

BASE SAS® Fuzzy Search Algorithm

Michael McCarthy PE, Austin Energy, Austin TX USA

ABSTRACT

Stored process users frequently need to search SAS data set with imprecise or “Fuzzy” criteria. A “Fuzzy” search requires logic that ranks and decides an outcome based upon a relative quality of a match to the submitted string. This paper is written for BASE SAS® programmers who want to perform a “Fuzzy Logic” search against a string field in a BASE SAS® data set. A method to search an existing data field against a string of words entered by a user is described, and a step-by-step example of a fuzzy street address search using BASE SAS® within stored process is presented.

INTRODUCTION

It’s possible to search a text column within a BASE SAS® data table with “mostly correct” information submitted by a user. The trick is to compare the individual words within the two strings, regardless of order, to arrive at an overall matching score. Here, the term “word” refers to a string of characters bounded on the left or the right by a delimiter, or not bounded at all. There are several key commands within BASE SAS® to help with this task. In particular, the **SCAN**, **SUBSTR**, and **LENGTH** functions make it easy to break any string into individual words, and the **COUNTW** command makes it possible to count the number words within a string. Thus, these BASE SAS® functions can be used together to parse any string into words, and then place these words into a SAS array.

While we could store both strings as arrays, it’s best only to store the smaller string this way. In particular, if we store both strings as arrays we will need N X M iterations to complete the comparison, where N and M are the respective number of words in each array. However, if we instead only store the smaller string as an array and search the larger string using the **COUNT** function, then we only need to iteratively search N times, where N < M. Mathematically, we are making at least the same number of comparisons since the relationship between the larger string and our search word from our array is given by

$$\text{Number of comparisons using COUNT} = (\text{LENGTH (LARGE STRING)} - \text{LENGTH (SEARCH WORD)} + 1)$$

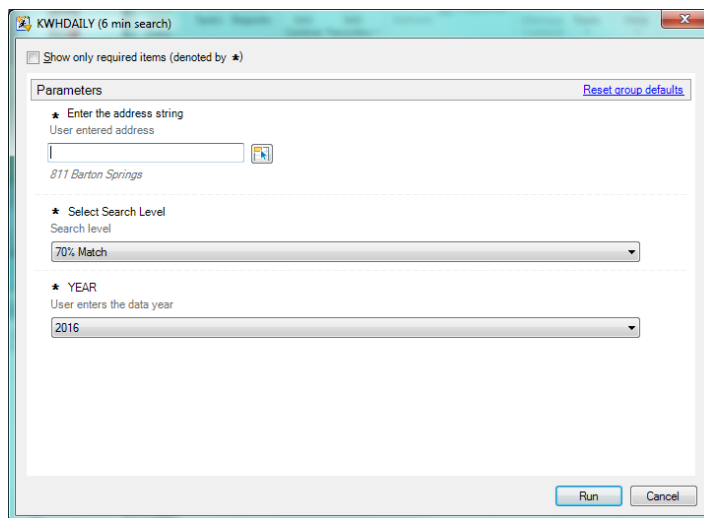
And therefore

$$\sum_{i=1}^N (\text{LENGTH(LARGE STRING)} - \text{LENGTH(SEARCH WORD}(i)) + 1) \geq N \times M$$

However, the **COUNT** function performs comparisons within memory and is therefore much faster than iteratively comparing two arrays. **COUNT** is a very handy search tool since it will search through the entire larger string in one step, counting the number of times the search word is found within the larger string. Comparing the strings in this manner also means that proper grammar, word order, and syntax are not important within either search string. Thus, a count of matching words for each array element is found in this way, resulting in a matching score for each comparison.

METHODOLOGY

The BASE SAS® script shown in the Appendix is stored in the metadata and executes on the Workspace server. It can be accessed from the Stored Process Web Application, Enterprise Guide®, or Microsoft Office®. When the user selects the report link, the menu below appears (MS Excel® version shown):



The user is required to enter search string, matching criteria, and select a table to search by year. These prompt values are processed within the stored process as the corresponding macro variables:

“&SEARCH_ADDRESS”, “&LEVEL”, and “&YEAR”. Here, the “&SEARCH_ADDRESS” represents a collection of words we are to search for, the “&LEVEL” is the acceptable threshold for the percentage of search words found, and the “&YEAR” is used to select the data table to be searched. In a nutshell, either the submitted string or the search string will be parsed and stored as array of words, depending on the string length. The words of the smaller string will then be searched against the larger string. The percent of matched words from the smaller string define the search score, and any records with a score above the threshold will be added to the output data set. In this particular example, the resulting output file is used to retrieve customer energy use.

BASE SAS®

Once the search information has been collected, the stored process begins by loading the macro “FUZZY” and assigning the data search library on the server (Appendix Steps 1-3). The macro FUZZY (called on step 24) will search the SOURCETBL (step 5) for matches to the SEARCH_ADDRESS. The length of the SEARCH_ADDRESS and ADDRESS are compared and stored in the strings SMALL and LARGE (steps 16 and 17), depending upon the overall length. The smaller string is then stored in array “PIECE” (step 11), and the array “FOUND” (step 12) is used to track how many times a corresponding word from PIECE is found within the LARGE string. When the ratio of the number of matched words to the total number words from the smaller string exceed the threshold LEVEL, the record is added to the output table “FOUND” (step 23). A PROC SQL join is then used to merge the energy data with the FOUND records, and the output table “STREAM” and sent to a user browser or Excel® (steps 25 to 27).

EXAMPLE OF THE SEARCH STRING

Suppose a user enters “101 S Oak” with a threshold level of 50%. When this string is compared to candidate “103 S Oak Wood”, our algorithm determines that the search string is the smaller of the two since the lengths of each are 9 and 14, respectively (Appendix steps 16, 17). The search string, now stored in the text variable “SMALL”, is parsed into words, stored in SAS array PIECE, and a word by word search is conducted (step 19), and the value of COUNT is stored in SAS array FOUND. The result of each iteration is shown in the table below:

PIECE	SEARCH LARGE	FOUND
101	103 S OAK WOOD 101 101→ ... TOTAL = 12	0
S	103 S OAK WOOD S S → ... TOTAL = 14	1
OAK	103 S OAK WOOD OAK OAK→ ... TOTAL = 12	1
TOTAL	38	2

A counter, called “MATCHWD” is tallied every time FOUND (i) ≥ 1. The percentage of the matching words from the search array is calculated (step 23):

$$\text{PRCT} = \text{MATCHWD} / \text{LENGTH (SMALL)} = 2/3 = 0.67$$

Since 0.67 is greater than 0.5, this record is added to the data set FOUND and results are sent to the output data set FOUND. The data set found is then joined with energy data in output table STREAM, and records from both FOUND and STREAM are sent to the user.

CONCLUSION

The BASE SAS code presented in this paper is an example of a user friendly fuzzy search of a SAS data table. The algorithm discussed finds a “best fit” within a SAS data table based upon a user entered search string and match criteria. This type of search can perform a quick and relevant search by leveraging the advantages of the COUNT function to search a string for specific word in memory. This allows users to efficiently search BASE SAS® data tables with “mostly correct” information, rather than exact values.

APPENDIX

```

/*****
/* SEARCH DAILY KHW BY ADDRESS */
/* WRITTEN BY MIKE MCCARTHY, PE */
/* DATE: 08/24/2016 */
/* DESCRIPTION: */
/* SEARCH THE DAILYKWH DATA FILE BY CUSTOMER ADDRESS */
/* AND STREAM DATA BY YEAR */
*****/
/*STEP*/
/*1 */ %STPBEGIN; /*BEGIN STORED PROCESS*/
/*2 */ %macro FUZZY(SOURCEDIR, SOURCE,SEARCH_ADDRESS=);
/*3 */ libname SOURCE "/home/SAS/Data/&SOURCEDIR/";

/*4 */ data FOUND; /*BEGIN FOUND*/
/*5 */ set SOURCE.&SOURCE;
/*6 */ format SEARCH SEARCHLEVEL $150.; /*Prompt Fields*/
/*7 */ format LARGE SMALL $200.; /*Large and Small strings to search*/
/*8 */ format LNLG LNSMLL 5.; /*Length of the large and small strings*/
/*9 */ format PRCT MATCHWD 4.2; /*Compare ratio to LEVEL*/
/*10*/ Retain MATCHWD;

/*11*/ array PIECE (12) $50. PIECE1 - PIECE12;
/*12*/ array FOUND (12) 3. FOUND1 - FOUND12;

/*Define the search criteria*/
/*13*/ SEARCH = "&SearchAddress";
/*14*/ SEARCHLEVEL = "&LEVEL";
/*Search this string for a match*/
/*15*/ ADDRESS = substr(&SEARCH_ADDRESS,1,LENGTH(&SEARCH_ADDRESS)); /*READ IN SEARCH ADDRESS*/

/*16*/ if LENGTH(ADDRESS) GE LENGTH(SEARCH) then do;
        LARGE = UPCASE(ADDRESS); LNLG = LENGTH(ADDRESS);
        SMALL = UPCASE(SEARCH); LNSMLL = LENGTH(SEARCH);
        end;

/*17*/ else do;
        SMALL = UPCASE(SEARCH); LNLG = LENGTH(SEARCH);
        LARGE = UPCASE(ADDRESS); LNSMLL = LENGTH(ADDRESS);
        end;

/*SMALL FITS IN LARGE*/
/*18*/ SMWORD = countw(SMALL); LGWORD = countw(LARGE);
/*19*/ do i = 1 to SMWORD;
        PIECE(i) = scan(SMALL,i);
        FOUND(i) = count(LARGE,substr(PIECE(i),1,LENGTH(PIECE(i)) ));
        end;
/*20*/ MATCHWD = 0;
/*CALCULATE HOW DIFFERENT SMALL AND LARGE ARE*/
/*21*/ do i = 1 to SMWORD;
        if FOUND(i) GE 1 then MATCHWD = MATCHWD + 1;
        end;
/*22*/ PRCT = MATCHWD/SMWORD;

/*Decide the match level for the entered string*/
/*23*/ if PRCT GE Input(SEARCHLEVEL, 4.2) then output;
        Return;
run; /*END FOUND*/

%mend FUZZY;

/*24*/ data _NULL_; /*CALL MACRO FUZZY*/
        Call Execute ('%FUZZY(PROD/READS/YR&YEAR,CUSTLIST,SEARCH_ADDRESS=ADDRESS1_UPR)');
run;

```

```

/*****
/* Merge Data Sets Using SQL Command */
*****/
/*25*/ proc sql;
      create table STREAM as
      select distinct
      a.*,b.SP_TYPE_CD
      from SOURCE.DailyKWH as a inner join FOUND as b on a.SP_ID=b.SP_ID
      order by a.SP_ID, a.DAY;
      quit;

/*26*/ proc print data=FOUND (KEEP = ACCT_ID PREM_ID SP_ID SP_TYPE_CD SPLAT SPLONG ADDRESS1_UPR
                        CITY POSTAL IN_CITY_LIMIT DISTRICT) noobs;

      run;

/*27*/ proc print data=STREAM noobs;
      run;

%STPEND; /*END STORED PROCESS*/

```

ACKNOWLEDGEMENTS

Special thanks to John Trowbridge, PE (DABI Engineering Supervisor, Austin Energy), Liz Jambor, DABI Manager, Austin Energy), and Debbie Kimberly (VP Customer Energy Solutions, Austin Energy) for their continued support of our SAS Data Analytics Server.

REFERENCES

Paper 233-30 An Introduction to SAS® Character Functions (Including Some New SAS®9 Functions)

Ronald Cody, Ed.D

<http://www2.sas.com/proceedings/sugi30/233-30.pdf>

Paper 255-2012 The Use and Abuse of the Program Data Vector

Jim Johnson, Ephicity Corporation, North Wales, PA

<http://support.sas.com/resources/papers/proceedings12/255-2012.pdf>

CONTACT INFORMATION

I value your comments. You can contact the author at:

Michael McCarthy, PE

Austin Energy, Customer Energy Solutions, Austin TX USA

michael.mccarthy@austinenenergy.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.