Paper 2557-2018

A simple approach to text analysis using SAS functions

Wilson Suraweera[1], Jaya Weerasooriya[2], Neil Fernando[3]

ABSTRACT

Analysts increasingly rely on unstructured text data for decision making than ever before. Text data mining is a process of deriving actionable insights from a lake of texts. It discovers unseen patterns of words in data or known words or textual patterns in undetected records in data bases. SAS has its own dedicated text mining tools such as SAS® Contextual Analysis, SAS® Text Minor. However, their use by general users is precluded by affordability and availability.

We developed a simplified but robust approach for text analysis using a combination of 3 simple SAS string functions namely Index, IndexW and SoundeX in Base SAS® macro environment. Application was further condensed to a few lines of SAS codes and tested with several large scale datasets in epidemiological studies to ascertain its applicability, reliability and scalability.

This article explains our approach and discusses its applicability by using a commonly available SASHELP dataset.

INTRODUCTION

Organizations have traditionally relied on structured data to derive value for their businesses but in recent times an unprecedented interest on unstructured data for decision making has been witnessed. With advances in computing power and availability of more economical data storage, the generation of unstructured data continues to expand at an exponential rate. Natural language processing and text analytic technologies are emerging forces and will be basic tools in future analytics.

Text mining is the process of deriving actionable insights from a lake of texts. It is used to discover meaningful textual patterns that would otherwise go undetected in the text fields in databases and enables, understanding the human emotions, digging out the creative and systematic stuffs in underline texts through statistical modeling.

Programmers are busy developing the necessary tools and almost all commercial data analysis products have added text analysis components to their products. SAS has its own dedicated text mining products such as SAS® Text Minor, SAS® Contextual Analysis, SAS® Sentiment Analysis, SAS® Visual Data Mining and Machine Learning and SAS® Visual Text Analytics. However, their usage by general Base SAS users is precluded by affordability, availability and flexibility.

We developed a simple but robust approach for text mining using a combination of three simple SAS string functions, namely Index, IndexW and SoundeX in the SAS® macro environment. Our application generates Bag of Words (BoW), Term Document Matrix (TDM), Term frequencies and other standard outcomes needed to proceed to advance text analysis. Our approach was tested with several large scale datasets to ascertain its applicability, reliability and scalability.

This article explains our methodology and discusses its applicability using a commonly available SAS HELP dataset.

METHODS

Definitions

We use the following terminology and standard definitions in presenting our methodology.

Term: a word or sequence of words of interest to search for in the target documents. **Target document:** Collection of text strings to be analyzed (e.g., Emails, can span several pages). **Document database:** database that contains the documents of interest along with other associated variables. **Bag of Words (BoW):** list of words or combinations of words picked up from the target documents in the document

database, presented along with their frequencies. The BoW model is mainly used as a tool to generate features or terms to understand the textual patterns available in the documents. The BoW in our application creates unigrams (single words) and n-grams (multiple word combinations). **Input Term Matrix (ITM):** an array of tuples consisting of a listing of terms and labels assigned to identify each term. These labels will be used as variable names in the Term Document Matrix to hold term frequencies. **Term Document Matrix (TDM):** a two dimensional array with rows representing the documents, columns representing the terms (represented by variable names assigned in ITM) and cells representing the frequencies of successful matches of objective terms in the document.

Key SAS string functions used in this text mining application

Following three SAS string functions are the key components of our application.

1. **INDEXW** - searches for a string which could be a single or multiple words and returns the starting position of the first occurrence of the search expression in the target expression. Strings can be character or alphanumeric. If a delimiter is not specified, the default delimiter is the blank character.

   Syntax: INDEXW(*target-expression, search-expression [, delimiter]*)

2. **INDEX** - searches for a string that can be a word or a pattern of characters in a part of a word or a string, and returns the position of the first occurrence of the search expression within the target expression. The function does not use delimiters.

   Syntax: INDEX(*target-expression, search-expression*)

3. **SOUNDEX** - returns a copy of the argument that is encoded having same key for similar sounding words in English language. The SOUNDEX algorithm keeps the first letter of the string and discards any of subsequent letters of (A E H I O U W Y) in the string. Remaining letters in the string are encoded by numbers 1 - 6 grouped as follows: (1 for B F P V; 2 for C G J K Q S X Z; 3 for D T; 4 for L; 5 for M N; 6 for R) while discarding two or more adjacent letters in the same group. The SAS SOUNDEX function generates all possible codes for the word sequence of the string. For example, Washington has a code of W25235. The SOUNDEX code generated by SAS is different from the standard American SOUNDEX code. Even if there are spaces in the string, the SOUNDEX function will generate the code for the entire string, disregarding spaces (Zizhong F., 2004).

   Syntax: SOUNDEX(*expression*)

INDEX and INDEXW are similar; the difference being that INDEXW looks for a word-like string with delimiter (space by default) at either end, while INDEX simply searches for designated patterns which do not necessarily to be a separate word in the target expression. For example, the term “Ever”, would be found within the word “Fever”. In this context, unlike INDEXW, the search outcome of INDEX may not be accurate but the function is useful when words in the target document are not properly separated.

Both INDEX and SOUNDEX outcomes are guesses but useful when string words are misspelled or words are not separated correctly. INDEX outcomes are more efficient for longer expressions while SOUNDEX outcomes are more efficient for small strings.

Additional SAS string functions used for text parsing and normalization

TRANSLATE, SUBSTR, STRIP, COMPRESS, COMPBL and UPCASE functions are used in text parsing to restructure the strings by suppressing unnecessary symbols, white spaces etc. associated with words. COUNTW, SCAN functions are used in the manipulation of strings.

Main program interface

Figure 1 presents the textual data mining process of our application. Out of two modules in the flow diagram, Module 1 creates the ITM, Module 2 is for development of the TDM. The ITM is developed using the list of objective terms specified for the purpose of data mining. The objective terms are usually picked from the BoW. TDM creation in Module 2 is a process of iterative search for each term of ITM in the target documents in the document database, in order to create frequencies of their occurrences. We use a SAS macro for iterative search for terms in the document database.

To connect Module 1 and Module 2, a well-defined ITM is required. The ITM candidate term generation task is an open research question. It depends upon the purpose of the text mining and needs human intervention to decide which terms are appropriate. Therefore, our approach was not designed to automatically connect Module 1 and Module 2.

For the convenience of explaining the SAS code in the algorithm, we divided our SAS code into distinct parts and presented them as in Program listings 1 to 6.

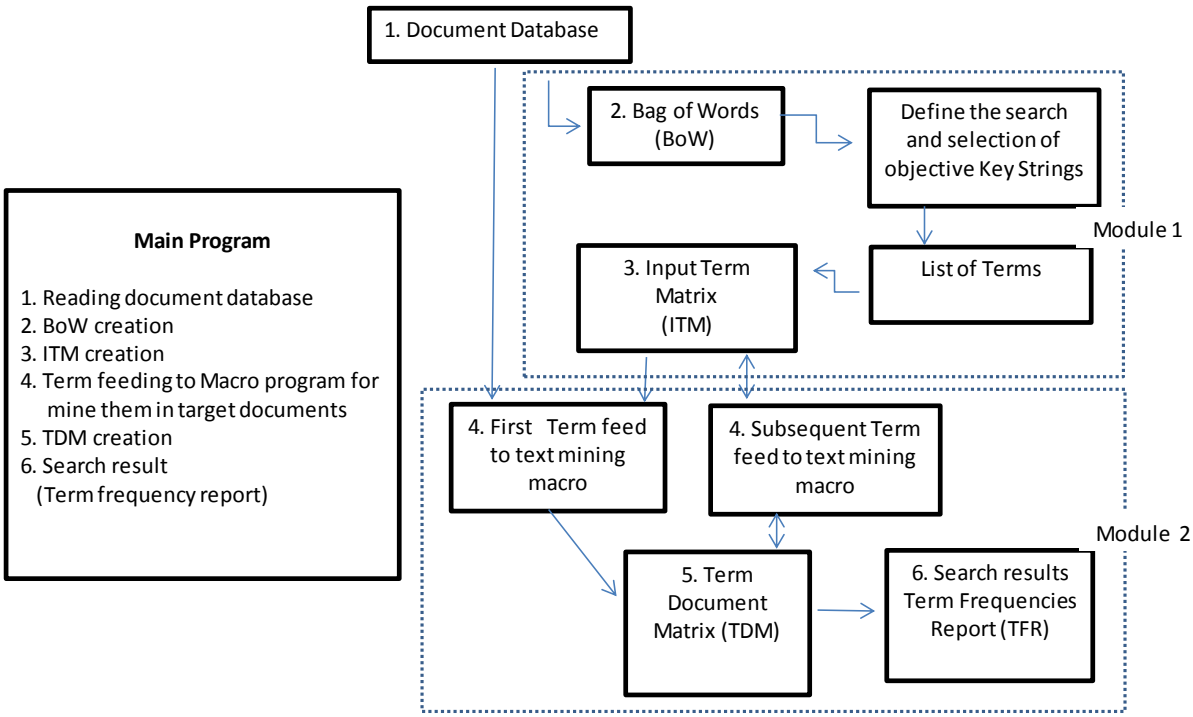

Figure 1: Textual data mining process

Document database and target documents

We chose the "adsmsg" dataset in the SASHELP library to demonstrate our application because it is readily available to any SAS user. The dataset has 426 records with 6 variables. The variable name "TEXT" was selected to represent the variable that contains the target text documents. In Program Listing 1, we assign the dataset name and the name of the variable that contains the target documents, to two macro variables so that they can be efficiently reused throughout our program.

Program Listing 1: Document database

```
/* Module 1: Definitions - document database */

%LET ds_name = sashelp.adsmsg;    /* <-- document database name*/
%LET Var_Data_Source = TEXT;      /* <-- Variable name, where target documents are stored in
                                       the document database*/
```

Bag of Words generation

Program Listing 2 is the SAS code that creates the BoW. Its term building process creates unigrams to 5-grams and the frequency with which they are found in the target documents. In fact, 5-gram limit is just for demonstration only; users can easily add more words to the terms, depending on their requirements.

In our algorithm, we first get rid of trouble-making characters, punctuation marks etc. using the TRANSLATE function and all unnecessary spaces using the COMPBL function. Users can add any character that needs to be omitted into the TRNSLATE function argument. We split each document string

into atomic words (assume maximum single word size=50) using the SCAN function and then each word is assigned to a temporary array. N-grams are generated iteratively concatenating the consecutive array items until the DO loop reaches the last word of each string. Table 1 lists the first few lines of the BoW created by this SAS code.

Program Listing 2: Bag of Words creation

```
 * Module 1: Bag of Words creation
DATA Bag_of_words;
 SET &ds_name ;
     Var_Data_Source=COMPBL(TRANSLATE(&Var_Data_Source, "" , ".,;:?!-/\
                                                    %1234567890$@#)("));
  N_words=COUNTW(&Var_Data_Source);/* N_words = No. of words in the document*/
  ARRAY word {1:1000} $50 _TEMPORARY_;
      DO i=1 TO N_words ;
      RETAIN word_i;
        word(i)=SCAN( &Var_Data_Source, i, ' ' );
       word_i=word(i);
       IF (i>1) THEN  word_2i= catx(" ",word(i-1), word_i); /* Bi-grams*/
       IF (i>2) THEN  word_3i= catx(" ",word(i-2),word(i-1), word_i); /* Tri-grams*/
       IF (i>3) THEN  word_4i= catx(" ",word(i-3),word(i-2),word(i-1),word_i); /* 4-grams*/
       IF (i>4) THEN  word_5i= catx(" ",word(i-4),word(i-3),word(i-2),word(i-1),word_i);/5-g
     OUTPUT;
 END;
RUN;

PROC PRINT DATA=Bag_of_words; RUN;
PROC FREQ DATA=Bag_of_words; TABLE word_i word_2i word_3i word_4i/NOCUM; RUN;
```

Table 1: Bag of Words dataset (Output of Program listing 2)

MSGID	Var_Data_Source	N_words	i	word_i	word_2i	word_3i	word_4i
50	Data set already exists	4	1	Data			
50	Data set already exists	4	2	set	Data set		
50	Data set already exists	4	3	already	set already	Data set already	
50	Data set already exists	4	4	exists	already exists	set already exists	Data set already exists

Input Term Matrix development

As we discussed earlier, the ITM development remains an active area of research. The SAS code in Program Listing 3 was created to demonstrate, how to develop the ITM manually. We selected several terms observed in the BoW module which may helpful to exemplify our strategic terms matching process later. These terms are read into a variable named “in_words” and we defined another variable named “out_words” to assign labels for each term that could be used as a variable name at the TDM later. The strategy applied in label creation was to select the first segment of the Term’s word string that fits a maximum length of 32 characters, and then replace each space gap between the words with underscores. The 32 character maximum is because SAS allows a maximum of 32 characters for a variable name. Table 2 shows the ”Term_matrix“ SAS dataset created by the code in Program Listing 3.

Program Listing 3: Illustration of Input Term Matrix

```
* Module 1: Input Term Matrix
      DATA Term matrix;
         INFILE datalines;
         INPUT in_words $1-50;
            in_words=TRANSLATE(UPCASE(in_words), "" , ".,;:?!-/\"); /* Text parsing */
           out_words=SUBSTR(TRANSLATE(STRIP(in_words),"_"," "),1,32);
        DATALINES;
               Data length must be between 1 and 200
               culation
               calcultion
               calculation
               Calculation requires
               Calculation requires more data
               Calculation requires more data please choose a column
               Parameter
               ;
        RUN;
        PROC PRINT DATA= Term_matrix; RUN;
```

Table 2: Input Term Matrix created by Program Listing 3

In_words	Out_words
ALREADY EXISTS	ALREADY_EXISTS
DATA LENGTH MUST BE BETWEEN 1 AND 200	DATA_LENGTH_MUST_BE_BETWEEN_1_AN
CULATION	CULATION
CALCULTION	CALCULTION
CALCULATION	CALCULATION
CALCULATION REQUIRES	CALCULATION_REQUIRES
CALCULATION REQUIRES MORE DATA	CALCULATION_REQUIRES_MORE_DATA
CALCULATION REQUIRES MORE DATA PLEASE CHOOSE A COL	CALCULATION_REQUIRES_MORE_DATA_P
PARAMETER	PARAMETER

Term Document Matrix creation

The most important step of text analysis is the development of the TDM. For this purpose, we created a simple SAS macro named "%KW_SEARCH" which is the heart of our application. Details of the text mining strategy used in the macro program will be discussed later. First, let's see how we are going to make use this in our main program. Program Listing 4 is the SAS code of the main program.

The Macro program is stored outside the main program. We compile it by providing the file name and the folder path to its location in the "%include" statement. Term feeding into the macro for mining each term in the target documents was designed in a "data _null_; run;" loop.

The "%KW_SEARCH" macro was designed to mine one term at a time, sequentially in each target document in the database and count the frequency of occurrences. At each term execution of the macro, these frequencies are outputted to a new variable and added to the right side of the database. Finally, this process forms the TDM with all objective terms.

Following are the parameters needed to execute the "%KW_SEARCH" macro program.

Parameter	**Description**
KW	Objective term wants to be searched (This is from ITM)
Var_KW_out	Output Variable name to identify each term (This is from ITM)
Var_Target_Doc	Variable name in which target documents are stored in the database
Data_IN	Database name where target documents are stored
Data_OUT	Database name to output text mining outcomes

When our term is a multiple word string, it needs to be kept within quotation marks, enabling SAS to understand that it is a single string. By using CALL SYMPUT statements, both terms and corresponding labels created in the ITM are assigned to two macro variables, " in_words" and "out_words" respectively. These macro variables will be used later as input parameters for "KW" and "Var_KW_out" in the macro program.

We use the "CALL EXECUTE" routine to execute our macro program. You can see that we use two "CALL EXECUTE" macro routines in the main program. This is a trick used to capture all outcome frequencies produced by the INDEXW, INDEX and SOUNDEX criteria for each term into a single database.

The first macro "%CERTAINTY_FACTOR" adds a new column name "certainty_factor" to the document database and then replicates the database three times assigning values 1, 2 and 3 to the new variable (See Program Listing 5). This replication facilitates the retention of each outcome of the INDEXW, INDEX and SOUNDEX functional evaluations of a term, under a single output variable, with row label values 1, 2 and 3 respectively for the "certainty_factor". To avoid replication of the database at each time, we restricted this step to the first term search only.

Again you can see that we assign the output dataset name of the "%CERTAINTY_FACTOR "macro to both the "Data_in" and "Data_out" parameters of "%KW_SEARCH" macro. By this strategy, at each iteration of the term feed, a new variable that captures the term frequencies will be added to the right of the target document database.

The entire Module 2 shown in Figure 1 earlier is captured in Program Listing 4 below.

Program Listing 4: Main program interface

```
* Module 2: Main program Interface
*(1) Compilation of the macro programs

   %INCLUDE "C:\SAS Global Forum 2018\SAS program\Macro_KW_search.sas";

*(2) Iterative term feeding to Macro Program
   DATA _NULL_;
     SET Term_matrix END=eof;
       in words 1=STRIP("'"||in words||"'");
       CALL SYMPUT('Keyword_in',in_words_1);
       CALL SYMPUT('Keyword_out',out_words);
*(3) First run: (Replication of document database and create "certainty factor" variable
     IF (_N_=1) THEN DO;
          CALL EXECUTE ( '%CERTAINTY_FACTOR(Data_IN=&ds_name, Data_OUT=Term_Doc_Matrix)');
     END;
 *(4) Term feeding to text mining macro
    CALL EXECUTE ('%KW_SEARCH(KW=&Keyword_in, Var_KW_out=&Keyword_out,Var_Target_doc=TEXT,
                       Data_IN=Term_Doc_Matrix, Data_OUT=Term_Doc_Matrix)');
   RUN;
```

Text mining macro program (%KW_SEARCH)

The "%KW_SEARCH" macro (Program Listing 5) provides a robust approach for text mining. It uses the INDEXW, INDEX and SOUNDEX basic SAS functions to assess our text mining outcomes. In addition to the exact match, it provides two additional options for misspelled or improperly-spaced words in the text documents.

Application of INDEXW, INDEX and SOUNDEX functions

Use of both the INDEXW and INDEX functions in our application is simple and straightforward. However, the use of SOUNDEX is complicated. As did in the Bag of Word generation (Program Listing 2), we split each document string into atomic words using the SCAN function. And then creates N-grams, by concatenating the consecutive atomic words, depend upon the "N" equals to the number of words in the objective term. For example, if the term string consists of two words, this step will make all consecutive pair of 2 words of the target document.

Within each document, these N-grams and the objective term string are sequentially assessed using INDEXW, INDEX and SOUNDEX functions. All successful evaluations of each function are counted and stored in temporary array variables using a DO Loop.

Each positive value returned by the INDEXW or INDEX evaluation implies that the objective term (KW) was found within the target document. Unlike the INDEXW evaluation, INDEX evaluation was designed to track the exact text patterns. Sometimes these patterns are found within part of another word string of the target document that we may not be interested. For example, word "Ever", could be found within the word "Fever".

The SOUNDEX evaluation was designed to track misspelled words. At SOUNDEX evaluation, both objective term and the corresponding N-grams of document string are converted into SOUNDEX codes and then both of these codes are assessed using the INDEX function to determine both encodings are matched.

Program Listing 5: "%Certainty Factor" and "%KW_SEARCH" SAS Macro Program

```
   %MACRO CERTAINTY_FACTOR ( Data_OUT=, Data_IN=);
        DATA &Data_OUT;
        SET &Data_IN;
          DO Certainty_factor=1 to 3;
            OUTPUT;
          END;
   %MEND;
  ----------------------------------------------------------------
  %MACRO KW_SEARCH(KW=, Var_KW_in=, Var_KW_out=, Data_IN=, Data_OUT=);
    DATA &Data_OUT;
    ATTRIB Flag_success length=3;
    KW = UPCASE(&KW);
    SET &Data_IN;
    Target_string=UPCASE(COMPBL(TRANSLATE(&Var_Target_Doc, " " , ".,;:?!-/\")));
    KW_words = COUNTW(KW);     /* Number of words in the objective Term string*/
    N_words  = COUNTW(Target_string);  /* Number of words in the target document string */
* Temporary array variable to retain information
    ARRAY word {1:1000} $50  TEMPORARY_ ; /* individual words of target document */
    ARRAY IdW {1:100}  _TEMPORARY_ ;      /* term matching frequencies using INDEXW function */
    ARRAY Idx {1:100}  _TEMPORARY_ ;      /* term matching frequencies using INDEX function */
    ARRAY Sdx {1:100}  _TEMPORARY_ ;      /* term matching frequencies using SOUNDEX function */
    Soundex_Count=0; Index_count=0; IndexW_count=0; /* Reset Temporary variables holding term
                                                      Frequencies */
* Make N-grams depend upon N equals to the No. of words in the Term string. E.g., If term string
* consists 2 words, this step make all consecutive 2 word pairs of Target document
    DO i=1 TO  N_words;
       word(i)=SCAN( Target_string, i, ' ' );  /* Explode the string into words by spaces  */
       IF i GE(KW_words) THEN DO;
          length Target_truncated $50;
          Target_truncated='';
          DO j=1 TO KW_words;
            Target_truncated= STRIP(CATX(" ", word(i-j+1) ,Target_truncated));
          END;
        END;
  *(1) Evaluation INDEXW function: Exact match
       IF (INDEXW(Target_truncated, STRIP(KW))>0) THEN IdW(i) =1; ELSE IdW(i) =0;
  *(2) Evaluation INDEX function:  Words not properly separated
       IF (INDEX(COMPRESS(Target_truncated), COMPRESS(KW))>0) THEN Idx(i) =1; ELSE Idx(i) =0
  *(3) Evaluation SOUNDEX function:  Misspelled
       IF(INDEX(SOUNDEX(Target_truncated), SOUNDEX(KW) )>0) THEN Sdx(i)=1; ELSE Sdx(i)=0;
  * Term Frequencies
        IndexW_count=IndexW_count + IdW(i);
        Index_count=Index_count + Idx(i);
        Soundex_Count=Soundex_Count+Sdx(i);
    END;
  * Final Evaluation
   IF (Certainty_factor=1) AND (IndexW_count>0) THEN DO;
                        &Var KW out=IndexW count;   END;
   IF (Certainty_factor=2) AND ((Index_count-IndexW_count)>0) THEN DO;
                         &Var_KW_out=(Index_count-IndexW_count);  END;
   IF (Certainty_factor=3) AND ((Soundex_Count-IndexW_count)>0) THEN DO;
                         &Var_KW_out=(Soundex_Count-IndexW_count);  END;
   IF (&Var_KW_out>0) THEN Flag_success=1;/* This variable will be added to the outcome dataset,
                                          to flag the record whether the search was successful*/
   DROP i j KW KW_words N_words Soundex_Count Index_count IndexW_count
        Target_truncated Target_string;
%MEND;
```

The final evaluation

For each positive value of INDEXW evaluation, both INDEX and SOUNDEX evaluations will also be positive. Therefore, at the final assessment of term frequencies, we adjust the INDEX and SOUNDEX outcomes for INDEXW before writing them into the output variables.
To easily identify which documents were successfully mined (with at least one count for a term), we created a flag variable named "Flag success" and added to the output dataset. By sorting this flag variable in descending order, we can have all documents having successful text mining outcomes.

RESULTS

We developed the following SAS code (Program Listing 6) to extract the results of our text mining task discussed in this paper. It first creates a macro variable to retain the list of Term output labels of the ITM. In fact these labels are now being a part of the list of variables of the Term Document Matrix.
In the PROC TABULATE procedure; this macro variable was used to define the analysis variables in the VAR statement. We produced two table reports to present the final outcomes of our application.

Program Listing 6: Report generation

```
* (a) List of Variable names created for identify each key key-string search
   PROC SQL  NOPRINT;
        SELECT out_words INTO : v_list_comma_sep separated BY ',' FROM Term_matrix;
        SELECT out_words INTO : v_list_blank_sep separated BY ' ' FROM Term_matrix;
   QUIT;

* (b) Summary of text mining results and TDM
  PROC TABULATE DATA=Term_Doc_Matrix;
    CLASS  certainty_factor text ;
    VAR &v_list_blank_sep;
  * Summery of term frequencies
     TABLE (&v_list_blank_sep), (certainty_factor='certainty factor' ALL='No of Terms
                               found')*SUM='' N='No of Documents';
  * Detailed term matrix
     TABLE text='Target Document'*(&v_list_blank_sep), (certainty_factor='certainty factor'
                                                  ALL='Total')*SUM='';
  run;
```

Table 3 shows the overall summary of the text mining task that we have discussed in this article. Our ITM had 9 terms (See Program Listing 3) while the document database (SAShelp.adsmg) that we have used for demonstration of our application had 426 documents. Table 3 presents the frequencies of these 9 terms assessed over 426 target documents by certainty factor. Our application has returned some of the misspelled words (CULATION, CALCULTION) for the word “CALCULATION” under the certainty factor 2 or 3, suggesting that our application works in consistent with our priori expectations.

Table 3: Summary of Term Document Matrix (TDM) frequencies

Term	Certainty factor			No of Terms found	No of Documents
	1	2	3		
ALREADY EXISTS	22	.	.	22	22
DATA LENGTH MUST BE BETWEEN 1 AND 200	.1	.	.1	.2	2
CULATION	.	6	.	6	6
CALCULTION	.	.	6	6	6
CALCULATION	6	.	.	6	6
CALCULATION REQUIRES	1	.	.	1	1
CALCULATION REQUIRES MORE DATA	1	.	.	1	1
CALCULATION REQUIRES MORE DATA PLEASE CHOOSE A COL	1	.	.	1	1
PARAMETER	13	1	.	14	9

Note: Certainty factor 1: INDEXW evaluation, Certainty factor 2: INDEX evaluation, Certainty factor 3: SOUNDEX evaluation.

Table 4 presents the first few rows of the TDM created by this application. The rows in this table represent target documents; columns represent the 9 terms that we have used in the text mining task while cells are the term frequencies. The second column is the certainty factor that we discussed in TDM creation section. As we discussed earlier, the frequencies related to certainty factor values 2 or 3 should be accepted only after a careful review of them to make sure whether they actually represent the terms of our interest.

Table 4: Resulting Term Document Matrix (TDM)

Target Document	Certaint	ALREADY_EXIS	LENGTH_MUS	CULATION	CALCULTION	CALCULAT	CALCULAT	CALCULAT	CALCULAT	PARAMETI
		.	.	.	.	.	.	.	.	.
Cancel.	1									
%$ could not be opened in update mode.	1	.	.	.	.	.	.	.	.	.
%1$ is used in the calculation of %2$.	1	.	.	.	.	1	.	.	.	.
	3	.	.	.	1	.	.	.	.	.
%L %1$ already exists in %N %2$ %N Do you wa	1	1	.	.	.	.	.	.	.	.
%L Select a path and database to copy to or sel	1	.	.	.	.	.	.	.	.	.

CONCLUSION

Text mining is used to pick up important information in unstructured documents that could be transformed into structured variables that a computer can understand and further process for discoveries. Still a huge gap between the human intelligence of natural languages and computer understanding continues to exist. This is a challenge for programmers regardless of the platform, and SAS programmers are no exception.

The Base SAS is a popular and widely used analytical tool in many areas of businesses and research. A vast majority of Base SAS users strives to develop text mining code within Base SAS environment without going for dedicated text mining products in a bid to avoiding additional costs required for acquiring such products. However, we have seen in many SAS discussion forums that many Base SAS coders are still struggling to do simple text mining tasks.

Our goal in this article was to introduce a simple application tool that can be easily used by general SAS users and to make their life easy by integrating our approach appropriately with their applications. Our application is robust, it not only does the exact matches of terms in target documents, but also detects all plausible results even where spellings are mistaken or words are not properly separated in the texts.

Our application has some Limitations.

- Our approach can be used only on English language texts, because the SOUNDEX function does not recognize other languages.
- Our document handling is limited to a string length of 32,767 characters because this is the maximum number of characters that can be held in a SAS string variable (SAS Institute, 2012). However, this string length is large enough to hold a 10 pages document of font size 10.
- We may not sure this application will run with older SAS versions because some functions like the COUNTW are recently introduced. We tested our code with the release 9.2 and found the application runs well.

Arguably, our main contribution is methodological. It is our fervent hope that this application and its simple approach will provide some contribution to advance the knowledge frontier in text mining methodology while equally contributing to making the text mining a simple and a common task for general SAS users.

REFERENCES

1. Suraweera W., (2010) Text analysis: An epidemiological case study in Malaria Deaths in India, paper presented at SAS Canada Health User Group (HUG), November 2010 Link
2. Zhong C., (2017) Text Analytics - Use Cases & Value, SAS Canada Data Science Forum , November 2017 Forum Link
3. Zizhong F., (2004) Matching Character Variables by Sound: A closer look at SOUNDEX function and Sounds-Like Operator (=*), SUGI 072-29, SUGI 29 Proceedings, Montréal, Canada May 9-12, 2004 Link
4. SAS Institute (2012) Step-by-Step Programming with Base SAS® Software Link
5. Cormen T.H., Leiserson C.E., and Ronald L. Rivest (2009). Introduction to Algorithms (third edition, MIT Press, 2009) - Chapter 34: String matching, Link
6. Feldman R., Fresko M., Kinar Y. et. al., (1998) Text Mining at the Term Level; Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Second European Symposium, PKDD '98 Nantes, France, September 23–26, 1998 Proceedings pp. 65-73 Link.

ACKNOWLEDGMENTS

We place on record our gratitude to Professor Prabhat Jha, Director, Centre for Global Health Research at St Michaels Hospital, University of Toronto for facilitating this work. The support provided by Matt Malczewski, Communities Manager, SAS Canada and the encouragement extended by our fellow colleagues at Toronto Health User Group are gratefully acknowledged. A special word of thanks is due to both Ruth Croxford at Institute for Clinical Evaluative Sciences, Toronto and Dr. Mireille Gomes at Center for Global Health Research for their valuable inputs on the manuscript.

CONTACT INFORMATION

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

1. Wilson Suraweera, Bio Statistician, Centre for Global Health Research, Li-Ka Shing Knowledge Institute, St. Michael's Hospital, Toronto, Canada, E-mail: suraweeraw@smh.ca
2. Jaya Weerasooriya, Data Program Analyst, Ontario Ministry of Health and Long-Term Care, Toronto, Canada E-mail: jaya.weerasooriya@ontario.ca
3. Neil Fernando, Senior Analyst, Canadian Imperial Bank of Commerce (CIBC), Toronto, Canada Neil.Fernando@cibc.com