

PHONETIC MATCHING OF CHARACTER DATA

S. David Riba, JADE Tech, Inc., Clearwater, FL

ABSTRACT

There are many occasions when character values must be compared on non-precise, or "fuzzy", criteria. For example, how do you find all the values in your data for a city such as Raleigh, NC. when the values could be spelled (or mis-spelled) as:

Raleigh, Ralleggh, Ralehg, or even Roleg

Traditionally, SAS programming looks at the TEXT of the character values and attempts to define all of the possible text variations. There is another way. Most people, when they do not know how to spell something, will phoneticize it and spell it based on how it sounds. You can compare text the same way, based on how the text value SOUNDS – phonetically. There are several tools in Base SAS® software to compare character data based on phonetic values. This paper will review these SAS tools, and present several techniques for comparing and summarizing data based on phonetic, instead of text, values. These techniques can be useful for all levels of SAS programming expertise.

INTRODUCTION

Most programs that need to compare character data rely on the SUBSTR function and other similar functions that extract portions of text for comparison. One of the primary problems with this approach is the need to anticipate all of the many different ways that text could appear. Data entry errors, typographical mistakes, misspellings, or just plain guesswork complicate the problem of comparing character data. The majority of SAS programmers have never considered comparisons based on the phonetic values of the text. The SOUNDSEX function and the Sounds-like operator on the WHERE statement can be useful tools for determining the phonetic equivalent of text.

THE CONCEPT OF SOUNDSEX

The concept of soundex has been around for many years, but is relatively new to SAS. It was first introduced in Release 6.07 of the SAS System. The

SAS soundex function is documented in Technical Report P-222. The Sounds-like operator on the WHERE statement was introduced later, and is documented on page 502 of the SAS Language Reference.

The soundex algorithms were first patented by Margaret K. Odell in 1918 and Robert C. Russell in 1922.

Soundex is based on an underlying principle of English and other Indo-European languages.

Fr scr nd svn yrs
-brhm lncln

The above introduction to the Gettysburg Address is moderately understandable even without the vowels or all the consonants. It is a technique that has long been used by linguists and speed-writers. That is, most English words can reasonably be represented by their consonants alone. It is this technique that forms the basis of soundex phonetic comparisons. Using soundex, all words can be reduced to a phonetic equivalent character string, which can then be compared.

The steps used by soundex to derive the phonetic equivalent of a character string are as follows:

1. Retain the first letter of the character string
2. Discard the letters A E H I O U W Y
3. Assign a numeric value to the following consonants:
 1. B F P V
 2. C G J K Q S X Z
 3. D T
 4. L
 5. M N
 6. R
4. Discard all duplicate classification values if they are adjacent (that is DT will result in a single value of 3, NN will result in a single value of 5).

In addition, the original algorithms stated that the maximum length of the resulting character equivalency must be no more than four characters, which the designers felt would represent the sound of a given word and any other words whose pronunciation was fairly similar. Also, words that had a soundex value with less than four characters were to be padded to a length of four characters for comparison. However, the SAS implementation of soundex does not follow either of these particular requirements. The results of both the SAS SOUNDSEX function and the Sounds-like operator in the WHERE statement are non-truncated and non-padded character strings. To be totally in accord with the recognized soundex algorithms, the results should be either truncated or padded to a length of four characters, as appropriate.

The soundex concept enables the user to select information based on phonetic equivalents. It will often pick up additional values that are similar to the target string, but will rarely miss values that it should find. As a hashing algorithm, the original concept of a four character evaluation string produces at most 8,918 unique pronunciation groups.

SOUNDEX IN THE SAS SYSTEM

There are two implementations of the soundex concept in the SAS System.

As mentioned previously, Release 6.07 introduced the SOUNDSEX function. The purpose of the SOUNDSEX function is to return the phonetic value of a character string as a character variable based on the soundex algorithm. Unlike other character assignments in SAS, the length of a new character variable that stores the results of the SOUNDSEX function is not defined based on the function results. Instead, it is assigned a length identical to the argument for the SOUNDSEX function.

For example

```
value = SOUNDSEX('RALEIGH');
```

The variable VALUE has a character value of 'R42' and a length of 7, the same length as the characters 'RALEIGH'. The value of 'R42' is derived from the first letter 'R', a '4' for the letter 'L', and a '2' for the letter 'G'. All of the other sample variations of the name 'RALEIGH' also have a value of 'R42', and they will compare as identical phonetic values. It is important to note that soundex is not case sensitive. All results are returned as upper-case values.

Subsequent assignments to the variable VALUE are treated by SAS the same as any other character assignments. A shorter value is stored without change, while a longer value is truncated to the defined length, in this case 7. To avoid any unexpected results, it is always prudent to use the LENGTH statement to specify the length of character variables that will contain the results of a soundex evaluation.

Besides the SOUNDSEX function, the SAS System also has the Sounds-like operator (=*) in the WHERE statement. This operator is one of four special operators that can only be used in conjunction with a WHERE statement. It uses soundex to extract data based on phonetic values.

```
WHERE city =* 'RALEIGH' ;
```

It is important to be aware that both the SOUNDSEX function and the Sounds-like operator can be used as part of a WHERE statement in either PROC or DATA steps. Thus, data can be extracted and compared phonetically in both PROC and DATA steps. In fact, SAS documentation has specific examples of using the Sounds-like operator as part of a WHERE clause in PROC SQL.

```
PROC SQL ;  
  SELECT * from sql.people  
  WHERE lname =* 'Johnson' ;
```

USING SOUNDEX

In the case of the city of Raleigh, the soundex value returned is R42. Each of the phonetic spellings at the beginning of this paper also will return a value of R42, allowing for a comparison based on phonetic equivalents. Using soundex, relatively long or commonly misspelled character names can be reduced to a simple phonetic equivalent.

For example:

```
cityval = SOUNDSEX('Philadelphia') ;  
stateval = SOUNDSEX('Pennsylvania') ;
```

The value for CITYVAL is P4341 and the value of STATEVAL is P52415. The results are the same whether the target character strings are defined as upper or lower case.

One of the major drawbacks to using SOUNDSEX is that the first letter must be an exact match. In the case of PHILADELPHIA, for example, SOUNDSEX would not have found any records spelled as

FILEDELFIA. However, since SOUNDSEX returns a character string it would be easy to compare based on all characters except the first character. The SOUNDSEX value of PHILADELPHIA is P4341, while FILEDELFIA has a value of F4341. These two values are identical except for the initial letter, even though the words are spelled completely different.

This problem with soundex can be eliminated by using either the SUBSTR function or the LIKE operator with the _ wildcard in a WHERE statement. By comparing all key consonant sounds in a string except for the initial letter, the resulting string can provide a fairly accurate approximation of the phonetic value of the target text.

To compare on the text string PHILADELPHIA:

```
WHERE (SUBSTR(SOUNDSEX(cityname),2) =:
      '4341') ;
```

or

```
WHERE (SOUNDSEX(cityname) LIKE '_4341') ;
```

Since the above examples are part of a WHERE statement, they can be placed in either DATA or PROC steps. Thus, if these statements were part of a PROC PRINT or a PROC REPORT, for example, the resulting report would consist only of those records that phonetically compared to the city of Philadelphia regardless of the spelling of the city name. Soundex can also be used in various SAS statistical procedures, like FREQ, MEANS, and TABULATE, to control the records that the procedure includes for analysis.

Another major limitation of soundex is that each of the principle consonants must be in the same relative order. For example, 'Florida' has a soundex value of 'F463', but 'Flodria' has a value of 'F436'. Transposing the letters 'r' and 'd' yields a different soundex value. Even so, the resulting soundex values are similar enough that a formula could be derived to evaluate whether or not there was sufficient similarity to constitute a phonetic match.

One of the primary criticisms leveled at soundex is that it is too inclusive. That is, soundex will find more phonetic matches than might be desirable. Typically, as you refine your character matching routines it might be prudent to consider additional tests to eliminate unwanted character matches.

Since the SAS implementation of soundex is not limited to a four character maximum, some

additional care must be taken to assure that phonetic matches are comparable. For example, when evaluating the names 'Smith', 'Smythe', 'Smithers', and 'Smithson' the SAS implementation of soundex returns 'S53', 'S53', 'S5362', and 'S5325' respectively. Notice that while 'Smith' and 'Smythe' have exactly the same soundex values, 'Smithers' and 'Smithson' have values that exceed four characters. The first three characters are identical in all instances, but the endings of each returned value are slightly different. This needs to be taken into account when comparing the results of a soundex evaluation.

CONCLUSION

Depending on the data involved and the types of comparisons that are needed, the SAS System contains a variety of options besides the SUBSTR function which are particularly useful for non-precise or 'fuzzy' character comparisons. With an understanding of these functions as well as several of the useful operators available with the WHERE statement, character data can be evaluated more effectively in both DATA and PROC steps.

The SOUNDSEX concept has been around for many years and is now available in Base SAS software. Soundex allows for the phonetic matching of character data. With the use of soundex (both the SOUNDSEX function and the Sounds-like operator in the WHERE statement) it is possible to extract and compare character data based on its phonetic values. Several examples of uses for soundex were presented and limitations to soundex were discussed.

REFERENCES

- SAS Institute, Inc., *SAS® Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07*, Cary, NC. SAS Institute Inc., 1991. pp 64-65
- SAS Institute, Inc., *SAS® Language: Reference, Version 6, First Edition* Cary, NC. SAS Institute Inc., 1990. pp 502
- SAS Institute, Inc., *SAS® Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition* Cary, NC. SAS Institute Inc., 1989. pp 164-165
- Riba, S. David (1997). Efficient Comparison of Nonprecise Character Data. *Observations*. Vol 6, No 2. pp. 17-20.

TRADEMARK INFORMATION

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

AUTHOR

The author welcomes comments, suggestions, and questions by phone (813-726-6099) or by e-mail JADETECH@MCIMAIL.COM.

The author may be contacted at:

S. David Riba
JADE Tech, Inc.
P O Box 4517
Clearwater, FL 33758
(813) 726-6099
INTERNET: jadetech@mcimail.com

AUTHOR BIO:

S. David Riba is CEO of JADE Tech, Inc., a SAS Institute Quality Partner who specializes entirely in applications development, consulting and training in the SAS® System.

Dave is the founder and President of the Florida Gulf Coast SAS Users Group. He chartered and served as Co-Chair of the first SouthEast SAS Users Group conference, SESUG '93, and serves as President of the Executive Council of the SouthEast SAS Users Group. He has been a SAS user for 15 years, and has been actively involved in both SUGI and the Regional SAS Users Group community since then. He has presented papers on SAS-related topics at SUGI, SESUG, NESUG, MWSUG, SCSUG, and PharmaSUG.

Dave is an unrepentant SAS bigor. His major areas of interest are efficient programming techniques and applications development using the SAS® System, particularly using Screen Control Language with SAS/AF® and FRAME technology.

