Paper PO-03

Using SAS ® to Ease the Proofing of Messy Text

Richard W. La Valley, Strategic Technology Solutions, Inc., Herndon, VA

Nat Wooding, Independent Consultant, Midlothian, VA

ABSTRACT

The SAS Global Users Group has had a recent initiative to scan and OCR the older paper proceedings from the SAS Users' Group International (SUGI) conference (SAS.One to SUGI 21) and make these available in machine readable form on sasCommunity.org. These are now available for SAS Users.

Part of this project involved digitizing the attendance lists for use for future planning and analysis for the SAS Global Forum. The proofing of these lists proved to be a challenge due to misinterpretations in the optical character recognition process (OCR). The misinterpretations were due both to very minor printing variations in the text as well as the difficulty that the OCR software had in recognizing seeming clearly printed text. As an example, lower case ells were often rendered as upper case "I"s so that the name Bill was seen as BiII or Bi II.

Methods used in recognizing misspellings included viewing the text with Microsoft Word ® so that the spell check would flag problems and also SAS routines were written that pointed out some generic types of errors such as misspelled state names. This paper will provide a overview of the entire process and specifically detail the SAS routines which were used to get the data ready for analysis. In addition, it illustrates the use of various SAS string handling functions to parse, identify, and alter text.

Keywords: OCR Errors**,** OCR Correction**,** String Handling Functions, Proc Spell

INTRODUCTION

Recently, the older proceedings from the SAS Users' Group International (SUGI) were converted into digital form and put up on sasCommunity.org for research and retrieval. These proceedings provide a historical reference for the SAS Global Users Group community and provide vital information for planning of future SAS Global Forums. One of the key pieces of information of interest is the attendance lists which were provided in the older paper proceedings. These historical attendance lists will be analyzed for patterns of attendance, geographic analysis of attendance and other items of interest for improving the SAS Global Forum.

The process of scanning or digitizing these paper proceedings created a series of images of each page, into pixels in computer memory which are then converted into text using an optical character recognition (OCR) program and delivered in PDF format files that could be saved as text. This process is error prone.

Common Problems with most OCR programs encountered in this project

OCR technology commonly has trouble distinguishing between the similar characters. Some examples are:

- The digit '1' (one), the lowercase letter 'l' (ell), and the uppercase letter 'I' (i). Note that in some fonts the number one may look like I (like a small capital letter 'i').

- The digit '0' (zero), the uppercase letter 'O', and in some fonts the uppercase “D”
- Dashes & hyphens: OCR'd text often has only one hyphen for an em-dash that should have two
- Parentheses ( ) and curly braces { }.
- The character “,” being confused with “~” or “-“ or “."
- Capital ‘N’ rendered as an ‘H’.

Another common OCR issue is misrecognition of characters. This misrecognition can create a word that appears to be correct at first glance, but is actually misspelled and are changed to a different but otherwise valid word that does not match what is in the page image. Possibly the most common example of this type is "and" being OCR'd as "arid." Other examples are "eve" for "eye", "Torn" for "Tom", "train" for "tram".

Another class of problems encountered are words that appears to be correct but additional spaces are put into the word such as “co rp at ion”. This type is harder to spot and definitely the use of Proc Eyeball or reviewing the document in Microsoft Word and looking at the portions of words highlighted.

Some of these problems can only be corrected via manual inspection and some can be found and corrected using automated techniques. Some of these automated techniques will be discussed in this paper.

Cleaning the Data

This process consisted of first using the “Save As Text” feature of Adobe Acrobat Reader ® to create a text copy of the file. We then edited this file prior to conversion to a SAS data set. While a word processor such as MS Word could be useful when proofing text, the attendance list was so diverse that a saved text page appeared to be bleeding to death when opened in Word. We chose to use Microsoft’s Notepad ® for our editing.

One aspect of visually comparing the PDF with the text file that caused a little confusion at the start was that in the PDF, we could see a word but the associated text might not match it due to its not having been translated properly. This caused problems during the disambiguation work when we needed to search for specific names in different years and some names were not properly rendered so the Adobe Acrobat Reader ® could not find the strings that we were seeking..

```
Abba□tt, Doug D.
AT&Sf Railway Co.
Chicago, IL
Abde I a I 1, Mona
Ford Motor Company
Dearborn, MI
Abe I, Jane R.
Abbott Laboratories
Abbott Park, I L
Adams, Beverly V.
UTHSCD
Da lias, TX
Adams, Bruce
Digital Equip Corp.
Chapel Hi II, HC
```

Figure 1. A Sample Set of Names and Addresses Which Illustrates the Scanned Text after Conversion to a Text File.

Figure 1 shows some of the complexities of the text with which we were working. Of the four names and addresses, all have at least one problem. Some of the errors such as “Da lias” or “Chapel Hi II”, once recognized, can be safely corrected with the Windows Find/Replace feature available in many text editors since it is unlikely that they are part of any other strings. However, most of the other errors cannot be so readily fixed since so many different words could have been corrupted. For example, “I L” could have been legitimate initials but printed without periods or they could have been, say, a pair of ells within a word or even a Roman numeral. In the case of the ells within a word, if we changed them to two lower case ells, we would still have to recognize the word, determine the correct spelling and then decide whether it would be safe to do a global replace or fix the errors one at a time.

Had we been faced with a smaller set of input lines, a direct application of “Proc Eyeball” where we simply looked for problems and did a manual find/replace fix might have been acceptable. Given that there were some 16,000 registrants over the first 17 years (this was the subset of the data that we first worked on), we were faced with visually scanning nearly 48,000 lines of text. Moreover some of the errors were subtle and hard to recognize, particularly after an hour or two of work, so it was quite easy to overlook mistakes. Hence, a visual one-at-a-time find/replace approach was a daunting task.

SAS Enters the Process

The examples that will be shown illustrate the processes that we used in cleaning our data. Someone dealing with a different set of input files would probably find other misspellings but the approach for fixing these would be the same.

Proc Spell

Early in the effort, we considered using the no longer-documented (in SAS Institute documentation) Proc Spell and it did have some use (The documentation from an early SAS manual is now available at sasCommunity.org (http://www.sascommunity.org/wiki/Proc_spell) and Barbara Okerson gives examples of using it in data cleansing (Okerson, 2007).) The standard output shows each “word” listed in the order that it is first encountered in the input file and also shows the line locations of all instances of this string. Here, “word” simply refers to a group of letters bounded by one or more spaces

```
Hughes          6    36, 415, 3462, 8209, 9948, 11458
  Ai           25     36, 415, 582, 1276, 1277, 2019,
                      2474, 2660, 2845, 3462, 3822,
                      4657, 4855, 5315, 6286, 6649,
                      6773, 6805, 8019, 8209, 9569,
                      9948, 11137, 11458, 11509
  rcraft        7     36, 415, 2474, 6034, 8209,
                      9948, 11458
```

Figure 2. Sample Output from Proc Spell Which Shows 3 Strings, Their Frequency, and Line Locations.

Figure2 was taken from a portion of Proc Spell output and shows the results when the word “Aircraft” was rendered

as the “words” “Ai” and “rcraft”. To complicate matters, other words starting ‘Ai” also were split.
Upon recognizing a problem such as the one shown here, we could either do a find/replace operation in our text file or we could use the line numbers shown by Proc Spell to try to locate and manually fix the problem. However, note that only six of the instances of “Ai” were associated with “Hughes Aircraft” whereas there were 18 others that needed to be found and investigated and one of these was evidently not the word “aircraft”. Unfortunately, Proc Spell does not tell us the strings associated with these 18 instances of “Ai” so it would be helpful to have SAS list them for us.

In general, Proc Spell helped us identify word fragments and showed their frequency but was not as robust a tool as we wanted. What we needed was the ability to find types of word fragments and display them along with the line in which they were found. Here, some basic string handling functions and SAS Data Step code came into use.

String Checking With SAS

Since we were using a text editor that does not show the line that was currently being edited, it was helpful to add a line number and create a copy of the file with this number prepended to each line. The SAS code

```
Filename Spell 'C:\Proofing Paper\For Paper SUGI 12 Registration.txt';
Filename SpellOut 'C:\Proofing Paper\For Paper SUGI 12 RegistrationLined.txt';
Data _NULL_;
Infile Spell;
Line + 1;
File SpellOut;
Input;
Put Line @10 _INFILE_;
Run;
```

simply makes a copy of the text file and adds line numbers which could be found using Cntl F.

Working with the problems shown in Figure 1, a short SAS job finds the instances of the word “Ai” and lists the line number as well as the line. This and similar jobs used SAS string handling functions to search for suspicious text and display the line where it was to be found. Using the following code, we can see the 25 lines that have “Ai” as a word as shown in Figure 2.

```
Filename SpellOut 'C:\Proofing Paper\For Paper SUGI 12 RegistrationLined.txt';
Data Find;
Infile SpellOut firstobs = 7  ;
Input  ;
lINE = Scan( _Infile_ , 1 );
String = Substr( _Infile_ , 10 );
String = Compbl( String);* remove extra blanks;
Do I = 1 to 15;
     Word = Scan( String  , I , ' ' );
     If Word = 'Ai' then output;
     If Word = '' then return;
End;
Drop I Word;
Run;
```

This example illustrates some of the complexities of our input data. The corrupted words include “Aircraft”, “Airways”, “Airlines”, “Airport”, “Air Test Center” and “Air Force”. Note that “Airlines” is broken into three words. Also, line 12 is a person’s name which appears to have been Allen. To make matters worse, “Aircraft” has a comma inserted in one case. For most of these errors, we can pick a specific bad string such as “Ai rcraft” and do a global find/replace with our text editor. We would then go through the list fixing the obvious bad words. Obs. 12 looks like the first name should be “Allen” and we would probably be safe in assuming so; however, some names are not so easily guessed and in these instances, we used the PDF copy to verify the spelling. Since we can assume that the PDF

does not show “Ai len”, we would have to search the person’s surname, “Jacque” in order to verify the proper spelling of his first name. At this point, the normal approach would be to either go to our desktop and open Acrobat Reader, find the file and open it or we could use “My Computer” to locate the file and double click it to open it. Since we had a number of files to process, we used a macro to allocate our Filerefs since we used year as a macro variable and simply specified the particular year in the macro call. In this macro, there were Filerefs for both the text file and the PDF. Having these Filefrefs available gave us the third alternative of clicking on “File Shortcuts” in the SAS Explorer window, clicking the Fileref for the library that has the PDF, and clicking the PDF to open it. The latter approach can be a lot faster than the other two since only the allocated files appear in the list that is presented.

Obs	Line	String
1	36	Hughes Ai rcraft
2	415	Hughes Ai rcraft Co.
3	582	Qantas Ai rways
4	1276	American Ai rl ines
5	1277	DFW Ai rport, TX
6	2019	Grumman Ai rcr,aft Sys
7	2474	Northrop Ai rcraft Co
8	2660	Ai r Force
9	2845	Trans World Ai rl ines
10	3462	Hughes Ai rc ra ft Co.
11	3822	Nava I Ai r Test Ctr.
12	4657	Jaques, Ai len
13	4855	U. S. Ai r Force
14	5315	Trans World Ai rl ines
15	6286	U. S. Ai r Force
16	6649	United Ai rl ines
17	6773	United Ai rl jnes
18	6805	United Ai rl ines
19	8019	DFW Ai rport, TX
20	8209	Hughes Ai rcraft
21	9569	Western Ai rl ines
22	9948	Hughes Ai rcraft
23	11137	DFW Ai rport
24	11458	USAi r Hughes Ai rcraft
25	11509	United Ai rl lnes

Figure 3. Listing of the Lines Which Contain the String “Ai”.

With the output of Figure 3 in hand, we can do such global changes as converting “Ai rcraft” to “Aircraft” and work our way through the list. Then another submission of our SAS code will (hopefully) confirm that we have fixed all of the instances of this particular problem.

In addition to words being fragmented, there were situations when a lower case “I” was rendered as an upper case ell in the middle of a word. In order to reduce the amount of visual checking, this type of situation was easily found using the line

```
If Propcase( word ) ne word then output;
```

in place of the If statement in the code shown above.

At this point, all we had to do was find and fix all the other erroneous strings in our text! Gradually we abandoned using Proc Spell in lieu of simply identifying an error and then using a combination of find/replace coupled with SAS string searches as illustrated in the preceding sections.

Disambiguating the Names of the Attendees

Our goal was creating a SAS data set containing the lists of attendees. In order for these to be analyzed in terms of how people attended the conferences, names and addresses of repeat attendees needed to be as uniform as possible. As an example of how names of an attendee and his company could change over time, N. H. Wooding also appears as Nat H. Wooding and N. H. Wooding, Jr. His company appears as VEPCO, VEPCO Environmental Department, Virginia Electric and Power Co., Virginia Electric and Power Company, Virginia Electric and Power, and Virginia Power.

At this point in the project, we had assembled the SAS data set of 16000 attendees. The approach to making the names uniform became one of creating a subset of those surnames that had appeared more than once and editing this set.

To simplify first names that were often nicknames, simple code similar to

```
If fname = 'Sandy' then fname ='Sandra'; else
If fname = 'Tommy' then fname ='Thomas'; else
If fname = 'Tom'   then fname ='Thomas';
```

was applied. We had to be careful in making such changes since simply checking for the initial sequence of “Tom” could change “Thompson” or “Thomasina” to “Thomas” which we did not want to do.

Then, we sorted the multiple year list by last name, first initial and looked for those who used initials in some years while spelling out their names in others.

The names of firms were a bit more of a challenge. Many names included abbreviations which may or may not have included periods and some names appeared with and without commas. Code such as

```
D_Company = compress ( D_Company , ',' );
D_Company = tranwrd ( D_Company , 'Dept. '  , 'Department ');
D_Company = tranwrd ( D_Company , 'Univ. '  , 'University ');
D_Company = tranwrd ( D_Company , 'Univ '   , 'University ');
D_Company = tranwrd ( D_Company , 'Co. '    , 'Company ');
D_Company = tranwrd ( D_Company , 'Corp. '  , 'Corporation ');
D_Company = tranwrd ( D_Company , 'Corp '   , 'Corporation ');
D_Company = tranwrd ( D_Company , 'Corp, '  , 'Corporation ');
D_Company = tranwrd ( D_Company , 'Inc '    , 'Inc ');
D_Company = tranwrd ( D_Company , 'Inc. '   , 'Inc ');
D_Company = compbl ( D_Company );
```

was used to globally make these names more uniform. We created a list of company names and sorted this with the NODUP option and we then created a copy of the company names calling the D_Company. We opened the sorted SAS data set as a viewtable with the edit option selected and manually edited D_Company to create a uniform list of company names. This list then served as the input to Proc Format and was used to create a format which via an Input function, fixed variations in company names as in the following code:

```
Data MakeFmt;
  set names;
  length start label $75.;
```

```
   retain fmtname 'D_cmp'
   default 75
   type    'J';
   Start =   Company  ;
   Label = D_company;
 run;
 proc format cntlin = MakeFmt;
 run;
 Data Goodnames;
    set MixedName;
    D_Company = InPut(  Company  , $D_cmp43. );
 run;
```

The project needed further effort to add such information as zip codes but most of this work used techniques similar to those already described. One instance that is a little different was that of making certain parts of place names uniform in order to better match our list of zip codes. In the following case, “Fort” appears abbreviated at times so the following code

```
if City =:'Ft' then City = Compbl( 'Fort ' || Scan( City , 2 ));
```

looks for cases where the city name starts “Ft” (using the colon after the equal sign lets this statement recognize “Ft” and “Ft.”) and then adds “Fort” to the second part of the city’s name. This, of course, would not work if the city name had three words in it.

Conclusion

We needed to prepare a reasonably uniform list of some 16,000 conference attendees that was drawn from scanned copies of printed proceedings spanning a couple decades and whose print quality varied with the year of publication. The attendance information comprised nearly 48,000 lines of text not counting blank lines. Various SAS string manipulation techniques helped in this process. While a different set of data would likely not have identical problems, the techniques needed to discover the problems and to do the data cleansing would be similar.

References

Okerson, Barbara. Old But Not Obsolete: Undocumented Procedures, SESUG 2007
http://analytics.ncsu.edu/sesug/2007/SD06.pdf

sasCommunity.org Proc Spell documentation
http://www.sascommunity.org/wiki/Proc_spell

Acknowledgements

Contact Information

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

Nat Wooding
1000 Hallsboro Road
Midlothian, VA 23112
(804) 379-3769
Email: nathani@verizon.net

Richard W. La Valley
Strategic Technology Solutions, Inc.
Herndon, VA 20171
(703) 732-1447
Email: rwlavalley@gmail.com.