Paper CC94

SEARCHING FOR (AND FINDING) A NEEDLE IN A HAYSTACK: A BASE SAS® EXPLORATORY SEARCH TOOL TO FACILITATE TEXT MINING

Troy Martin Hughes

ABSTRACT

Text mining can add analytic and business value by uncovering hidden truths and trends in unstructured, semi-structured, and structured text data. While SAS Text Miner presents a comprehensive solution to text mining and content analysis, simpler, targeted questions warrant a more straightforward solution. When triaging a new data set of unknown content or structure, or to the new hire unfamiliar with a customer's complex data repository, answering questions about text data—or even knowing where to begin your search—may represent a formidable challenge. Initial analytic questions may include the need to determine if a word or phrase appears within a database and, if so, in which libraries, data sets, and fields and with what frequency. This text describes a Base SAS tool that iteratively searches SAS libraries and data sets for a single word, phrase, or a list of words or phrases. An HTML report displays search results, including the location, frequency, and context in which search terms appear. Moreover, because libraries and data sets are identified and parsed dynamically, this out-of-the-box tool is scalable and extensible to diverse SAS environments and can be implemented in seconds.

INTRODUCTION

A primal joy to an analyst or data scientist can be the acquisition or introduction of a new database, which can signal new capabilities, business opportunities, or simply analytic intrigue. In this era of big data, a new hire similarly may be facing not only an unknown workplace and workforce, but moreover a complex data infrastructure comprising thousands of data sets and tens of thousands of fields. Technical deficiencies such as lax data quality standards, lack of data dictionaries, unknown data structure, or lengthy, unstructured text fields that obfuscate data content each can add further complexity and challenges. Even under ideal circumstances in which data are well documented, defined, organized, and intuitive, data relationships, dependencies, constraints, and business rules may require development of extract transform load (ETL) infrastructure before data analysis can commence. Thus, to many analysts, much of the excitement of a new data source may stem from the anticipation of challenges yet to be overcome through meticulous and creative problem solving.

But challenges and complexities quickly can turn to crises when customers and decision-makers are clamoring for answers. New databases have a way of arriving with accompanying expectations for immediate analytic intent and business use, often by stakeholders who are neither responsible for data staging nor realize its inherent technical requirements. Especially where the initial data acquisition process has taken longer than expected or promised, or where data are intended to answer tactical questions whose business value decreases exponentially in hours or days, a substantial push may be made to deliver simple solutions or metrics immediately. This can place developers, analysts, and data base administrators (DBAs) on the opposing side of the table from other stakeholders requesting instant access and analysis of data, leading to frustrations on both sides.

Agile methodologies—such as Scrum, Extreme Programming (XP), Lean, and others—espouse the incremental delivery of business value which affords a compromise between these potentially opposing perspectives. Decision-makers are able to prioritize the analytic questions they find most valuable and exigent, while analysts and developers provide estimates of the time required to produce no-frills solutions and quickly act to deliver that value[i]. Thus, the "30 percent solution" delivered today may lack infrastructure, automation, and an overall polished finish, but despite these and other potential caveats, nevertheless provide immediate value to those requesting it. Flexible, dynamic tools too can support tactical decision-making and, to that end, the %NEEDLE macro represents a solution that can answer straightforward data content questions as soon as raw data sets are loaded into SAS libraries.

SAS Text Miner provides a comprehensive solution to text mining and content analysis challenges. Its implementation, however, may be excessive or unwarranted for simple, straightforward questions that aim to assess

content inclusion and location. As a component sold and licensed separately from Base SAS, its implementation also may not be feasible for some organizations. The %NEEDLE tool answers the simple question of whether a word, phrase, or list of words or phrases can be located within data sets. SAS data sets are dynamically identified, iteratively parsed, and results are displayed in an HTML report that includes the library, data set, field, and observation of each search term, as well as the frequency of occurrence. When run with the optional "maximum verbosity" parameter, the tool additionally displays the text in which the search term was discovered, with all search terms highlighted for readability. Because the macro requires only the modification of the search term(s), it quickly can be implemented in any environment and, as soon as data are staged in SAS libraries, can provide business value that directs further analytic research or reporting.

%NEEDLE INVOCATION AND EXAMPLE

The %NEEDLE macro is defined with the following parameters:

```
macro NEEDLE(liblist= /*space-delimited list of SAS libraries to be searched */,
   termlist= /* ASTERISK-delimited list of terms to be searched */,
   path= /* path to which search report is written
   */, rpt= /* name of HTML search report */,
   highlightcolor=red /* color used for highlighting search terms in HTML text */,
   limit=100 /* limit of highlighted terms returned per observation */,
   maximumVerbosity=Y /* (Y) or (N), displays highlighted terms and text */);
```

The following example and implementation of %NEEDLE leverages the US Fire Administration (USFA) National Fire Incident Reporting System (NFIRS), the database of record for fire-related incidents, as well as firefighter injuries and fatalities.[ii] NFIRS represents a complex relational database comprising 19 tables and hundreds of fields[iii] and, despite being well documented, requires some degree of effort to understand given its comprehensive scope.

One hour ago, your organization received the NFIRS database and temporarily staged all 19 tables in a SAS developmental library, NFIRS. Analysts and developers are still exploring data relationships, and are focused on conceptualizing how best to transform and represent the data when management makes a preliminary request for you to determine the presence and scope of methamphetamine labs in NFIRS data.

A development and analytic shop that embraces Waterfall life cycle philosophy might view this request only with contempt and frustration—everyone knows you have to develop a refined ETL process and ensure quality standards before analyzing any data. In your Agile environment, however, business value is released incrementally, and because interest in meth labs reflects your customer's immediate top priority, your team foregoes the ETL finery and instead embraces the analytic task. The %NEEDLE macro provides a first look into text data content and is invoked on the NFIRS library with the following statement:

```
%NEEDLE (liblist=NFIRS, termlist=meth*meth lab*methamphetamine, path=C:\methlab\,
   rpt=meth, highlightcolor=yellow, limit=10, maximumVerbosity=N);
```

After iteratively identifying and parsing all NFIRS data sets, the macro produces the following output file, saved to C:\methlab\meth.html, comprising two reports, abbreviated and represented below with fictitious data:

NFIRS HAZCHEM

Field	Total Count	meth	meth lab	methamphetamine
chem_name	7	5	1	1

NFIRS ARSONAGENCYREFERRAL

Field	Total Count	meth	meth lab	methamphetamine
ag_street	1	1	0	0
ag_city	2	2	0	0

With a single line of code, %NEEDLE produces a contingency table that demonstrates observed frequencies within each data set. All character fields are searched and a report is produced for every data set having at least one hit. To provide more context and granularity, the macro parameter maximumVerbosity—an Infocom® homage—subsequently can be activated. In cases in which data structure and content are unknown, maximumVerbosity should be turned off the first time %NEEDLE is executed to prevent potentially enormous HTML reports, due to the combined effect of lengthy text fields and a high number of returned search results. Continuing with the above example, %NEEDLE is executed below a second time with "Y" indicated:

```
%NEEDLE (liblist=NFIRS, termlist=meth*meth lab*methamphetamine, path=C:\methlab\,
   rpt=meth, highlightcolor=yellow, limit=10, maximumVerbosity=Y);
```

In addition to the above two HTML reports, %NEEDLE now also produces two additional reports that display full text with highlighted search terms. In each report, the Obs field represents the observation or record number for which the term was discovered, and Term(s) is the comma-delimited list of terms that are found in the field:

NFIRS HAZCHEM

Obs	Field	Total Count	Term(s)	Text
343	chem_name	3	meth, meth lab	The trailer was discovered to contain charred sulfur residue from the production of meth, common for mobile meth labs.
567	chem_name	4	meth, methamphetamine	Methamphetamine was present in trace amounts, although most of the meth had been incinerated. Matches used in meth production were recovered.

NFIRS ARSONAGENCYREFERRAL

Obs	Field	Total Count	Term(s)	Text
145	ag_street	1	meth	Plymeth Place
1,454	ag_city	1	meth	Methville
1,530	ag_city	1	meth	St. Methodistburg

Concluding this example, the full text results demonstrate the field chem_name is potentially relevant and useful for the requested analysis. Because the search term "meth" is contained within both the terms "methamphetamine" and "meth lab," each occurrence of the latter two terms also will be credited for "meth." Thus, in observation 343, "meth" is observed twice and "meth lab" once for a total count of three terms. Additional terms of analytic value—such as "mobile meth lab"—also may be discovered, which can lead to further model or search refinement. False positives as well will occur, as evidenced by two geographic fields (ag_street and ag_city) that have values containing the letters "meth." False positives also help refine later text mining models as they illustrate additional (unwanted) contexts that should be eliminated. Now armed with this information about the new NFIRS database, you were able to leverage %NEEDLE to focus analytic efforts on the HAZCHEM data set to begin providing instant value to decision-makers.

CONCLUSION

The analysis and exploitation of new databases often represent a challenging yet creative commitment to be met by analysts, DBAs, and other developers. Agile methodologies, when applied to analysis and analytic development, can provide incremental benefit and business value to customers, thereby further directing subsequent analyses or decision making through responsible flexibility. The %NEEDLE macro facilitates this pursuit by quickly and dynamically generating a report that displays search terms that were discovered, identifies where they are located, with what frequency, and in what context, thus obviating the need for a tedious, manual text mining.

APPENDIX A. CONVERT AND SPACETOCOMMA MACROS

```
%macro CONVERT (convertword= /* word transformed into SAS V7 variable name */);
%if %sysfunc(nvalid(&convertword,v7)) %then %let newconvertword=&convertword;
%else %do;
       %if %eval(%sysfunc(anyalpha("&convertword"))^=2) %then %let
                     convertword=_&convertword;
       %let convertword=%lowcase(&convertword);
       %let converti=1;
       %let convertvalue=1;
       %let newconvertword=;
       %do %while(&converti <= %length(&convertword));
              %let convertpos=%substr(%bquote(&convertword),&converti,1);
              %if ("&convertpos">="a" and "&convertpos”<="z") or
                     ("&convertpos">="0" and "&convertpos"<="9")
                     or ("&convertpos"="_") %then %do;
                     %let newconvertword=&newconvertword%substr(%bquote
                            (&convertword),&converti,1);
                     %let convertvalue=1;
                     %end;
              %else %do;
                     %if &convertvalue=1 %then %let newconvertword=
                     &newconvertword._; %let convertvalue=0;
                     %end;
              %let converti=%eval(&converti+1);
              %end;
       %end;
&newconvertword
%mend;
```

```
options minoperator;
%macro SPACETOCOMMA(list= /* space-delimited list of words to be
              separated with double quotes and commas */,
       sep=SP /* the delimiter to be used, which is defaulted to SPACE (SP), but which
              can include any other single character */);
%global spacetocommaout;
%local i;
%local list;
%local mod;
%let i=1;
%let mod=;
%let spacetocommaout=;
%if %length(&sep)=0 %then %let sep=SP;
%if "&sep" in "SP" "sp" "SPACE" "space" %then
       %do; %let sep=;
       %let mod=S;
%end;
%do %while(%length(%scan(&list,&i,&sep,&mod))>1 and %eval(&i<10));
       %if %eval(&i=1) %then %let spacetocommaout="%scan(&list,&i,&sep,&mod)";
       %else %let spacetocommaout=&spacetocommaout,"%scan(&list,&i,&sep,&mod)";
       %let i=%eval(&i+1);
       %end;
&spacetocommaout
%mend;
```

APPENDIX B. NEEDLE MACRO

```
macro NEEDLE(liblist= /*space-delimited list of SAS libraries to be searched
       */, termlist= /* ASTERISK-delimited list of terms to be searched */,
       path= /* path to which search report is written
       */, rpt= /* name of HTML search report */,
       highlightcolor=red /* color used for highlighting search terms in HTML text */,
       limit=100 /* limit of highlighted terms returned per observation */,
       maximumVerbosity=Y /* YES (Y) or NO (N) displays highlighted search terms and
text */);
%local i;
%local j;
%local k;
%local liblist;
%local path;
%local rpt;
%local highlightcolor;
%local tablist; %local
varlist; %local limit;

%local maximumVerbosity;
%local termtot;
%local termcnt;
%if %upcase(&maximumVerbosity)=Y or %upcase(&maximumVerbosity)=YES %then %let
maximumVerbosity=Y;
proc sql;
       create table dic as
       select libname, memname, name, type,
       sortedby from dictionary.columns
       where libname in(%upcase(%spacetocomma(list=&liblist))) and
              (upcase(type)='CHAR');
       quit;
run;
data _null_;
       set dic end=eof; by
       libname memname;
       length tablist varlist
       $32000; if _n_=1 then do;
              tablist='';
              i=0;
              end;
       if first.memname then
              do; i+1;
              tablist=strip(tablist) || ' ' || upcase(strip(libname)) || '. ' ||
                     upcase(strip(memname));
              varlist='';
              end;
       varlist=strip(varlist) || ' ' ||
       strip(name); if last.memname then do;
              call symput('varlist' ||
              strip(put(i,$10.)),strip(varlist)); end;
       if eof then do;
              call symput('tablist',strip(tablist));
              end;
       retain tablist varlist i;
run;

* commence text
search; %let i=1;
%let termtot=;
%do %while(%length(%scan(&termlist,&i,*))>1);
       %let termcnt=_%convert(convertword=%scan(&termlist,&i,*));
       %let termtot=&termtot &termcnt;
```

```
      %let i=%eval(&i+1);
      %end;
data searchlist; *initialize the base data set;
      length _libname _memname $32 _obs 8 _var $32 _cnt 8 _term $100 _htmltext $12000
            &termtot 8;
run; %let
i=1;
%do %while(%length(%scan(&tablist,&i,,S)>1);
      %let tabs=%scan(&tablist,&i,,S);
      data x (keep=_libname _memname _obs _var _cnt _term _htmltext &termtot);
            set &tabs (keep=&&varlist&i);
      length _libname _memname $32 _obs 8 _var $32 _cnt _pos 8 _term $100
            _text $10000 _htmltext $12000 _arr1-_arr&limit $10 &termtot 8;
      array _arrfindrange{&limit} $10 _arr1-_arr&limit;
      %let j=1;
      %do %while(%length(%scan(&&varlist&i,&j))>1); %let
            vars=%scan(&&varlist&i,&j); _cnt=0;

            _term='';
            %let k=1;
      %do %while(%length(%scan(&termtot,&k))>1);
            %scan(&termtot,&k)=0;
            %let k=%eval(&k+1);
            %end;
      do i=1 to &limit;
            _arrfindrange{i}='9999999999';
            end;
      %let k=1;
      %do %while(%length(%scan(&termlist,&k,*))>1); %let
            term=%scan(&termlist,&k,*); _oldpos=.;

            _pos=1;
            do while(find(upcase(&vars),upcase("&term"),_pos)>0);
                  _cnt+1; %scan(&termtot,&k)=%scan(&termtot,&k)+1;
                  _pos=find(upcase(&vars),upcase("&term"),_pos); if
                  missing(_oldpos) then _term=strip(_term) ||

                      ifc(length(_term)<=1,strip("&term"), ', ' || strip("&term") );
                  _oldpos=_pos;
                  _arrfindrange(_cnt)=put(_pos,z5.) || put(length("&term"),z5.);
                  _pos+1;
                  end;
            %let k=%eval(&k+1);
            %end;
      if _cnt>0 then do;
            _libname="%scan(&tabs,1)";
            _menmame="%scan(&tabs,2)";
            _obs=_n_;
            _var="&vars";
            _text=&vars;
            _htmltext=_text;
            call sortc(of _arrfindrange{*});
            i=1;
            do while(_arrfindrange{i}^='9999999999 and i<=&limit);
                  _posstart=input(substr(_arrfindrange{i},1,5),8.);
                  _poslen=input(substr(_arrfindrange{i},6,5),8.);
                  if i>1 then do;
                        if (_oldposstart+_oldposlen-1)>=_posstart then
                              do; _arrfindrange{i-1}='';
                              _tempposstart=min(_oldposstart,_posstart);
                              _poslen=max(_oldposstart+_oldposlen-
                              1,_posstart+_poslen-1)-_tempposstart+1;
                              _posstart=_tempposstart;
```

```
                                end;
                        end;
                    _oldposstart=_posstart;
                    _oldposlen=_poslen;
                    _arrfindrange{i}=put(_posstart,z5.) || put(_poslen,z5.);
                    i=i+1;
                    end;
             i=1;
             _addchar=0;
             do while(_arrfindrange{i}^='9999999999' and i<=&limit); if
                    ^missing(_arrfindrange{i}) then do;
                        _posstart=input(substr(_arrfindrange{i},1,5),8.);
                        _poslen=input(substr(_arrfindrange{i},6,5),8.);
                        _htmltext=ifc(_posstart=1,strip(''),substr(_htmltext,1,
                               _posstart+_addchar-1)) || "<FONT COLOR=
                               &highlightcolor>" ||
                               substr(_htmltext,_posstart+_addchar,_poslen) ||
                               "</FONT>" || ifc(length(_htmltext)=
                               _posstart+_poslen-1+_addchar, '',
                               substr(_htmltext,_posstart+_poslen+
                               _addchar,length(_htmltext)-(_posstart+_poslen)+1));
                               _addchar=_addchar+length("<FONT COLOR=
                               &highlightcolor></FONT>");
                        end;
                    i=i+1;
                    end;
             _htmltext='<HTML>' || strip(_htmltext) || '</HTML>';
             output;
             end;
       %let j=%eval(&j+1);
       %end;
       retain &termtot;
run;
proc append base=searchlist
data=x; run;
%let i=%eval(&i+1);
%end;

* print summary report;
ods listing close;
ods html path="&path" file="&rpt..html" style=sasweb;
proc report data=searchlist (drop=_term _htmltext);
       by _libname _memname;
       where ^missing(_libname);
       label _libname='Library' _memname='Table';
       column _var _cnt &termtot;
       define _var / 'Field' group;
       define _cnt / 'Total Count' sum;
       %let i=1;
       %do %while(%length(%scan(&termtot,&i))>1);
             define %scan(&termtot,&i) / "%scan(&termlist,&i,*)"
             sum; %let i=%eval(&i+1);
             %end;
run;
 %if &maximumVerbosity=Y %then %do;
    proc report data=searchlist; by
                _libname _memname;
             where ^missing(_libname);
             label _libname='Library' _memname='Table';
             column _obs _var _cnt _term _htmltext;
             define _obs / 'Obs' display;
             define _var / 'Field' display; define
             _cnt / 'Total Count' display;
```

```
            define _term / 'Term(s)' display;
            define _htmltext / 'Text' display;
run;
%end;
ods html close;
ods listing;
%mend;
```

REFERENCES

[i] Hughes, Troy. 2014. *When Software Development is Your Means Not Your End: Abstracting Agile Methodologies for End-User Development and Analytic Application*. Western Users of SAS Software (WUSS).

[ii] U.S. Fire Administration. *About the National Fire Incident Reporting System (NFIRS).* Retrieved from http://www.usfa.fema.gov/fireservice/nfirs/about/.

[iii] Federal Emergency Management Agency (FEMA). *National Fire Incident Reporting System Version 5.0 Fire Data Analysis Guidelines and Issues.* July 2011. Retrieved from http://www.usfa.fema.gov/downloads/pdf/nfirs/nfirs_data_analysis_guidelines_issues.pdf.

CONTACT INFORMATION

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

Name: Troy Martin Hughes
E-mail: troymartinhughes@gmail.com