Paper 86-25

Use of SPEDIS Function in Finding Specific Values

Yefim Gershteyn, Ph.D., SCIREX Corporation, Chicago, IL

ABSTRACT

One of the new functions in Base SAS® released in Version 6.12 is SPEDIS, which returns a measure of how a word is close to another word in spelling - an asymmetric spelling distance between the two words. The function can be efficiently used for such tasks as finding specific values of character variables in a data set, merging data sets, or subsetting data using a character variable when there is a possibility of different spellings or misspellings of its value. In many cases, the SPEDIS function works better for these tasks than other SAS functions – INDEX, SCAN. It is especially useful as an addition to other functions to refine the search for specific values. The challenge is to set up a certain spelling distance, which allows the user to select the relevant values of the character variable. One of the approaches to that is to start with a deliberately large value of the spelling distance and then reduce the distance after studying the output until arriving at the value of the SPEDIS function which selects only or mostly the desirable values of the character variable. This presentation is intended for the advanced SAS users.

INTRODUCTION

The way the SPEDIS function works can be explained as follows. Let us assume that there are two words (strings of characters), the first called QUERY, and the second called KEYWORD. Usually, the query is the value of the character variable that you typed, while the keyword is the value you might really mean. For example, you wanted to type the word SUGI (keyword), but instead you mistakenly typed the word SAS (query). Now you want to measure how bad is your mistake, i.e. how far is SUGI from SAS. You can do this in two steps. First, you convert the keyword to the query using a sequence of operations. For each operation you assign some specific scores (called costs). The costs as they are currently defined for the SPEDIS function are as follows:

Operation	Cost	Explanation
match	0	no change
singlet	25	delete one of a double letter
doublet	50	double a letter
swap	50	reverse the order of two consecutive letters
truncate	50	delete a letter from the end
append	35	add a letter to the end
delete	50	delete a letter from the middle
insert	100	insert a letter in the middle
replace	100	replace a letter in the middle
firstdel	100	delete the first letter
firstins	200	insert a letter at the beginning
firstrep	200	replace the first letter

In our example, to convert the keyword SUGI to the query SAS you would need:

- change the second letter from U to A (replace a letter, according to the table, the cost=100)
- change the third letter from G to S (replace a letter, the cost =100)
- delete the letter I from the end (truncate, the cost=50).

So, the total cost for this conversion is 250.

Note that the cost will not necessarily be the same when converting SAS as a keyword to SUGI as a query. In that case, the underlying transformations are:

- change the second letter from A to U (replace a letter, the cost=100)
- change the third letter from S to G (replace a letter, the cost =100)
- add the letter I to the end (add a letter to the end, the cost=35).

The total cost for this conversion is 235.

The second step in calculating the distance between the keyword and the query is to divide the conversion cost by the length of the query in integer arithmetic. The quotient is the value of the SPEDIS function. In our first example it is equal to 83 (250/3=83), in the second example it is equal to 59 (235/4=59).

Thus, the SPEDIS function is defined as asymmetric (switching the keyword and the query does not necessarily brings the same result) spelling distance between two words expressed as the normalized (divided by the length of the query) cost for converting the keyword to the query via a sequence of operations. Before comparing the values of the keyword and the query, the SPEDIS function removes any trailing blanks in both words. The function returns a nonnegative value, usually less than 100, never greater than 200. Currently, the costs are assigned using the default table above, however, according to SAS Institute Inc., future enhancements will allow one to modify the default costs used in calculations.

USE of SPEDIS

We found the SPEDIS function useful in implementing such related tasks as:

- finding all observations essentially describing the same events when the values could be affected by typing errors, various spellings, or misspellings,
- merging or subsetting data sets using a character variable with the values that are expected to have various spellings and misspellings.

One example of such variable could be verbatim term for adverse events in clinical trials data. An adverse event, e.g., as simple as hypertension in practice can be recorded in many different ways, e.g. HYPERTENSION, HTN, HYPERTENSIVE, etc. Also, there are many possible ways of misspelling: HYPERTNSION, HPERTENSION, etc. One method to find all related names is to print out all

unique values of the variable, manually select those considered relevant to hypertension, and then create IF, SELECT, or WHERE statements incorporating the exact values which have been selected. Of course, in case of a large data set this way can be very tedious and time-consuming.

The other way is to create the same statements using the INDEX function, which would cover all possible ways of misspellings we can imagine:

```
if  index(upcase(aename), 'HTN ') or
    index(upcase(aename), 'HYPERTENSION') or
    index(upcase(aename), 'HYPERTENSION') or
    index(upcase(aename), 'HYPERTNSION') or
    index(upcase(aename), 'HPERTENSION') …
```

Although this method might work well in the case of relatively simple values, it will not select some relevant values when the event is described in many words or it is a long word, as it is hard to cover all possible ways of misspelling in statements involving the INDEX function.

The third way is to apply the SPEDIS function, preferably in addition to the previous alternative. Calculating the distances between the three possible major versions of recording hypertension and the value of the variable, setting cut-off values for these distances, and selecting the observations using the cut-off values finds the appropriate observations:

```
if spedis(aename, 'HYPERTENSION') le 22 or
   spedis(aename, 'HYPERTENSIVE') le 22 or
   spedis(aename, 'HTN') le 27  then
hyper=1;
if hyper;
```

The challenge is to set the cut-off values for the SPEDIS function – 22 and 27 in our case.

CHOOSING THE CUT-OFF VALUE

As the specific values we search for can be very different, there are no standard values of the SPEDIS function to serve as the cut-off points in all cases (although our experience shows that the cut-off point is usually situated between 20 and 35 when using a specific predefined value as a keyword and the value of a variable as a query). Thus, in each case an individual adjustment should be made. One way to do that is to use empirical results from several attempts, starting with the greater values of the SPEDIS function.

As the value of SPEDIS function cannot be more than 200, and is rarely more than 100, one point to start is to specify the value of the function between 50 and 100. In a real life example, we had a data set with 22922 unique adverse event names. The code such as that:

```
data find;
    set find;
    if spedis(aename, 'HYPERTENSION') le 100 or
       spedis(aename, 'HYPERTENSIVE') le 100 or
       spedis(aename, 'HTN') le 100  then
    hyper=1;
    if hyper;
```

kept 22847 names, which is obviously too much. So in the next step we limited the value of the SPEDIS function to 50 and still obtained a list of 15324 names, most of which had no connection with hypertension. The abbreviation HTN, being short – only three characters in length - mostly contributed to that, as almost any sum of costs cannot be too large for substituting/replacing only three letters. That is why the next step is to set the value of the SPEDIS function with the keywords HYPERTENSION and HYPERTENSIVE to, say, 40, while the value of the SPEDIS function with the keyword HTN to a smaller value, say, 30. Now only 179 names were selected, still with some obvious mismatches. The next iteration lessened the values of the function only slightly: to 35 for the first two keywords and to 27 for the third one. Now the number of names is 90, with some misspellings being correctly selected: HPERTENSION, HIPERTENSION, HYPERTENION. At the same time, some entries with the word HYPOTENSION were also chosen. This suggests that we can slightly lessen the value of the function for the first two keywords – to 32. Proceeding in the way like that, we ended up with the value of 22 for the first two keywords and the value of 27 for the third one, as the smaller values of the SPEDIS function did not select some valid names for hypertension.

The final cut-off values of SPEDIS allowed us to select 57 unique names, with 55 of those essentially reflecting hypertension. Two names, HYPOTENSION and HYPOTENSIVE, clearly did not belong to this phenomenon, but a further reduction of the cut-off point of the SPEDIS function to eliminate these names from the list would also eliminate some valid names. So we just hard-coded these two values to not allow them in the selected pool of names. Overall, six misspelled words, which were not intuitively obvious, had been selected: HIPERTENSION, HPERTENSION, HYPERTENION, HYPERTESION, HYPERTESNION, and HYPERYTENSION.

USING SPEDIS WITH OTHER SAS FUNCTIONS

The use of the SPEDIS function alone is justified when the possible value of a character variable consists of one word. In other cases, it is a good chance that not all of the values consisting of two or more words would be selected, even if one of the words has the exact match with the keyword. Thus, for example, for the query “WORSENING HYPERTENSION” and the keyword “HYPERTENSION”, the distance is rather large - 54, which is significantly larger than the “usual” cut-off points for the value of the SPEDIS function.

On the other hand, using an additional search with the keyword ‘WORSENING HYPERTENSION”, might result in selecting some irrelevant phrases. For example, the distance between the query “WORSENED SAS TENSION” and the keyword “WORSENED HYPERTENSION” is rather small – only 22, while the first group of words, probably, do not relate to hypertension. Note that the spelling distance between the words “SAS TENSION” and “HYPERTENSION” is large enough – 50. Such a difference comes from the division by the relatively long query in the first case and by the relatively short query in the second case.

In cases when the character variable can consist of two or more words, it is better to first use the INDEX function to

select any strings containing the keyword, and after that use the SPEDIS function to add some values with misspellings or different spellings which the INDEX function would not select:

```
if index(upcase(aename), 'HTN ') or
   index(upcase(aename), 'HYPERTENSION') or
   index(upcase(aename), 'HYPERTENSIVE') then
  hyper=1;
**ADDITIONAL SEARCH USING SPEDIS FUNCTION**;
 if spedis(aename, 'HYPERTENSION') le 22 or
   spedis(aename, 'HYPERTENSIVE') le 22 or
   spedis(aename, 'HTN') le 27  then
 hyper=1;
 if hyper and not index(aename, 'HYPOTENS');
```

Another possibility is to use the SPEDIS function in conjunction with the SCAN function to calculate the distance between each word in the query string and the keyword, and then select the whole string if there were any values of the SPEDIS function equal or less than the predefined cut-off point:

```
**CALCULATE NUMBER OF WORDS IN THE STRING**;
**TRANSLATE TWO+  BLANKS INTO ONE BLANK**;
  aename=compbl(aename);
**CALCULATE NUMBER OF EMBEDDED BLANKS **;
  numblank=length(left(aename)) -
              length(compress(aename));
**CALCULATE NUMBER OF WORDS**;
  numwords=numblank + 1;
**SPEDIS BETWEEN EACH WORD AND KEYWORD**;
  do i=1 to numwords;
    if spedis(scan(aename, i), 'HYPERTENSION') le 22 or
       spedis(scan(aename, i), 'HYPERTENSIVE') le 22 or
       spedis(scan(aename, i), 'HTN') le 27 then
      hyper=1;
  end;
  if hyper;
```

In our examples, we used the SPEDIS function in the form SPEDIS(aename, 'HYPERTENSION') rather than SPEDIS('HYPERTENSION', aename). The second version, which calculates the normalized cost of converting a string containing in the variable AENAME to the word "HYPERTENSION" tends to select more diverse names and requires to set up larger values for the cut-off point.

CONCLUSION

The SPEDIS function can facilitate a search for particular values of a character variable, as it helps to account for possible misspellings or various versions of the values. The SPEDIS function can be used alone, or in conjunctions with such well-known functions, as INDEX, SCAN, and others. Setting a value of the SPEDIS function as a cut-off point for making decisions about the value of the variable requires some experience and can be done using a number of iterations.

CONTACT INFORMATION

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

Yefim Gershteyn
SCIREX Corporation
255 East Lake St.,
Bloomingdale, IL 60108
Phone: (630) 307-1112
Fax: (630) 924-0402
E-mail: ygershteyn@scirex.com

TRADEMARKS