Paper 400-2013

Big Data Meets Text Mining

Zheng Zhao, Russell Albright, James Cox, and Alicia Bieringer
SAS Institute Inc.

ABSTRACT

Learning from your customers and your competitors has become a real possibility because of the massive amount of web and social media data available. However, this abundance of data requires significantly more time and computer memory to perform analytical tasks. This paper introduces high-performance text mining technology for SAS® High-Performance Analytics. Text parsing, text filtering, and dimension reduction are performed using the new HPTMINE and HPTMSCORE procedures in SAS® High-Performance Analytics Server and are accessed conveniently from within SAS® High-Performance Enterprise Miner™. The paper demonstrates and discusses the advantages of this new SAS® functionality and provides real-world examples of the types of performance improvements you can expect.

INTRODUCTION

Unstructured data account for more than 80% of enterprise data and are growing at an exponential annual rate of 60%. Many of the opportunities for analyzing unstructured data are attractive, but the volume of potential unstructured data can also be staggering:

- Google processes 11.4 billion queries per month.
- Recently Twitter announced that it was processing over half a billion tweets per day.
- Facebook reached a billion active users this past year and is averaging over a billion status updates per day.

This scale of activity hints at the number of opportunities for you to analyze, monitor, and predict what your customers are saying about you, your products and services, and your competitors. In addition to the web technologies in the previous list, unstructured content from forums, blogs, emails, and product review sites provides abundant sources of data. In many cases, you can capitalize on the volume of data from these sources only if you have the tools and the resources to deal with massive data.

A high-performance data mining project has been under way at SAS for several years. Initially, it implemented some key computational algorithms in a specific grid environment, and now it contains many algorithms that can run on a variety of hosts and grid environments. In 2012, SAS High-Performance Analytics Server 12.2 added high-performance text mining to the project.

SAS high-performance text mining provides full-spectrum support for text mining, including document parsing, term weighting and filtering, term-by-document matrix creation, dimensionality reduction via singular value decomposition (SVD), and scoring. It integrates the functionalities that are provided by multiple traditional SAS text mining procedures into two high-performance procedures, the HPTMINE and HPTMSCORE procedures. This integration simplifies programming and greatly improves computational efficiency. SAS high-performance text mining supports both symmetric multiprocessing (SMP) via multithreading and massively parallel processing (MPP) via distributed computing; both SMP and MPP enable you to analyze more data faster than ever before. High-performance text mining also supports various grid environments and distributed database systems and enables you to further boost the performance by using alongside-the-database processing to bring the computation to your data.

Despite its power, the SAS high-performance text mining interface is user-friendly and similar to the traditional SAS text mining interface. You can use your existing SAS programming skills to analyze your text data by using SAS procedures and macros. Alternatively, you can use the HP Text Miner node in SAS Enterprise Miner to develop repeatable and shareable process flows without needing SAS programming knowledge. As a key component of SAS High-Performance Analytics, SAS high-performance text mining has revolutionized the way that large-scale text data are used in predictive modeling for big-data analysis in processes for both model building and scoring.

This paper describes the basic components and architecture of the new SAS high-performance text mining technology and provides concrete examples to show how you can use SAS programming and the HP Text Miner node to tackle big text data and contribute to your predictive modeling projects.

SAS HIGH-PERFORMANCE TEXT MINING COMPONENTS

SAS high-performance text mining provides comprehensive support for effectively analyzing large-scale text data in a grid computing environment. The functionalities of SAS high-performance text mining are supported by 10 components in different levels, and its implementation supports a variety of hosts and grid environments. Despite its complexity and power, you will find its programming interface easy to use. You can also conveniently access the interface from SAS High-Performance Enterprise Miner to develop repeatable and shareable process flows to support predictive modeling. Figure 1 shows the SAS high-performance text mining components and their relationships.

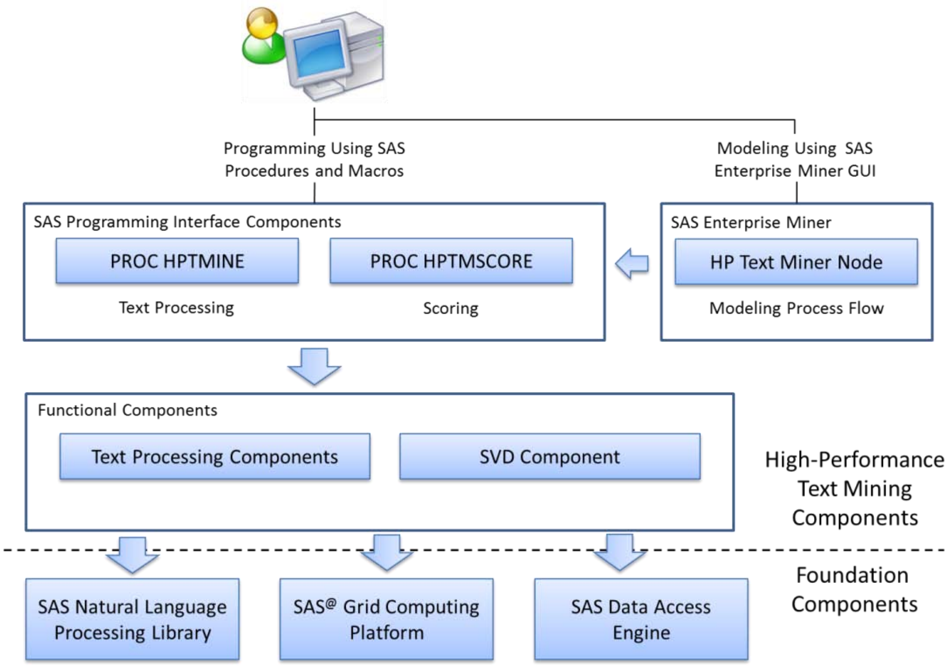

Programming Using SAS Procedures and Macros
Modeling Using SAS Enterprise Miner GUI
SAS Programming Interface Components
PROC HPTMINE
PROC HPTMSCORE
Text Processing
Scoring
SAS Enterprise Miner
HP Text Miner Node
Modeling Process Flow
Functional Components
Text Processing Components
SVD Component
High-Performance Text Mining Components
Foundation Components
SAS Natural Language Processing Library
SAS@ Grid Computing Platform
SAS Data Access Engine

Figure 1. Components of SAS High-Performance Text Mining

In general, the components of SAS high-performance text mining fall into three categories:

- **Functional components**: These components perform the primary functions of text mining, such as document parsing, term weighting and filtering, term-by-document matrix creation, and dimensionality reduction. These components are fully threaded and can perform distributed parallel computing by using the message passing interface (MPI) with the support of the SAS® Grid Computing platform. To achieve their goals, the text processing components also need to call a SAS natural language processing library for text parsing and the SAS data access engines for loading data onto different platforms.
- **SAS programming interface components**: These components wrap the functional components and provide an easy-to-use interface in the SAS programming environment.
- **SAS Enterprise Miner user interface component**: The SAS HP Text Miner node enables you to accomplish high-performance text mining from within SAS Enterprise Miner. This node uses the functions that are provided by the HPTMINE and HPTMSCORE procedures and enables you to conveniently bring text data into high-performance data mining process flows to improve the performance of predictive modeling.

The following sections show you how to use the programming interface that PROC HPTMINE and PROC HPTMSCORE provide to analyze large-scale text data on different platforms. You will see details about the functional components that support these procedures and about the GUI, which enables you to incorporate your text data for predictive modeling in SAS Enterprise Miner.

HIGH-PERFORMANCE TEXT MINING PROCEDURES

The HPTMINE and HPTMSCORE procedures are new high-performance procedures that enable SAS High-Performance Analytics Server to perform text mining. They support a wide range of fundamental text analysis features, which include tokenizing, stemming, part-of-speech tagging, noun group extraction, default or customized stop lists and start lists, entity parsing, multiword tokens, synonym lists, term weighting, term-by-document matrix creation, and dimension reduction by term filtering and singular value decomposition (SVD).

COMPARING TRADITIONAL AND HIGH-PERFORMANCE TEXT MINING PROCEDURES

PROC HPTMINE integrates the functions that are provided by the traditional text mining procedures (PROC TGPARSE, PROC TMUTIL, and PROC SPSVD), and PROC HPTMSCORE integrates the functionality that is provided by the DATA step function (TGSCORE) and the SPSVD procedure. Each new high-performance procedure achieves high efficiency and scalability through parallel processing both at the processor level by using multiple threads for symmetric multiprocessing (SMP) and at the machine level by using multiple machines for massively parallel processing (MPP). When these procedures run on an appliance that consists of a cluster of nodes, they also support various MPP modes, including client-data mode, alongside-the-database mode, alongside-HDFS mode, and alongside-LASR mode.

In the following example, the statements in the left and right columns show how traditional procedures and high-performance procedures, respectively, do text parsing and dimension reduction to process a document collection:

```
/* Parse the document collection */
proc tgparse data=documents_dataset
    out=parseOut key=key
    stemming=yes tagging=no
    entities=no ng=std
    stop=sashelp.engstop
var text;
run;

/* Run the TMUTIL procedure */
proc tmutil data=parseOut key=key;
    control init release;
    select reducef=4;
    weight termwgt=entropy cellwgt=log;
    output out=spsvdIn;
run;

/* Compute the SVD */
proc spsvd data=spsvdIn
    global=entropy local=binary
    max_k=50 res=med;
    row _termnum_;
    col _document_;
    entry _count_;
    output u=u s=s v=v;
run;

/* Score using tgscore function */
data doc_dataset_score;
   _document_ = _n_;
   rc=tgscore(text,"config","key",
              "spsvdIn_score");
   drop rc;
run;

/* Compute projects for documents */
proc spsvd data= spsvdIn_score in_u=u;
    row _termnum_;
    col _document_;
    entry _count_;
    output docpro=_docpro normdoc;
run;
```

```
/* Process the document collection */
proc hptmine data=documents_dataset;
    doc_id docid; var text;
    parse termwgt=entropy cellwgt=log
        stop=sashelp.engstop
        outconfig=config
        outterms=key;
    svd max_k=50 res=med
        svdu=u svdv=v svds=s;
run;
```

```
/* Score the document collection */
proc hptmscore data=doc_dataset_score
    terms=key config=config svdu=u
    svddocpro=_docpro keepvars=(_all_);
    doc_id docid; var text;
run;
```

When you use the traditional text mining technology, you need to call three procedures to process a document collection at training time. On the other hand, the SAS high-performance text mining technology requires only the HPTMINE procedure. Therefore, using the high-performance technology to process your text data enables you to make your code more compact and more maintainable. A similar compact representation is evident when you compare the score code that uses the traditional procedures (which require separate calls to parse documents and generate document projects) with the high-performance procedure (which requires only a single call to accomplish the same task). The traditional approach requires more communication time and resources because one procedure must store the intermediate results in a SAS data set[1] and then the next procedure must read in that data set in order to process it. When your text data set is large, storing and loading these intermediate results can be expensive because

---

[1] The HPTMINE and HPTMSCORE procedures also provide options that enable you to output intermediate results as SAS data sets.

of I/O operations. In contrast, when you use SAS high-performance text mining technology to process your text data, you do not need to store and load intermediate results as SAS data sets. Therefore, you can avoid the expensive I/O operations and improve the efficiency.

When you use the HPTMINE and HPTMSCORE procedures, you will find that many of the options in these procedures are similar to the options in the traditional SAS text mining procedures. For example, in the preceding code comparison, the TERMWGT=, CELLWGT=, STOP=, MAX_K=, and RES= options are used in both the traditional approach on the left and the high-performance approach on the right. The option values have the same meanings in the traditional and the high-performance procedures so that, as a beginning user of SAS high-performance text mining, you will be familiar with the features of the high-performance procedures and your learning curve will not be steep. This similarity also simplifies your migration from using the traditional text mining procedures to using the new high-performance text mining procedures.

PROC HPTMINE CODING OUTLINE

This section presents an outline for using the HPTMINE procedure to process a large-scale text data set on an appliance in alongside-the-database mode. This outline also shows you how to take the results from the HPTMINE procedure to build a predictive model.

The process of writing your own high-performance text mining code is straightforward, but it does involve understanding some new concepts that are related to working in a grid environment. The code in this section shows a complete predictive modeling process. The following steps configure the appliance for using a grid environment and then use the HPTMINE procedure to process the text data and extract the SVD dimensions, which are fed into the HPLOGISTIC procedure for predictive modeling:

1. Set up the appliance environment by using statements similar to the following:

   ```
   option set=GRIDHOST        = "hpa.sas.com";
   option set=GRIDINSTALLLOC = "/opt/TKGrid";
   option set=GRIDDATASERVER = "myserver";
   option set=GRIDMODE        = "sym";
   ```

   In these statements you specify the following options:

   - `GRIDHOST` identifies the domain name system (DNS) or IP address of the appliance node to which the SAS High-Performance Analytics Server connects to run in massively parallel processing mode.
   - `GRIDINSTALLLOC` identifies the directory in which the SAS High-Performance Analytics Server components are installed on the appliance.
   - `GRIDDATASERVER` identifies the database server on the appliance.
   - `GRIDMODE` specifies whether the procedures execute in symmetric or asymmetric mode.

   The statements in this step specify these options for an appliance whose DNS is `hpa.sas.com`, whose SAS High-Performance Analytics Server components are in the /`opt/TKGrid` directory, whose database server is named myserver, and whose execution mode is symmetric.

2. Gain access to the data by using a LIBNAME statement similar to one of the following, depending on the data source and the appliance that you are using:

   ```
   /** Greenplum appliance **/
   libname hptm greenplum user=foo password=bar database=hps server="myserver";

   /** Teradata appliance **/
   libname hptm teradata user=foo password=bar database=hps server="myserver";

   /** Oracle appliance **/
   libname hptm oracle user=foo password=bar path="myOralpath";

   /** Hive appliance **/
   libname hptm sasiohdp user=foo password=bar database=hps server="myserver"
   HDFS_TEMPDIR="mydir" config="myconfig.xml";

   /** LASR server **/
   libname hptm sasiola lasr="mypath\mylasr";

   /** Hadoop SASHDAT server **/
   libname hptm sashdat install="instPath" host="myserver" hdfs_path="mypath";
   ```

3. Use the `GRID_TEXTANALYTICS_BIN_LOC` macro to indicate where to find the language binary files that are installed on the appliance. These binary files are used by the text processing components of the HPTMINE procedure for parsing documents.

   ```
   %let GRID_TEXTANALYTICS_BIN_LOC=myTextBinPath;
   ```

4. Use the HPTMINE procedure to process your text data and extract the SVD dimensions from it:

   ```
   proc hptmine data=hptm.documents_dataset;
       doc_id docid; var text; target target_var;
       parse termwgt=mi cellwgt=log stop=sashelp.engstop;
       svd k=50 res=med p=25 outdocpro=hptm.docpro keepvars=(_all_);
   run;
   ```

   The `outdocpro=hptm.docpro keepvars=(_all_)` option in the SVD statement requests that the extracted SVD dimensions be merged with all the variables in the original data and written to the output data set `hptm.docpro`.

5. Use the HPLOGISTIC procedure to build a predictive model from the output of the HPTMINE procedure:

   ```
   proc hplogistic data=hptm.docpro;
       class airbag make model city;
       model airbag =col1-col50 make model city;
   run;
   ```

   The HPLOGISTIC procedure uses the `hptm.docpro` data set to build a predictive model for predicting whether a complaint is about a car's airbag. Like other SAS statistical procedures, the HPLOGISTIC procedure can handle only structured data. The HPTMINE procedure transforms your unstructured text data to a structured form that can be used as input to most SAS statistical procedures.

PROC HPTMINE PERFORMANCE OBSERVATIONS

You can expect to see some significant performance improvements when you transition from the traditional text mining procedures to the new SAS high-performance text mining procedures. The benefit derives not only from moving from a non-grid environment to a grid environment, but also because high-performance text mining procedures are multithreaded, thus providing a significant difference even on a single machine. The following example takes a data set that is large and consumes a lot of time when run by traditional procedures and then shows how quickly and efficiently the data set is processed both on a single machine with multiple threads and on the grid with multiple machines[2].

EVALUATION DATA

The data for this evaluation are records of safety-related defect complaints that were received by the National Highway Traffic Safety Administration (NHTSA) over the course of several years. This Consumer Complaints data set (nhtsa_cmp) includes 683,076 records, where each record is about a paragraph in length. For each record, the variable `cdescr` contains the content of a complaint, the variable `cmplid` is the ID of the record, and the variable `airbag` (the target predictor variable) indicates whether the complaint is related to the car's airbag.

PERFORMANCE OBSERVATIONS

Figure 2 compares the run times for processing the NHTSA's Consumer Complaints data set (nhtsa_cmp) by using the traditional text mining procedures versus the run times for the HPTMINE procedure. The results are obtained on a Windows server with two Intel Xeon E5-2667 CPUs[3] and 128 GB of memory. In the comparison, the HPTMINE procedure was tested in the SMP mode with one thread and eight threads, respectively.

When it uses only one thread, the HPTMINE procedure is about 35% faster than the traditional text mining procedures. This speed increase is largely because traditional text mining procedures need to communicate by writing their intermediate results as SAS data sets, whereas the HPTMINE procedure does not. When it uses eight threads, the

---

[2] Each machine also uses multiple threads to process its data.

[3] Each Intel Xeon E5-2667 CPU has six cores and 15M cache, and its maximum turbo frequency is 3.5 GHz.

HPTMINE procedure is about seven times faster than the traditional text mining procedures because of the power of parallel processing.

Figure 2 also shows that for processing a text data set, the document parsing phase dominates the run time. In the document parsing phase, natural language processing (NLP) techniques are used to extract meaningful information from natural language input. SAS high-performance text mining procedures can effectively parallelize the computation in this phase to improve the efficiency of text mining.

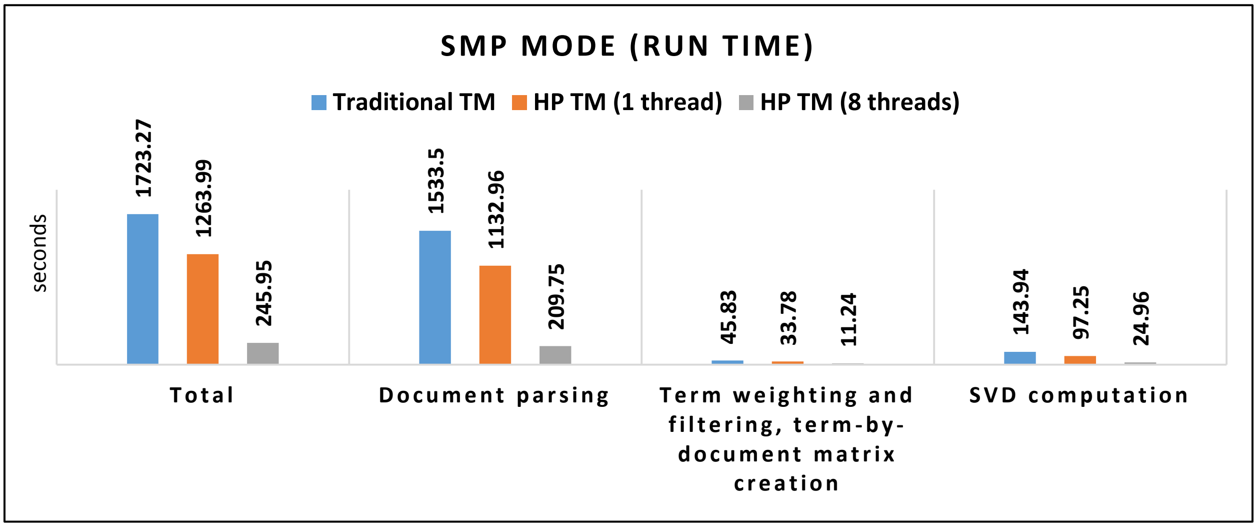


Figure 2. Run-Time Comparison of Traditional Text Mining Procedures versus PROC HPTMINE Procedure in SMP Mode

Figure 3 shows the run time for processing the nhtsa_cmp data set when the HPTMINE procedure is used in MPP mode with a varying number of grid nodes. The results are obtained on a cluster system that has 16 computing nodes. Each computing node has two Intel Xeon X5675 CPUs[4] and 98 GB of memory. The HPTMINE procedure uses all the cores on a computing node for processing. Figure 3 shows that when only one node is used, the HPTMINE procedure finishes processing in about 3 minutes. When four nodes are used, the HPTMINE procedure finishes in less than 1 minute. And with eight nodes, it finishes in just 37 seconds. The results also show that with more than eight nodes, the speed increase of the procedure is limited. The increase is limited because the computing nodes need to communicate and synchronize their results when they process text data in a grid environment. When the input data set is not big enough and the number of computing nodes is large, communication cost starts to offset the computing power that is gained by using more nodes. Therefore, to maximize the efficiency of your hardware usage, you need to coordinate the number of computing nodes that are used for processing text data with the size of your problems.

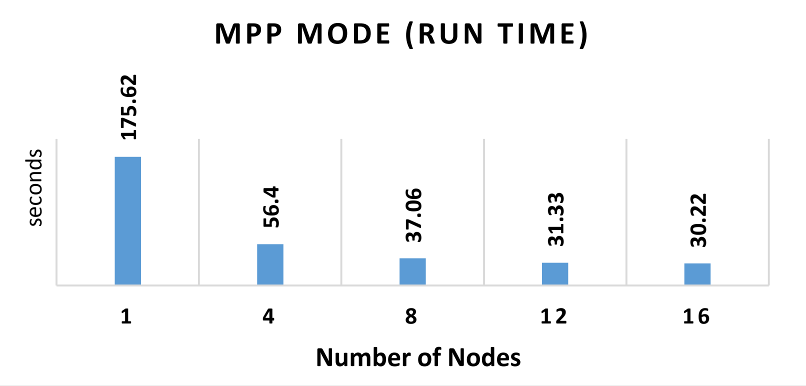


Figure 3. Run-Time Comparison for Running PROC HPTMINE in MPP Mode on Different Numbers of Nodes

---

[4] Each Intel Xeon X5675 CPU has six cores and 12M cache, and its maximum turbo frequency is 3.46 GHz.

In summary, the results show that a 30-minute task for the traditional text mining procedures can be reduced easily to less than a minute in a grid computing environment. This reduced run time means that you can be more effective in building predictive models to support your business goals. You can handle very large data sets and take advantage of their volume to obtain the most reliable models. You can also repeatedly retrain your models with different settings, so that you can optimize model performance in a short period of time.

TEXT-PROCESSING COMPONENTS

SAS high performance text mining has three components to process your unstructured text data set and generate the term-by-document matrix, which can be fed into the SVD component to compute the SVD dimensions. The three components are:

1. The document parsing component applies natural language processing (NLP) techniques to extract meaningful information from natural language input. The operations include document tokenizing, stemming, part-of-speech tagging, noun group extraction, processing default or customized stop lists or start lists, entity identification, and multiword terms handling.
2. The term handling component supports term accumulation, term filtering, and term weighting. The operations include quantifying each distinct term that appears in the input text data set, handling default or customized synonym list, and filtering (removing terms based on frequencies or stop lists) and weighting terms.
3. The text processing control component supports grid and threading control, manages intermediate results, controls input and output, and uses the results that are generated by the document parsing component and the term handling component to create the term-by-document matrix stored in a compressed form.

Figure 4 shows how your documents are processed by the text components in the SMP mode.

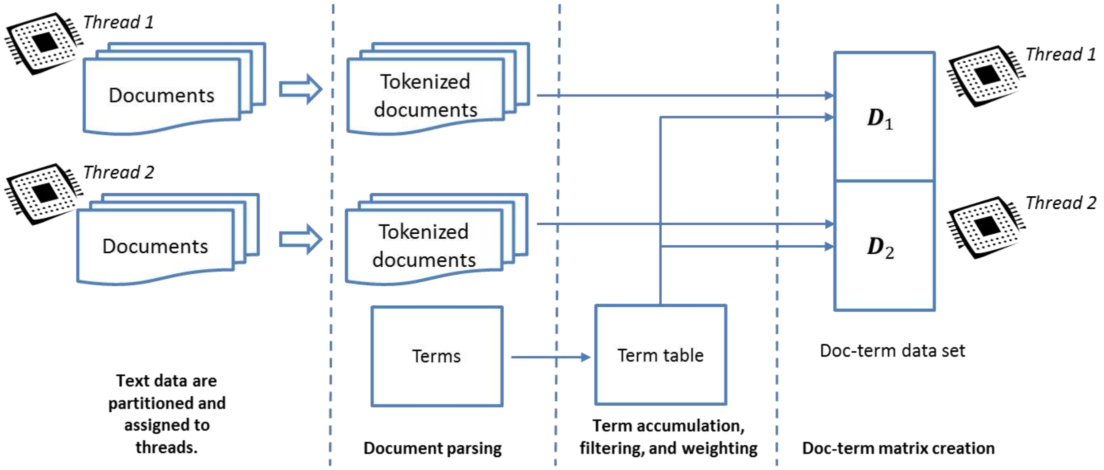


Figure 4. Text Processing in SMP Mode

Assume that two threads are used to process the data. First the text processing control component sets up two threads, and in each thread it reads documents from the input data and sends them to the document parsing component. The document parsing component parses the documents to generate the tokenized documents. And the terms identified in the input documents are stored in a central dictionary (an associative array), which is shared by the two threads. After all documents have been parsed, the resulting dictionary is sent to the term handling component, which accumulates, filters, and weights the terms. The result is stored as a term table. In each thread, the control component uses this term table and the tokenized documents to create a part of the document-by-term matrix. This process effectively parallelizes the steps of document parsing and document-by-term matrix creation, which are the most expensive steps in text processing.

Figure 5 shows how the text-processing components are used in MPP mode.

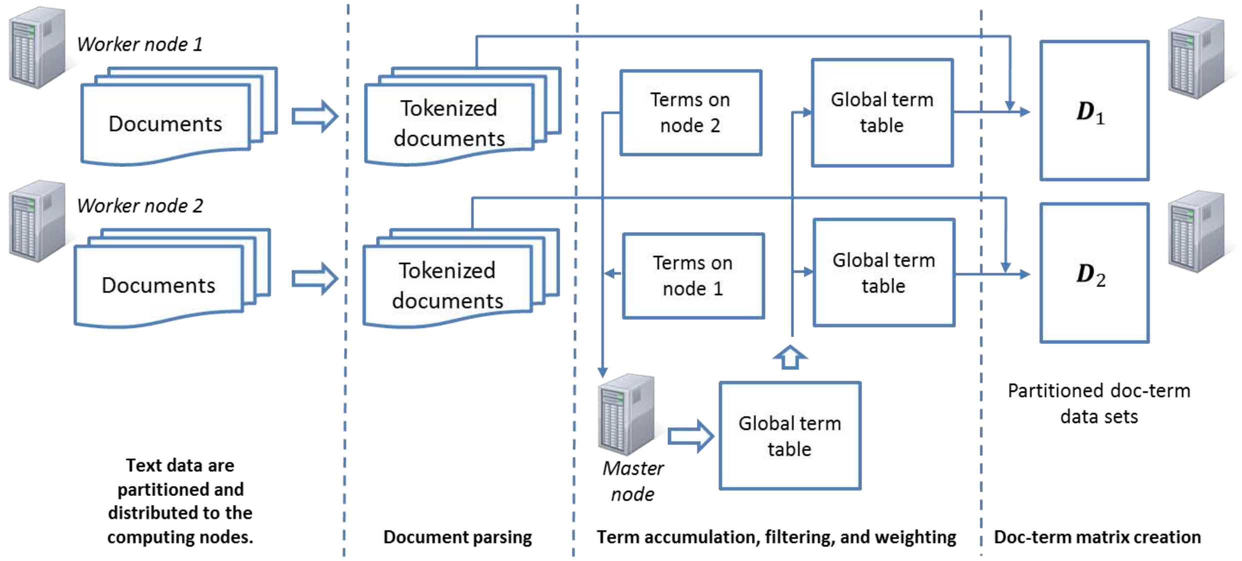


Figure 5. Text Processing in MPP Mode

The process is similar to the process for SMP mode. On each computing node (worker node), documents are loaded by the control component and parsed by using multiple threads. After the document-parsing step, the term-handling component sends the terms that are identified on each computing node to the master node to create a global term table. The global term table is then sent back to each computing node. The control component uses this table along with the tokenized documents to create a part of the document-by-term table on each node with multiple threads. As in SMP mode, the steps of document parsing and document-by-term matrix creation are effectively parallelized.

SVD COMPONENT

Singular value decomposition (SVD) is used to transform the high-dimensional, sparse term-vector representation of each document in your document collection into a reduced-dimensional, dense representation. The truncated SVD is a matrix factorization that enables you to approximate the original weighted term-document frequency matrix, $\boldsymbol{D}$, by using the product of three factors,

$$\boldsymbol{D} \approx \boldsymbol{U\Sigma V}^T$$

where $\boldsymbol{U}$ is an $m \times k$ matrix that contains a weight for each document at each of the k dimensions, $\boldsymbol{V}$ is an $\mathrm{n} \times k$ matrix that contains a weight for each term at each of the k dimensions, and $\boldsymbol{\Sigma}$ is a $k \times k$ diagonal matrix whose diagonal elements indicate the importance of the dimension. The factor $\boldsymbol{U}$ is the primary matrix that is needed to project a document into the k-dimensional subspace.

In the traditional SAS® Text Miner, the SVD is calculated using the SPSVD procedure, which is a single-threaded procedure that requires all the nonzero elements of the matrix $\boldsymbol{D}$ to be stored in a single chunk of memory. Storing $\boldsymbol{D}$ in this way has the advantage of making the computation fast, but it has the disadvantage of prohibiting the mining of document data sets that are too large to be stored in a single chunk of memory. You will eventually run out of memory when you attempt to solve larger and larger problems.

Unlike the text-processing components, the SVD component's primary bottleneck is space and its secondary bottleneck is time. This section focuses on how the SVD computation is altered from a single-threaded environment that uses PROC SPSVD to a multithreaded high-performance environment that uses PROC HPTMINE.

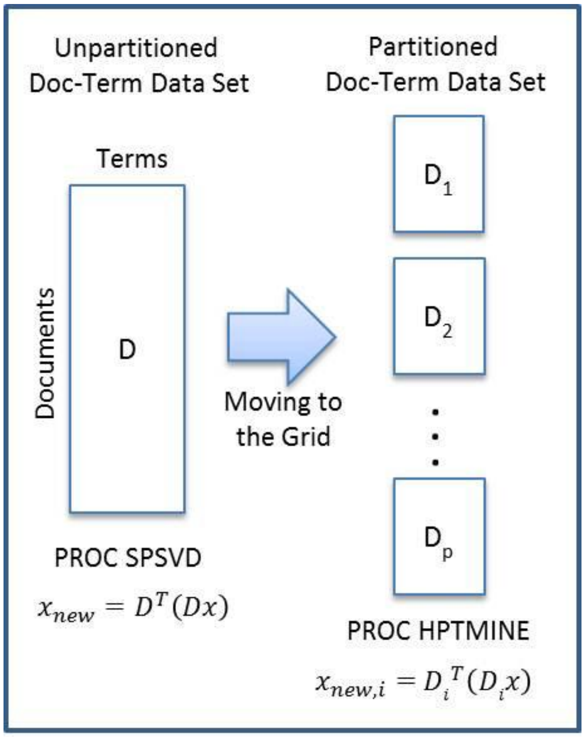


Figure 6. Two Approaches to SVD Computation

A fairly complex iterative algorithm computes the factors $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}^T$, and this section does not go into detail about every aspect of that computation. Instead, this section focuses on the particular piece of the computation that is addressed when you move to the grid environment. Fortunately, the part of the algorithm that is computationally expensive is also the part that requires the input data to be in memory. When this single aspect of the algorithm is moved to the grid, a small performance improvement is achieved and a large space bottleneck is removed. That crucial piece of the algorithm involves multiplying the original matrix $\boldsymbol{D}$ by a random vector $\boldsymbol{x}$ and then multiplying that result by $\boldsymbol{D}^T$: $\boldsymbol{x}_{new} = \boldsymbol{D}^T(\boldsymbol{D}\boldsymbol{x})$. Computing this multiplication only once would not be too expensive, but the algorithm requires that it is computed repeatedly. Eventually, the vector $\boldsymbol{x}_{new}$ approximates a key component for forming $\boldsymbol{U}$.

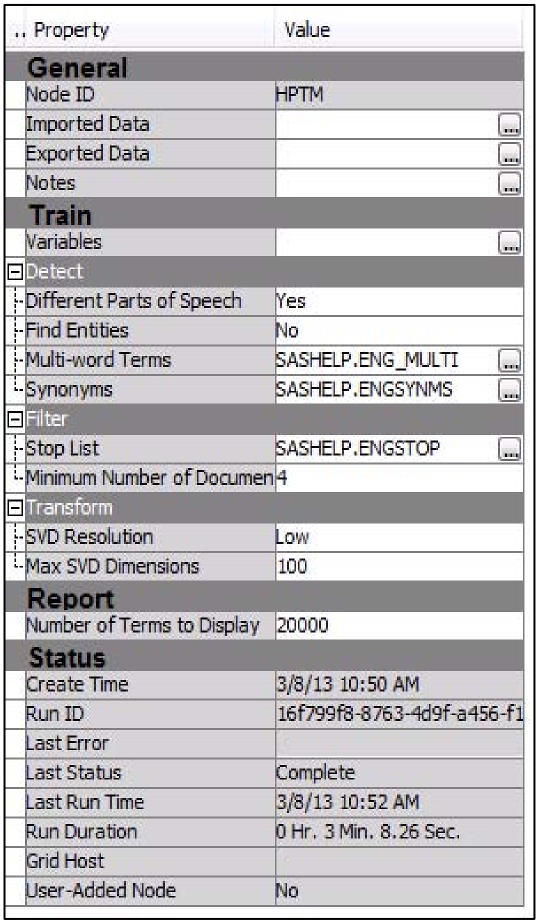
If you focus specifically on this multiplication computation, you notice two differences in the left and right sides in Figure 6. The first difference is that the huge chunk of memory that is allocated for $\boldsymbol{D}$ on the left becomes many small chunks that are individually allocated on different nodes of the grid on the right. These smaller chunks eliminate the memory bottleneck. The second difference is that the multiplication computation can be distributed by partitioning the data into rows and assigning a number of rows to each node. Each node computes the multiplication for its assigned rows, $\boldsymbol{x}_{new,i} = \boldsymbol{D}_i{}^T(\boldsymbol{D}_i\boldsymbol{x})$, and the results from each node are added to produce the updated vector $\boldsymbol{x}_{new}$:

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{new,1} + \boldsymbol{x}_{new,2} + \cdots + \boldsymbol{x}_{new,p}$$

Not all parts of the algorithm were distributed multiple nodes. Instead only the portion of the algorithm that causes the bottleneck to improving the performance was distributed. The end result is a huge gain in the size of the problems that can be factored with the SVD and, in addition, a minor performance increase.

SAS HIGH-PERFORMANCE TEXT MINER

SAS High-Performance Text Miner is a plug-in for SAS Enterprise Miner and part of the SAS High-Performance Analytical Server. You can access it through the HP Text Miner node on the HPDM tab of SAS Enterprise Miner. The node calls the HPTMINE and HPTMSCORE procedures to process text data and enables you to develop repeatable and shareable process flows for predictive modeling that use your unstructured text data in Enterprise Miner.



Property	Value
General	
Node ID	HPTM
Imported Data	
Exported Data	
Notes	
Train	
Variables	
⊟Detect	
Different Parts of Speech	Yes
Find Entities	No
Multi-word Terms	SASHELP.ENG_MULTI
Synonyms	SASHELP.ENGSYNMS
⊟Filter	
Stop List	SASHELP.ENGSTOP
Minimum Number of Documen	4
⊟Transform	
SVD Resolution	Low
Max SVD Dimensions	100
Report	
Number of Terms to Display	20000
Status	
Create Time	3/8/13 10:50 AM
Run ID	16f799f8-8763-4d9f-a456-f1
Last Error	
Last Status	Complete
Last Run Time	3/8/13 10:52 AM
Run Duration	0 Hr. 3 Min. 8.26 Sec.
Grid Host	
User-Added Node	No

Figure 7. HP Text Miner Node Properties

HP TEXT MINER NODE PROPERTIES

The HP Text Miner node integrates the functionalities that are provided by the traditional Text Miner nodes (the Text Parsing, Text Filter, and Text Cluster nodes) into a single node. Figure 7 shows the properties of the HP Text Miner node. The following settings are specific to the HP Text Miner node:

Different Parts of Speech specifies whether to identify each term's part of speech.

Find Entities specifies whether to extract entities in text parsing. If you select "Yes," the ENTITIES=STD option is used in both PROC HPTMINE and PROC HPTMSCORE for entity extraction when they parse the documents.

Multi-word Terms specifies a data set that contains multiple-word terms for text parsing.

Synonyms specifies a data set that contains user-defined synonyms for text parsing.

Stop List specifies a data set that contains the stop terms. All the terms in this data set are dropped in text parsing**.**

Minimum Number of Documents specifies a cutoff value. In text parsing, each term that has a frequency less than the cutoff value is excluded from the term-by-document matrix.

SVD Resolution specifies the resolution to be used when computing the SVD dimensions.

Max SVD Dimensions specifies the maximum number of SVD dimensions to be calculated.

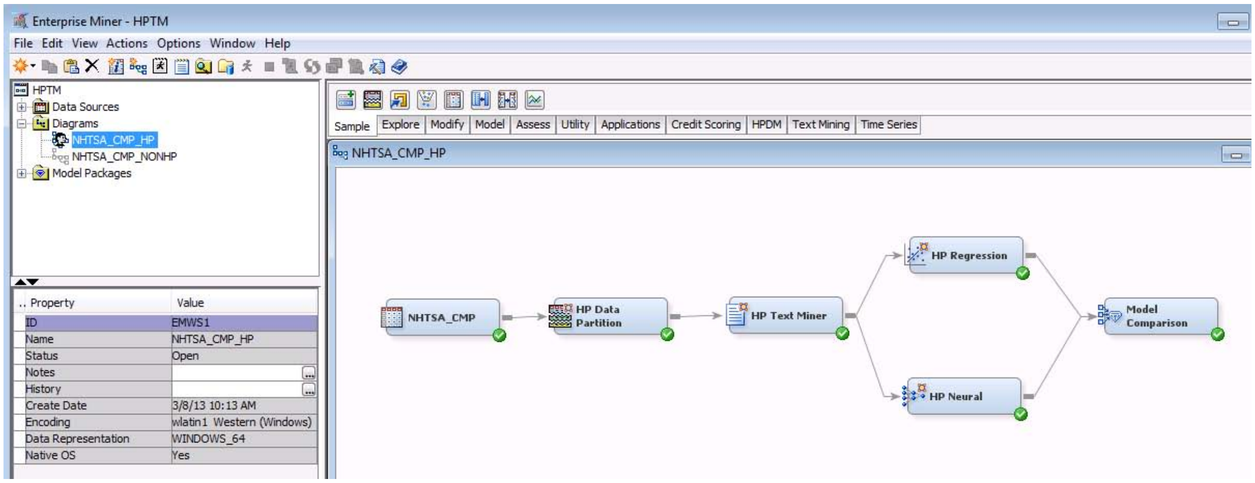
Number of Terms to Display specifies the maximum number of terms to display in the results viewer of the HP Text Miner node.

The default value for this property is 20,000. You can also set it to “All,” but the term list is usually very large for a large text corpus. If you specify “All” for this property, loading the term list might take a significant amount of time.

As on all other nodes in Enterprise Miner, you can also view the imported and exported data sets and node status on the property sheet of the HP Text Miner node.

PREDICTIVE MODELING EXAMPLE THAT USES THE HP TEXT MINER NODE

Figure 8 shows a predictive modeling example that uses the HP Text Miner node. In this example, the input text data set is first partitioned to generate the training and validation data sets by using the HP Data Partition node. The data sets are then fed into the HP Text Miner node for parsing documents, extracting the SVD dimension, and creating an output data set by merging the extracted SVD dimensions with the original variables in the input data sets. The resulting data set is fed into the HP Regression node and the HP Neural node to build predictive models. Finally, the Model Comparison node compares the performance of the models that are generated by the HP Regression node and the HP Neural node.

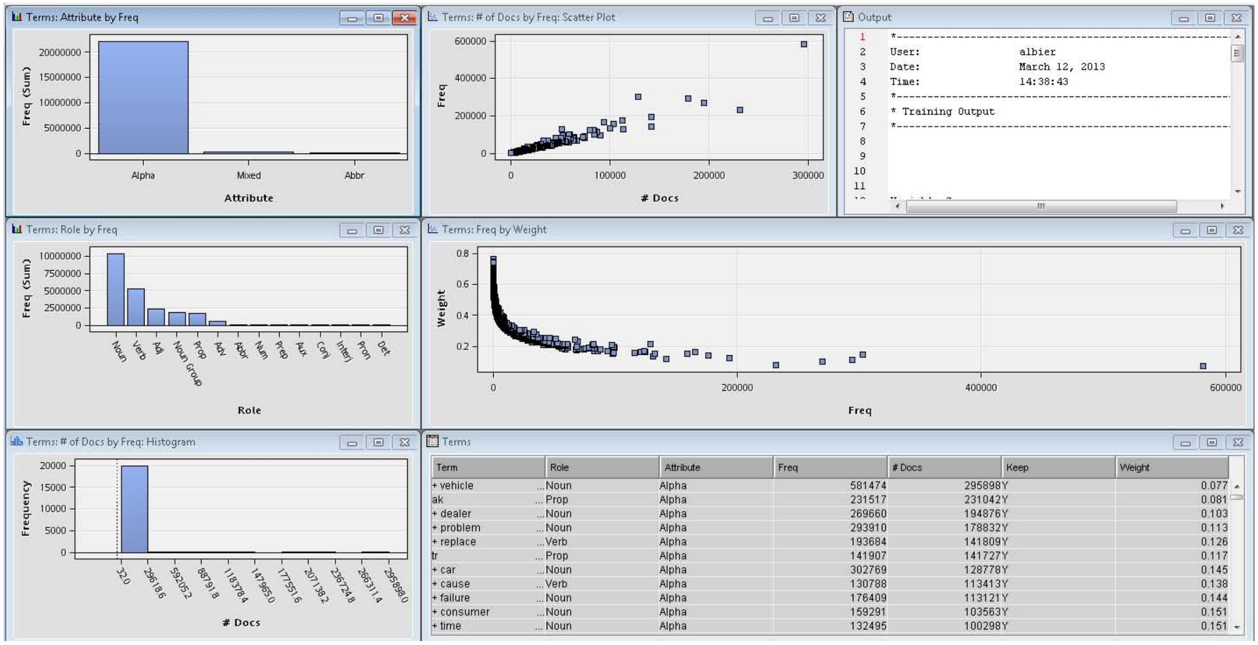


Property	Value
ID	EMWS1
Name	NHTSA_CMP_HP
Status	Open
Notes	
History	
Create Date	3/8/13 10:13 AM
Encoding	wlatin1 Western (Windows)
Data Representation	WINDOWS_64
Native OS	Yes

Figure 8. Predictive Modeling in the HP Text Miner Node

To run the HP Text Miner node in a grid environment, you must specify the grid information in the Project Start Code property of your project. This information includes the GRIDHOST, GRIDINSTALLLOC, GRIDDATASERVER, and GRIDMODE options for specifying your grid environment; the LIBNAME statement for accessing your data; and the GRID_TEXTANALYTICS_BIN_LOC macro for indicating the installation location of the language binary files. For your input data set, the HP Text Miner node also requires you to specify a variable that has the role TEXT and a unique ID variable that has the role KEY.

Figure 9 shows the result diagrams that are displayed by the HP Text Miner node. Just as in the traditional Text Miner node, the report provides various statistics about the terms that appear throughout the document collection. These diagrams are generated from a subset of all the terms identified in your text data (20,000 by default) unless you change the Number of Terms to Display property in the HP Text Miner node’s property sheet.



```
*-----------------------------------------------------------------
User:                    albier
Date:                    March 12, 2013
Time:                    14:38:43
*-----------------------------------------------------------------
* Training Output
*-----------------------------------------------------------------
```

Term	Role	Attribute	Freq	# Docs	Keep	Weight
+ vehicle	...Noun	Alpha	581474	295898	Y	0.077
ak	... Prop	Alpha	231517	231042	Y	0.081
+ dealer	...Noun	Alpha	269660	194876	Y	0.103
+ problem	...Noun	Alpha	293910	178832	Y	0.113
+ replace	...Verb	Alpha	193684	141809	Y	0.126
tr	... Prop	Alpha	141907	141727	Y	0.117
+ car	...Noun	Alpha	302769	128778	Y	0.145
+ cause	...Verb	Alpha	130788	113413	Y	0.138
+ failure	...Noun	Alpha	176409	113121	Y	0.144
+ consumer	...Noun	Alpha	159291	103563	Y	0.151
+ time	... Noun	Alpha	132495	100298	Y	0.151

Figure 9. Result Diagrams of the HP Text Miner Node

PERFORMANCE OBSERVATIONS

Figure 10 shows the run time for building predictive model from the nhtsa_cmp data by using both the traditional Text Miner nodes and the HP Text Miner node. For the traditional nodes, the results are obtained on a Windows server that has two Intel Xeon E5-2667 CPUs and 128 GB of memory. For the HP nodes, the results are obtained on a cluster system that has 16 computing nodes. Each computing node has two Intel Xeon X5670 CPUs[5] and 64 GB of memory.

The results show that, for both text mining and predictive modeling, the HP nodes are about 10 times faster than the traditional nodes. This demonstrates the huge speed increase that the HP Text Mining nodes bring to your predictive modeling process when you use SAS Enterprise Miner. Figure 10 also shows that summarizing results and generating reports (the "others" category in Figure 10) require a similar amount of time (about 1 minute) from both traditional nodes and HP nodes. The reason is that these operations are performed using single threads in both cases.

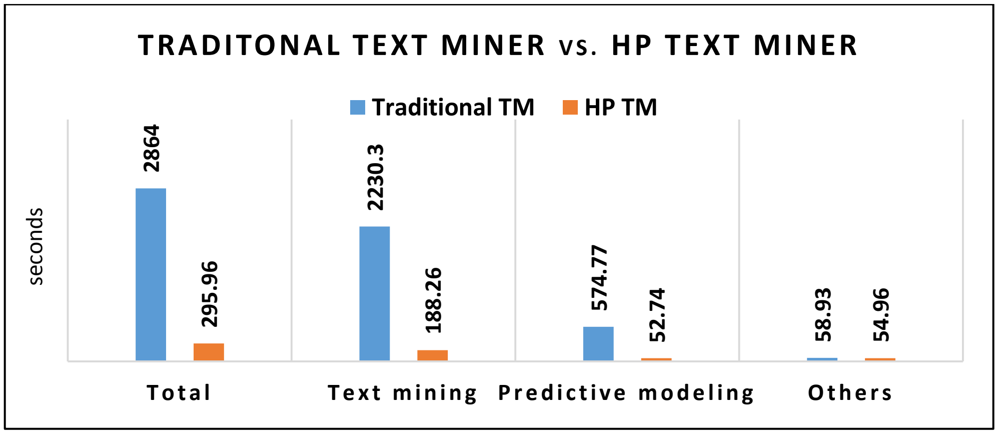


Figure 10. Run-Time Comparison of Traditional Text Miner Nodes and HP Text Miner Node

---

[5] Each Intel Xeon X5675 CPU has six cores and 12M cache, and its maximum turbo frequency is 3.33 GHz.

CONCLUSION

Pressing needs to process greater amounts of unstructured text data in less time to assist predictive modeling in a data mining environment are met by SAS high-performance text mining techniques. These techniques enable SAS users to continue to work in the environments that they are familiar with and, at the same time, to benefit from the computation power that is provided by SAS High-Performance Analytics. The NHTSA’s Consumer Complaints example shows how users can work effectively in SAS Enterprise Miner. The example fully demonstrates the great speed increase that SAS High-Performance Text Mining brings to your text mining and predictive modeling applications.

ACKNOWLEDGMENTS

The authors would like to thank Anne Baxter and Ed Huddleston for their editorial contributions.

RECOMMENDED READING

Albright, R. 2004. *Taming Text with the SVD.* Cary, NC: SAS Institute Inc. Available at *ftp.sas.com/techsup/download/EMiner/TamingTextwiththeSVD.pdf*.

SAS Institute Inc. 2012. *SAS Enterprise Miner High-Performance Data Mining Node Reference for SAS 9.3.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2013. *SAS High-Performance Analytics Server 12.2: User’s Guide.* Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Zheng Zhao
SAS Institute Inc.
SAS Campus Drive
Cary, NC, 27513
Zheng.Zhao@sas.com

Russell Albright
SAS Institute Inc.
SAS Campus Drive
Cary, NC, 27513
Russell.Albright@sas.com

James Cox
SAS Institute Inc.
SAS Campus Drive
Cary, NC, 27513
James.Cox@sas.com

Alicia Bieringer
SAS Institute Inc.
SAS Campus Drive
Cary, NC, 27513
Alicia.Bieringer@sas.com