missing(_v[i]) then do;
         _line = catx(",",_line, "{'text':"||catx(_v[i],"'","'}"));
      end;
    end;
    put _line;
    if eof then put ']';
    else put ',';
  run;
```

Figure 1 shows an example the body of the JSON file.



```
[
{'text':'化疗'},{'text':'病史'},{'text':'右乳癌改良根治术'},{'text':'A'}
,
{'text':'化疗'},{'text':'病史'},{'text':'左乳癌改良根治术'},{'text':'B'}
,
{'text':'靶向治疗'},{'text':'不良事件'},{'text':'右乳癌改良根治术'},{'text':'C'}
,
{'text':'辐射'},{'text':'预防用药'},{'text':'甲状腺结节切除术'},{'text':'A'}
,
{'text':'化疗'},{'text':'预防用药'},{'text':'右乳癌改良根治术'},{'text':'B'}
,
{'text':'化疗'},{'text':'预防用药'},{'text':'于外院行左侧乳腺肿物切除+左乳癌改'},{'text':'C'}
,
{'text':'靶向治疗'},{'text':'病史'},{'text':'左乳癌改良根治术'},{'text':'A'}
]
```

Figure 1. JSON Body File

CALLING THE TRANSLATION SERVICE

The translation service is called using PROC HTTP, passing the required headers and the JSON body file. The JSON response from the API is captured in a temporary fileref. The response is parsed into SAS data format using the JSON engine on the libname statement:

```
  proc http url="&api/translate?api-version=3.0%nrstr(&from)=&from_lang
                 %nrstr(&to)=&to_lang"
    METHOD="POST"
    AUTH_NEGOTIATE
    in=jbody
    ct="application/json"
    out=outjson;
    headers "accept"="application/json"
            'Ocp-Apim-Subscription-Key'="&key"
            'Ocp-Apim-Subscription-Region'="&region"
            'content-type'='application/json';
  run;
  libname jsn JSON fileref=outjson;
```

ASSEMBLING THE FINAL DATA SET

The data set created from the response is normalized to just a single text variable. Group the observations by counting the number of character variables and incrementing the grouping variable once that number is exceeded. The data can then be transposed by that grouping variable, restoring the original record structure.

```
  data &dsn._trn;
    retain id 1;
    set jsn.translations;
    counter+1;
    if counter>&ncvars. then do;
      id+1;
      counter=1;
    end;
  run;

  proc transpose data=&dsn._trn out=&dsn._trn_t(drop=_name_) prefix=_var;
    by id;
    var text;
  run;
```

The original variable names are restored from the CVARS macro variable. Since the translated value may be larger than the original string, it is also necessary to identify the longest string for each new variable from the grouped data set.

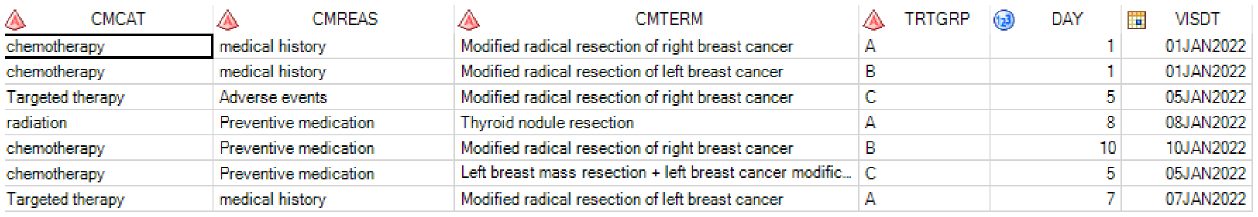
```
  proc sql noprint;
    select max(klength(text)) into :len_cvars separated by ' '
    from &dsn._trn
    group by counter;
  quit;
```

The restored data set is merged with the original data set, keeping only the unmodified numeric variables and a join variable. The final data set has the translated value text values in corresponding variables with the original numeric variables. The retain statement maintains the original order of the variables.

```
  data &dsn._new(drop=_var: id i);
    retain &original_order.;
    %do i = 1 %to &ncvars.;
      length %scan(&cvars,&i,%str( )) $%scan(&len_cvars.,&i.%str( ));
    %end;

    merge &dsn._trn_t
          &dsn._in(keep=id &nvars.);
    by id;
    array cvars[*] &cvars.;
    array vars[*] _var:;
    do i = 1 to dim(vars);
      cvars[i]=vars[i];
    end;
  run;
```

The final, translated data set is shown in Table 3.



CMCAT	CMREAS	CMTERM	TRTGRP	DAY	VISDT
chemotherapy	medical history	Modified radical resection of right breast cancer	A	1	01JAN2022
chemotherapy	medical history	Modified radical resection of left breast cancer	B	1	01JAN2022
Targeted therapy	Adverse events	Modified radical resection of right breast cancer	C	5	05JAN2022
radiation	Preventive medication	Thyroid nodule resection	A	8	08JAN2022
chemotherapy	Preventive medication	Modified radical resection of right breast cancer	B	10	10JAN2022
chemotherapy	Preventive medication	Left breast mass resection + left breast cancer modific...	C	5	05JAN2022
Targeted therapy	medical history	Modified radical resection of left breast cancer	A	7	07JAN2022

Table 3: Translated Data set

CONCLUSION

Using SAS in combination with Azure Cognitive Services Translator to translate the text enabled the study team to begin working with the data well in advance of obtaining the officially translated data sets. Some content, like adverse event verbatim terms, didn't translate perfectly, but these peculiarities were manageable by the study team.

While any data sets or output generated based on translated text obtained via this pathway are not considered officially translated and hence should not be used in official reports, publications, or submissions, this setup is immensely helpful to get a lot of the programming development and testing work out of the way which avoids putting delivery timelines at risk. Once the officially translated data sets are available, the team simply switches off the Azure-based translation pathway and uses the official translations instead!

REFERENCES

Microsoft "Azure Cognitive Services" Available at https://azure.microsoft.com/en-us/products/cognitive-services

CONTACT INFORMATION

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

Shawn Hopkins
Seagen Inc.
shopkins@seagen.com

Matthew Ness
Seagen Inc.
mness@seagen.com