Technical Paper

Enabling High-Performance Text Mining Procedures

Effectively translate parsing and SVD procedural code to take advantage of SAS multi-core, in-memory server processing.

Table of Contents

Introduction

Two procedures are now available with SAS® Text Miner 12.3[1] to enable High-Performance processing namely, HPTMINE and HPTMSCORE. This option allows for high performance execution using SAS® code specification. Both of these high performance text mining procedures support a wide range of fundamental text analysis capabilities, including tokenizing, stemming, part-of-speech tagging, noun group extraction, default or customized 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).

Singular Value Decomposition (SVD)

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, numeric representation. The truncated SVD is a matrix factorization that enables the approximation the original weighted term-document frequency matrix, $\boldsymbol{D}$, by using the product of three factors,

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

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 $n \times k$ 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.

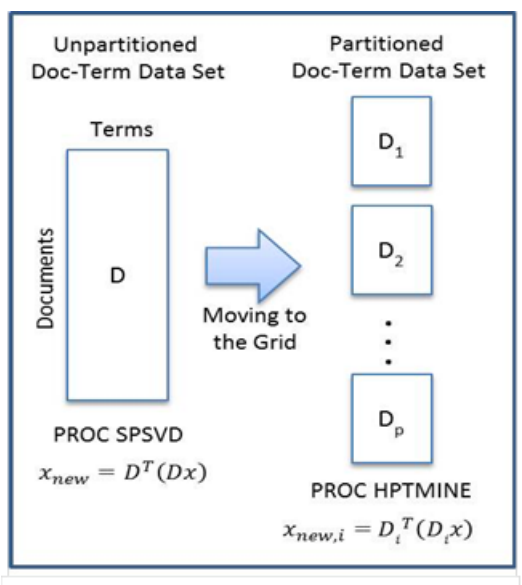

Figure 1: Two Different Approaches to SVD Computation

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, as depicted in Figure 1. Storing $\boldsymbol{D}$ in this way has the advantage of making the computation fast, but it has the disadvantage of prohibiting the mining of documents that are too large to be stored in a single chunk of memory. Eventually, and with big data especially, you'll run out of memory for solving more complex, larger problems.

Unlike the text-processing components, which are compute intensive calculations, the SVD bottleneck is space, with a secondary bottleneck of time.

A fairly complex iterative algorithm computes the factors $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}^T$. 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 a grid environment, the size of the data that can be readily processed is removed – the space issue is removed. That crucial piece of the algorithm involves multiplying the

[1] SAS® Text Miner 12.3 operates on SAS® 9.4, and is an add-on module to SAS® Enterprise Miner™

original matrix $\boldsymbol{D}$ by a random vector $\boldsymbol{x}$ and then multiplying that result by $\boldsymbol{D^T}$: $\boldsymbol{x_{new} = D^T(Dx)}$. Computing this multiplication only once would not be too computationally expensive, but the algorithm requires that it be repeatedly computed. Eventually, the vector $\boldsymbol{x_{new}}$ approximates a key component for forming $\boldsymbol{U}$.

In examining this multiplicative computation, as does Figure 1, it is apparent 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 a grid environment, on the right. Breaking this down eliminates the memory bottleneck. In this now partitioned document term data set, the multiplication computation can be distributed across the grid nodes, partitioning the data into groups of rows and assigning a number of rows to each node. Each node then computes the multiplication for its assigned rows, $\boldsymbol{x_{new,i} = D_i{}^T(D_i x)}$, and the results from each are then added to produce the updated vector $\boldsymbol{x_{new}}$:

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

So, while not all parts of the SVD algorithm are distributed across the nodes, the portion of the algorithm that causes the primary bottleneck issue, improving performance. The end result is a huge gain in the size of the problems that can be factored with the SVD calculation, in addition to an incremental improvement in run-time performance gains.

Comparing procedural definitions

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 of these procedures achieves high efficiency and scalability with 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)[2].

When we compare the coding specification of traditional text mining procedures relative to these high performance procedures, as depicted in Figure 2 we can see that traditional text mining statements call the three procedures needed to process a documentation collection[3] at training time. On the other hand, as illustrated on the right, the corresponding high-performance version only requires the HPTMINE procedure. The code specification is more compact under a high performance text mining paradigm, and so it is also easier to maintain.

Traditional SAS Text Mining procedures

```
/* 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;
```

SAS High-Performance Text Mining procedures

```
/* 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;
```

[2] SMP mode for executing high performance text mining procedures on a multi-core single machine is included with SAS® Text Miner. Scaling to MPP mode, for high performance computing across multiple machines is a separately licensed product (SAS® High Performance Text Mining, and which is accessible from the GUI interface of SAS® High Performance Data Mining).

[3] These are text parsing (proc tgparse), a dimension reduction procedure (proc tmutil), and the SVD computation (proc spsvd)

```
/* 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;
```

```
/* 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;
```

Figure 2: Traditional text mining statements (left) and comparable functionality written as high performance text mining statements (right), available in SAS® Text Miner 12.3.

Furthermore, the traditional approach requires both more communication time and resources simply because each procedure stores the intermediate results in a SAS data set[4] which is then read by the next procedure in order to process the associated data. When text data is large, storing, loading and reading each of these intermediate results can be computationally expensive due to these I/O operations. In contrast, high performance text mining doesn't need to store, load and read intermediate results as SAS data sets, improving efficiencies by eliminating these compute intensive I/O operations.

In the HPTMINE and HPTMSCORE procedures, similar specifications to traditional SAS® text mining procedures are available to customize the analysis. For example, we can see in Figure 2 the TERMWGT=, CELLWGT=, STOP=, MAX_K=, and RES= options are used in both the traditional definition and the high-performance approach. The option values also have the same meanings in both specifications. This similarity also simplifies migration from traditional text mining procedures to the high-performance text mining procedures.

Tips for High-Performance Text Mining Code Specification

In the alongside-the-database mode, the HPTMINE procedure can also process large-scale text collection on a hardware appliance. The process of writing your own high-performance text mining code is fairly straightforward, yet it does require an understanding of perhaps some new concepts that relate to multicore processor environments. In the following, the first three steps refer to hardware specifications needed when the high-performance text mining procedures are run in a distributed mode, across multi-threaded servers and are not required to be specified if the processing is occurring on a single, multiprocessor machine. The steps below outline how to configure the appliance for a distributed processing environment, and then use the HPTMINE procedure to process the text data and extract the SVD dimensions, then used by the HPLOGISTIC procedure (as an example) for predictive modeling.

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

Step 1. Set up the hardware appliance environment[5] by using statements similar to:

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

Within the statements listed in step 1, the following options are specified.

- `GRIDHOST` identifies the domain name system (DNS) or IP address of the hardware appliance node the SAS High-Performance Analytics Server connects to, in order to run in massively parallel processing mode.
- `GRIDINSTALLLOC` defines the directory in which the server components are installed on the appliance.
- `GRIDDATASERVER` specifies the database server associated with the appliance.
- `GRIDMODE` determines whether the procedures execute in symmetric or asymmetric processing[6] mode.

The above relate to an appliance whose DNS is `hpa.sas.com`, and which the server components are in the `/opt/TKGrid` directory. The database server specified in this example is named myserver, and symmetric mode processing is defined.

Step 2. Gain access to the data by using a LIBNAME statement similar to one of the following. The variations depicted below reflect different example scenarios for both the data source and the appliance being used.

```
/** 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";
```

Step 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. SAS parsing is dictionary-based, so the NLP functions are done in the native language.

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

Step 4. Specify the HPTMINE procedure to process the text data and extract the SVD dimensions.

```
proc hptmine data=hptm.documents_dataset;
    doc_id docid; var text; target target_var;
    parse termwgt=mi cellwgt=log stop=sashelp.engstop;
```

[5] Supported hardware appliance for SAS® High Performance Analytics can be found at: http://www.sas.com/software/high-performance-analytics/in-memory-analytics/analytics.html#section=5

[6] In the symmetric processing (SMP) mode, the data appliance and the computing appliance must be the same appliance whereas with asymmetric processing (and MPP mode) the data appliance and computing appliance can be different appliances.

```
    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`.

Step 5. With the HPLOGISTIC procedure, build a predictive model using 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, for example, whether a complaint is about a car 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 (via the SVD definition) that can be used as input to most SAS® statistical and data mining procedures.

Conclusion

The ever increasing need to process greater amounts of unstructured text data in less time to improve predictive modeling are addressed with high performance enablement of text mining methods. This revolutionary text analysis capability enables organizations to continue to work in the SAS® analytical environments that they are familiar with and, at the same time, to benefit from the computation power that is provided by taking advantage of in-memory analytic processing. The run-time gains affiliated with processing data in a high performance environment ensures that no big data is too big to ignore.