Paper SAS4880-2016

Running Text Analytics Models in Hadoop

David Bultman and Adam Pilz, SAS Institute Inc.

ABSTRACT

Hadoop took the world by storm as a cost-efficient software framework for storing data. Today, enterprises store large amounts of documents or big data, representing various aspects of their business in Hadoop for later analysis. Information Technology (IT) professionals are averse to moving this data out of Hadoop for the purpose of applying analytical models due to excessive network resource consumption. SAS recognizes that Hadoop is more than just a data store; it's an analytics platform. In this paper, we demonstrate how IT professionals can deploy and run text analytics models, generated by SAS® Contextual Analysis, in Hadoop.

INTRODUCTION

Enterprises store terabytes, and sometimes petabytes, of documents representing various aspects of their business in Hadoop. The documents represent client interactions, product reviews, call center logs, emails, blogs, tweets, customer reviews, and other forms of electronic text. Often the text, if analyzed, reveals information about the firm's products, the products of competitors, and other insightful information to improve business operations and performance. Text analytics is the mechanism that enables firms to automate identification of topics, entities, facts, and events, coupled with sentiment and other subjective information.

In order to analyze the volume of data without consuming an excessive amount of network resources, IT professionals seek tools that empower them to deploy analytical algorithms inside of their Hadoop infrastructure and avoid any movement of data. The combination of SAS® Contextual Analysis and SAS® In-Database Code Accelerator for Hadoop makes this possible.

This paper is organized into following sections:

- Background—Provides a general description of SAS Contextual Analysis and how the SAS In-Database Code Accelerator for Hadoop works in conjunction with SAS® Embedded Process
- Prerequisites—Helps you make sure you have everything in place to run text analytics models in Hadoop
- Text Analytics Scoring in Hadoop—Gives you specific recommendations for configuration and code examples in order to prepare you for scoring your data
- Output—Shows what you can expect after scoring your data
- Benefits—Outlines the benefits of using SAS Contextual Analysis and Hadoop in your organization.

BACKGROUND

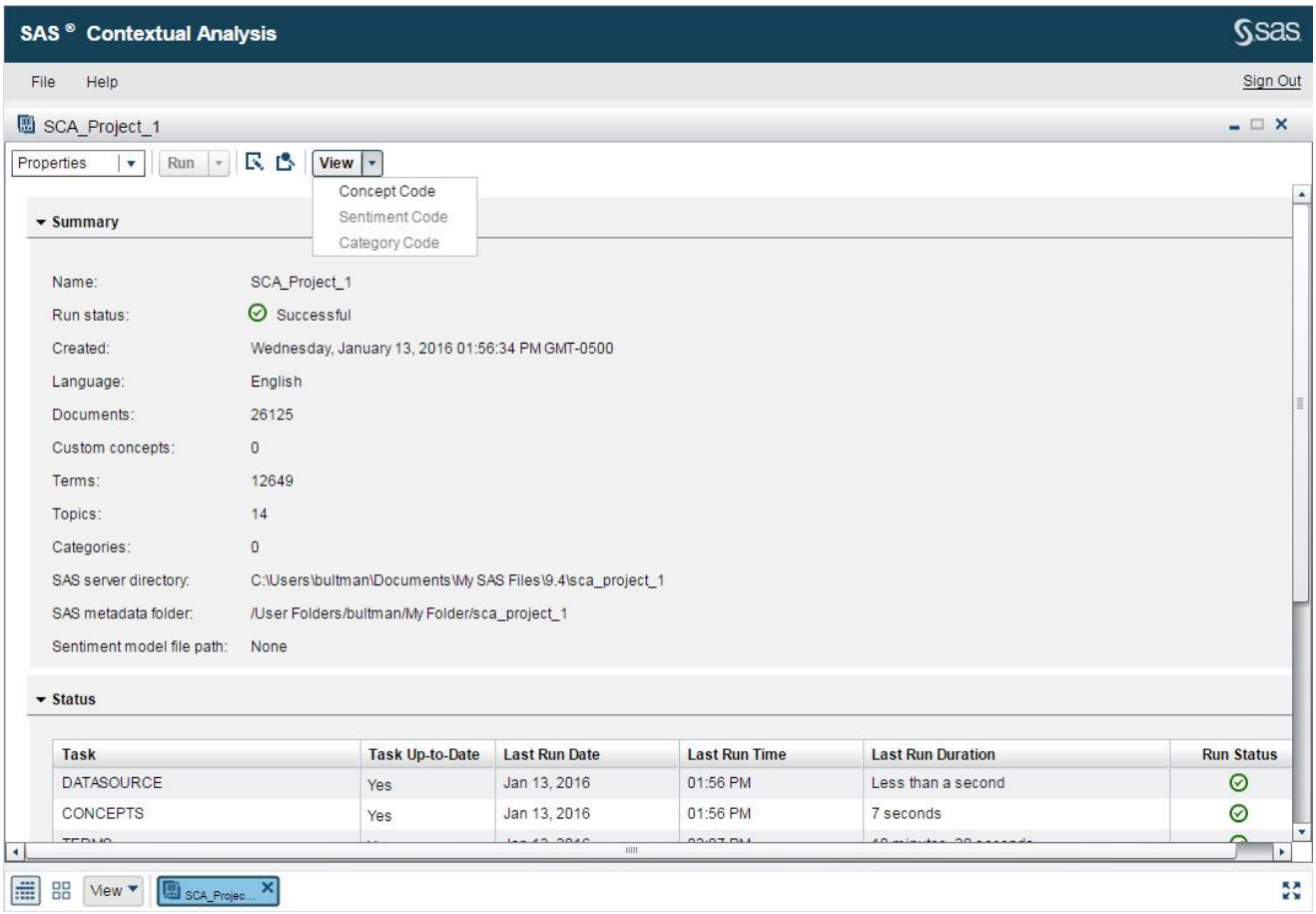
OVERVIEW OF SAS CONTEXTUAL ANALYSIS

SAS Contextual Analysis is a web-based application that uses natural language processing and machine learning to derive insights from textual data through topic identification, categorization, entity and fact extraction, and sentiment analysis—all from a single interface. Using this application, you can build models (based on training documents) and create taxonomies and rule sets to analyze documents. You can then customize your models for your business domain in order to realize the value of your text-based data.

At the end of the modeling process, SAS Contextual Analysis generates DATA step 2 (DS2) code and binary models for scoring text data. The score code can be retrieved from the View drop-down menu, which is first seen on the Properties page of any SAS Contextual Analysis project. SAS Contextual Analysis generates three types of DS2 score code and models corresponding to categorization, concept

extraction, and sentiment analysis. The DS2 code can be run within a SAS environment such as SAS® Studio or modified to run within your Hadoop cluster using SAS In-Database Code Accelerator for Hadoop. The binary models represent the rule sets for categorization (file extension: .mco), concepts (file extension: .li) and sentiment (file extension: .sam) taxonomies, which are highly optimized to apply all rules in parallel.

Display 1 shows the code generation options in SAS Contextual Analysis.



Display 1. SAS Contextual Analysis Score Code Generation

OVERVIEW OF SAS CODE ACCELERATOR

The SAS In-Database Code Accelerator for Hadoop enables you to publish a DS2 thread program and its associated files to the database and execute that thread program in parallel within SAS® Embedded Process. Benefits of in-database processing include reduced data movement, faster run-time, and the ability to leverage existing data warehousing investments without having to copy data to a secondary location for processing. Examples of thread programs include large transpositions, computationally complex programs, scoring models, and BY-group processing. For the scope of this paper, we focus only on SAS In-Database Code Accelerator for Hadoop and SAS Text Analytics models.

You can submit your DS2 program from a SAS client such as SAS Studio using Base SAS®. JProxy is used by Base SAS to submit a MapReduce job for execution. The MapReduce job starts on the Hadoop cluster. The data is processed on each data node where the data resides and the output result sets are given to the mapper. Output records are written to an output file in Hadoop Distributed File System (HDFS). Optionally, you can read the output results back to Base SAS. With in-database processing, data is distributed on different data partitions. Each MapReduce thread is running inside the database and has access to its own data partition, as illustrated in Figure 1.

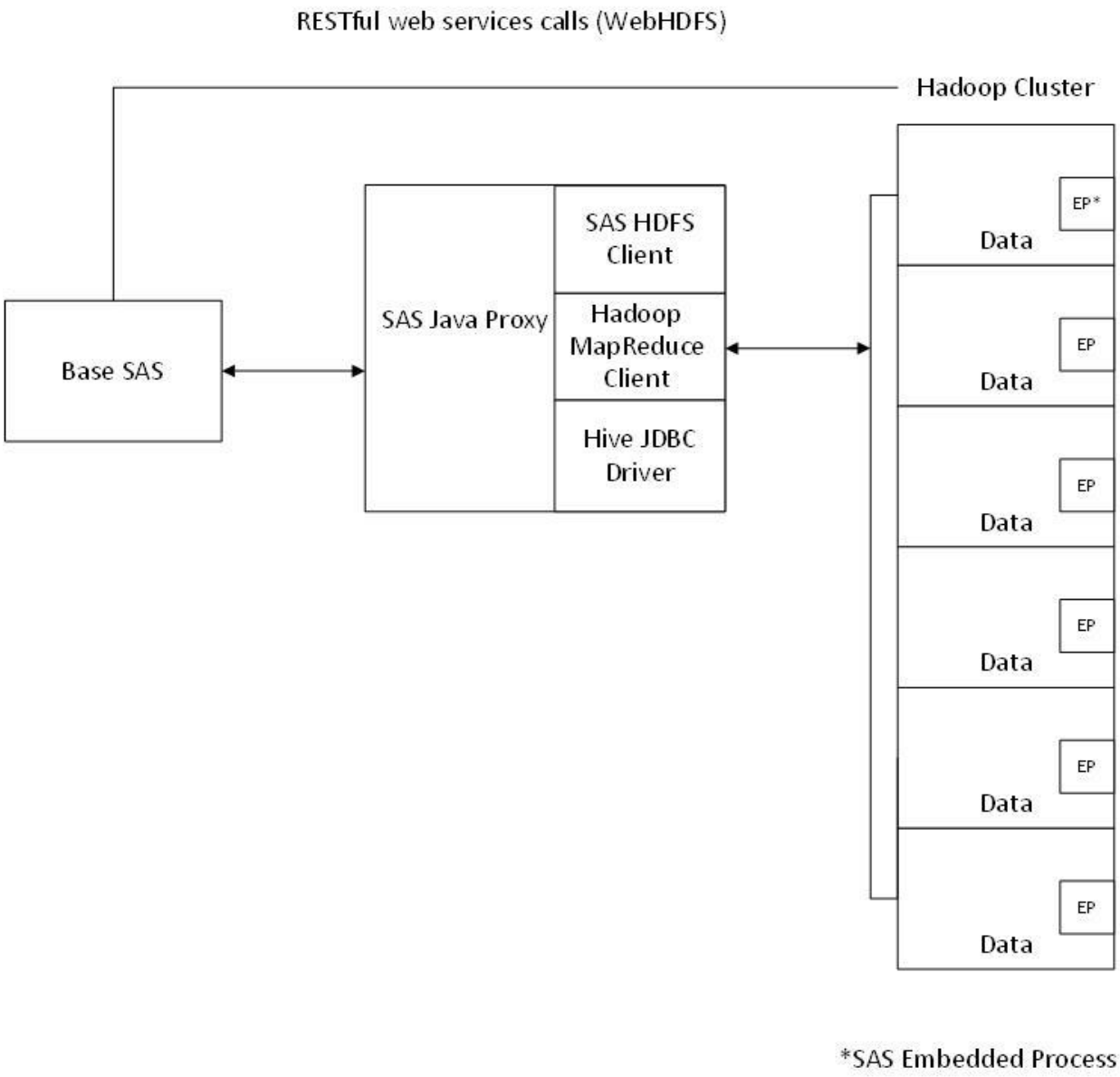


Figure 1. Communication between SAS and the Hadoop Cluster

PREREQUISITES

SAS EMBEDDED PROCESS

The SAS Embedded Process is a portable, lightweight execution container for SAS DS2 that makes SAS portable and deployable on a variety of platforms. SAS DS2 code processes data directly in the cluster, orchestrated by Apache Hadoop MapReduce and resources managed by YARN. An add-on to SAS Embedded Process is a SAS Text Analytics component, which allows you to score sentiment, category, and concept models in Hadoop.

The SAS Text Analytics component consists of additional natural language processing threaded-kernel (TK) extensions placed in SAS Embedded Process. The same TK extensions are installed for SAS Contextual Analysis and SAS® Text Miner. Therefore, your DS2 API (see code that follows) is identical to the API used under the hood for all SAS Text Analytics products. An additional component of the add-on

for SAS Text Analytics are the supporting binary files. These files are referenced in your DS2 code, giving you flexibility regarding their location in your cluster.

In order to run your SAS Text Analytics models via DS2 in parallel inside Hadoop, SAS Embedded Process needs to be installed on every node of the Hadoop cluster that is capable of running a MapReduce task or YARN application container. Each node in the cluster must also have access to the full pathname of the scoring models and helper binaries. SAS Embedded Process, along with the SAS Text Analytics components and the binaries, runs your DS2 code directly inside Hadoop and leverages the massive parallel processing and native resources such as MapReduce and YARN for your SAS Text Analytics models.

Please see David Ghazaleh's paper "Exploring SAS® Embedded Process Technologies on Hadoop" for complete information about SAS In-Database Code Accelerator for Hadoop and SAS Embedded Process.

Figure 2 illustrates the workflow. As part of your modeling step, SAS Contextual Analysis generates the DS2 score code and binary models. You extract the DS2 code from SAS Contextual Analysis and modify for your cluster's configuration. The models, categorization (file extension: .mco), concepts (file extension: .li) and sentiment (file extension: .sam) must be copied to a location where the cluster can access them. When the modified DS2 code is executed, a MapReduce job is created and run via YARN. The MapReduce job uses the SAS Text Analytics components in SAS Embedded Process. SAS Embedded Process runs on the data, where the data lives.

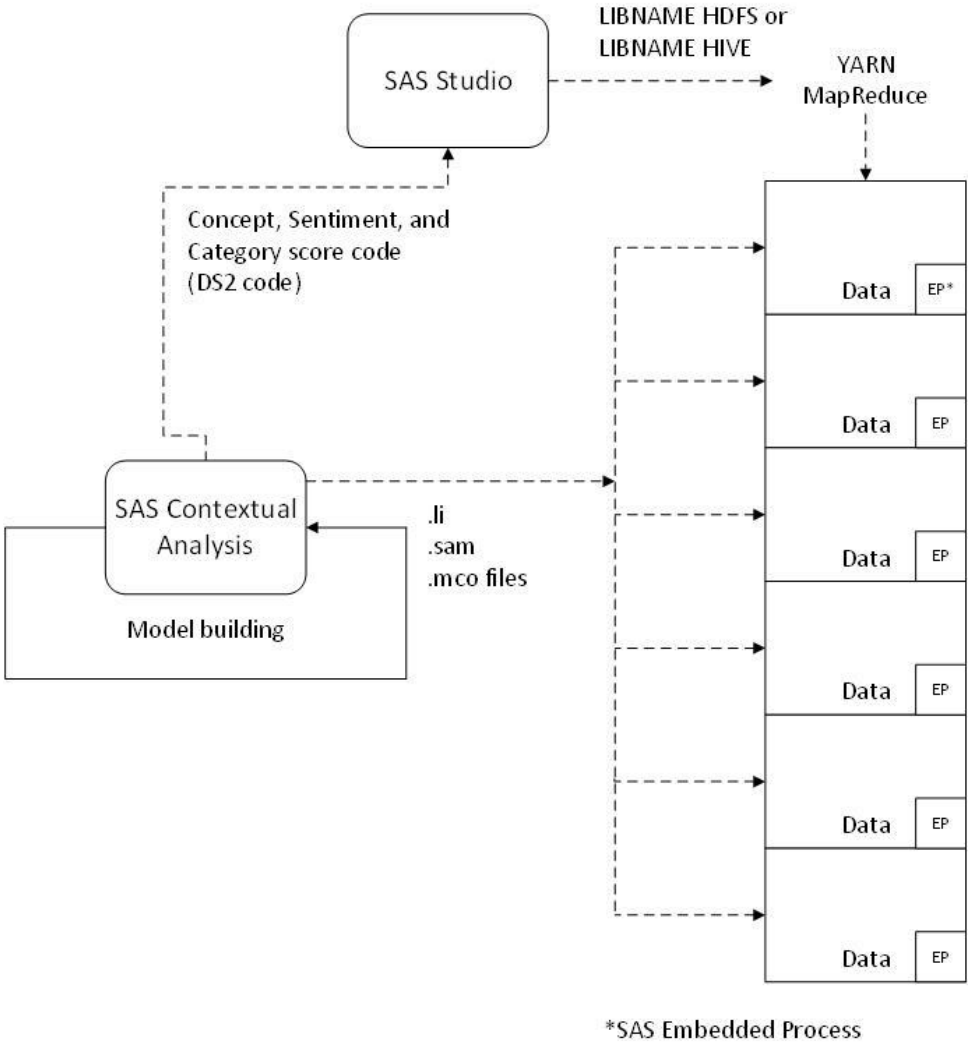


Figure 2. Running Text Analytics Models in Hadoop: Process Flow

SAS DS2 LANGUAGE

DS2 is a procedural programming language with variables and scope, methods, packages, control flow statements, table I/O statements, and parallel programming statements. It is influenced by the SAS DATA step. DS2 programs can run in Base SAS, SAS® High-Performance Analytics, SAS In-Database Code Accelerator for Hadoop, and SAS® In-Memory Analytics. DS2 became a production feature in SAS® 9.4.

DS2 has three system methods, init(), run(), and term(), which are called automatically when a DS2 program executes. INIT runs once on program start, RUN runs once for every row in the input table, and TERM runs once on program termination. The block of code declared with the DATA statement is called a data program.

When a SAS Text Analytics DS2 program runs, similar to any DS2 program, the init() method is called, so you can do any necessary initialization. Then the run() method is called for every row in the input table. Finally, the term() method is called, so you can clean up outstanding resources. The init(), run(), and term() methods do not take any arguments and cannot be explicitly invoked.

DS2 is not intended as a replacement for the SAS DATA step. DS2 is mainly focused on parallel execution. The advantage of using DS2 is that when you hit submit, the MapReduce job starts on the Hadoop cluster. The data is processed in each data node, where the data resides and the output result sets are given to the mapper. Output records are written to an output file on HDFS. Optionally, you can read the output results back into Base SAS.

TEXT ANALYTICS SCORING IN HADOOP

CONFIGURING YOUR ENVIRONMENT

Running your model in database requires you use the ds2accel=yes option with proc ds2.

```
proc ds2 ds2accel=yes;
```

This statement directs SAS to allow the SAS In-Database Code Accelerator for Hadoop to publish a DS2 thread program to the database and execute that program, in parallel, inside the database.

You will also need to specify some environment variables, create a LIBNAME, identify your input data and the text column to score, and specify where you would like SAS Embedded Process to put your output data.

The Hadoop client-side Java code used to access Hadoop services is provided by your Hadoop vendor in the form of JAR files. The SAS Embedded Process install script conveniently creates a ZIP file under the SAS Embedded Process install bin folder that contains the JAR files needed on the client side. Your system administrator will have unzipped the file under a folder of preference. You'll need to know the full path of that folder.

To make the Hadoop JAR files available to SAS, you must set the SAS_HADOOP_JAR_PATH environment variable to the location that contains all of the Hadoop JAR files. The Hadoop JAR files must be accessible by the SAS client machine. The variable is set by the OPTION SET statement. The following is a sample code that sets the SAS_HADOOP_JAR_PATH environment variable:

```
options set=SAS_HADOOP_JAR_PATH="/opt/sasinside/hadoop/lib";
```

Your system administrator has also collected the Hadoop service configuration files. The location of the files depends on the vendor. Common locations for Hadoop and Hive are **`/etc/hadoop/conf`** and **`/etc/hive/conf`**. The basic configuration files you need to collect are: core-site.xml, hdfs-site.xml, mapred-site.xml, hive-site.xml, and if applicable, yarn-site.xml. Your system administrator must copy all Hadoop configuration files from the Hadoop cluster to a physical location on the client machine that is accessible by SAS.

To make the required configuration files available to SAS, you must set the SAS_HADOOP_CONFIG_PATH environment variable to the location of the configuration files. The configuration files are used by the Hadoop client API to find the Hadoop cluster and its services. The variable is set by OPTION SET statement. The following is a sample code that sets the SAS_HADOOP_CONFIG_PATH environment variable:

```
options set=SAS_HADOOP_CONFIG_PATH="/opt/sasinside/hadoop/conf";
```

SAS/ACCESS® Interface to Hadoop provides enterprise data access and integration between SAS and Hadoop. SAS In-Database Code Accelerator for Hadoop works in conjunction with SAS/ACCESS Interface to Hadoop to read and write files on HDFS or tables in Hive. Before running any DS2 program, such as the score code generated by SAS Contextual Analysis, a library reference needs to be assigned to HDFS or the Hive server.

The LIBNAME statement associates a SAS library reference with Hadoop HDFS or Hive. To define a library reference to Hadoop you need specify **HADOOP** as the engine name. The following is an example of how to assign a LIBNAME to Hadoop using Hive. The library reference hive will be associated with a Hive server:

```
libname dblib hadoop server = "server.demo.example.com" user=usr
password=pw;
```

The initialization, main, and termination sections in your DS2 code are the same as the code you extracted from SAS Contextual Analysis with two exceptions. The first is that you must define your thread program.

```
thread workerth / overwrite=YES;
```

The output of the thread program is the input of the data program, which follows the proc DS2 program. Therefore, any values you want kept in the output data set are specified at the end of the proc DS2 code sequence, as shown in the following example:

```
DATA &output_position_ds(
         keep=(
            document_id
            name _tmp_term /* term added */
            start_offset
         )
         overwrite=yes
    );
dcl THREAD workerth THRD;
method run();
set from THRD;
end;
enddata;
run;
quit;
```

COMPLETE PROGRAM FOR CONCEPT SCORING WITH COMMENTS

The API you use in your DS2 code for scoring SAS Text Analytics models is identical to the API used under the hood for SAS Contextual Analysis, SAS Text Miner, SAS Sentiment Analysis, and all SAS text analytics products. This allows for more flexibility and configurability when scoring your data. The API is object oriented and is a direct mapping to the C programming code in SAS Text Analytics.

After you configure your environment, it is time to initialize. In the initialization section you specify the location of your models and supporting binaries by using the new_local_file() method of the txtanio object. For concept modeling you will need the case_mapping.bin and morphological dictionary as supporting binaries, along with your LITI model. You can also specify any options such as match type.

The main section is where you process your data. After specifying the run() method, you set your input data in usual SAS DATA step fashion.

In the term() method you clean up your resources.

The term() method is followed by the data program. The data program uses the first-stage thread program output as its input.

Code for the Concept Model

The following code is a modified version of the source code generated by SAS Contextual Analysis:

```
/* set up your Java Jar and Config paths */
options set=SAS_HADOOP_JAR_PATH="/opt/sasinside/hadoop/lib";
options set=SAS_HADOOP_CONFIG_PATH="/opt/sasinside/hadoop/conf";


/* the library reference gives access to HDFS or Hive */

libname dblib hadoop server = "namenode.demo.xyz.com" user=your.name
password=xyz123;

/* the use of the macro variables is optional */
%let input_ds = dblib.acct_remarks_key_8th;
%let document_column = remark_text;
%let output_position_ds = dblib.acct_remarks_key_8th_out;

/* ds2accel=yes tells Code Accelerator to execute inside Hadoop */

proc ds2 ds2accel=yes;

/* these packages are part of the Text Analytics add-on and are installed
in your EP */
require package tkcat; run;
require package tktxtanio; run;

/* the output of the thread program is the input of the data program which
follows */
THREAD workerth / overwrite=YES;

/* tell DS2 to load your Text Analytics packages and ready them for
processing */
    dcl package tkcat cat();
    dcl package tktxtanio txtanio();

/* these are object handles that are passed between the SAS Embedded
Process C-code and your DS2 program f*/
    dcl binary(8)  apply_settings;
    dcl binary(8)  case_mapping;
    dcl binary(8)  morphologic_dictionary;
    dcl binary(8)  liti_binary;
    dcl binary(8)  trans;
    dcl binary(8)  document;

/* define your variables just like SAS DATA step */
    dcl double status;
    dcl double num_matches;
    dcl double i;
```

```
    dcl double document_id;
    dcl double observation_id;
    dcl char(1024) name;
    dcl char(1024) _tmp_term;
    dcl double start_offset;
    dcl double end_offset;

/* retain statement just like SAS DATA step, these are object handles that
remain constant */
    retain apply_settings;
    retain case_mapping;
    retain morphologic_dictionary;
    retain liti_binary;
    retain trans;
/* the init() method */
method init();

/* the settings object links the binaries with run time options */
    apply_settings = cat.new_apply_settings();

/* load required case_mapping file and bind it to settings object */
    case_mapping = txtanio.new_local_file('/user_space/case_mapping.bin');
    status = cat.set_apply_case_mapping(apply_settings, case_mapping);

/* load the required morphological dictionary and bind it to settings
object */
    morphologic_dictionary = txtanio.new_local_file('en-utf8-
    AnaInfSinFas.mdic');
        status = cat.set_apply_morphologic_dictionary(
                                        apply_settings,
morphologic_dictionary);

/* the required LITI file and bind it to the settings object */
    liti_binary = txtanio.new_local_file('/user_space/en-ne.li');
    status = cat.set_apply_model(apply_settings, liti_binary);

/* 0 means all matches, 1 means longest match, 2 means best match */
        status = cat.set_match_type(apply_settings, 0);

/* here is where you bind the options to the binaries */
    status = cat.initialize_concepts(apply_settings);

/* the transaction object is bound to the document and the settings object
in the run() method */
        trans = cat.new_transaction();
    end;

/* the run() method */
method run();

/* set statement just like DATA step, keep your document identifier and
your text to score */
        set &input_ds(keep=(&document_column,docid));

/* create a document object, bind it to the transaction,
and bind your transaction to the settings object */
        document = txtanio.new_document_from_string(&document_column);
```

```
        status = cat.set_document(trans, document);
        status = cat.apply_concepts(apply_settings, trans);
/* your document id is unique throughout all documents,
your observation id is unique on the data node */
        document_id = docid;
        observation_id = _N_;

/* process the document using the transaction object,
loop through each match and obtain desired variables  */
num_matches = cat.get_number_of_concepts(trans);
        i = 0;
        do while (i LT num_matches);
            _tmp_term = cat.get_concept(trans, i);
            name = cat.get_concept_name(trans, i);
            start_offset = cat.get_concept_start_offset(trans, i);
            end_offset = cat.get_concept_end_offset(trans, i);
            output;
            i = i + 1;
        end;

/* transaction complete, free document, back to the top for the next
observation */
        cat.clean_transaction(trans);
        txtanio.free_object(document);
    end;

/* the term() method cleans up artifacts created in the init() method */
    method term();
        cat.free_transaction(trans);
        txtanio.free_object(liti_binary);
        txtanio.free_object(morphologic_dictionary);
        txtanio.free_object(case_mapping);
        cat.free_apply_settings(apply_settings);
    end;
endthread;
run;

/* Here is where you collect your output data */
DATA &output_position_ds(
         keep=(
            document_id
            observation_id
            name _tmp_term /* term added */
            start_offset
            end_offset
        )
        overwrite=yes
    );
dcl THREAD workerth THRD;
method run();
set from THRD;
end;
enddata;
run;
quit;
```

Similar code is available for the Category and Sentiment models.

PREPARATION FOR SCORING IN HADOOP

Before you can score your data in Hadoop, you must install your models so that they are visible to the Hadoop cluster.

The path to your binaries is available in the SAS Contextual Analysis generated code for each model type. For the concept code, the macro variable, liti_binary_path points to the full pathname of your concept model. If you are running your client on Windows, your path might resemble the following:

```
/* the path to the liti binary... should have been set to your SCA
project's concept binary path */
%let liti_binary_path =
'C:\\Users\\sasdemo\\Documents\\My SAS
Files\\9.4\\demonstration\\config\\concepts.li';
```

The binary files are host independent. Therefore, if they are generated on Windows or another platform other than the flavor of your Hadoop cluster, you can directly copy them.

You can use your favorite Hadoop utility to do this. SAS supplies a simcp Perl script that resides in **`/opt/webmin/util`** on your name node. You can copy the binaries to the appropriate locations with the following utility.

```
cd /opt/webmin/util
./simcp case_mapping.bin /your/user_space
```

See *SAS Contextual Analysis: User's Guide* for more details about the location of your binary models.

OUTPUT

Now you are ready to run your code from SAS Studio or MVA SAS.

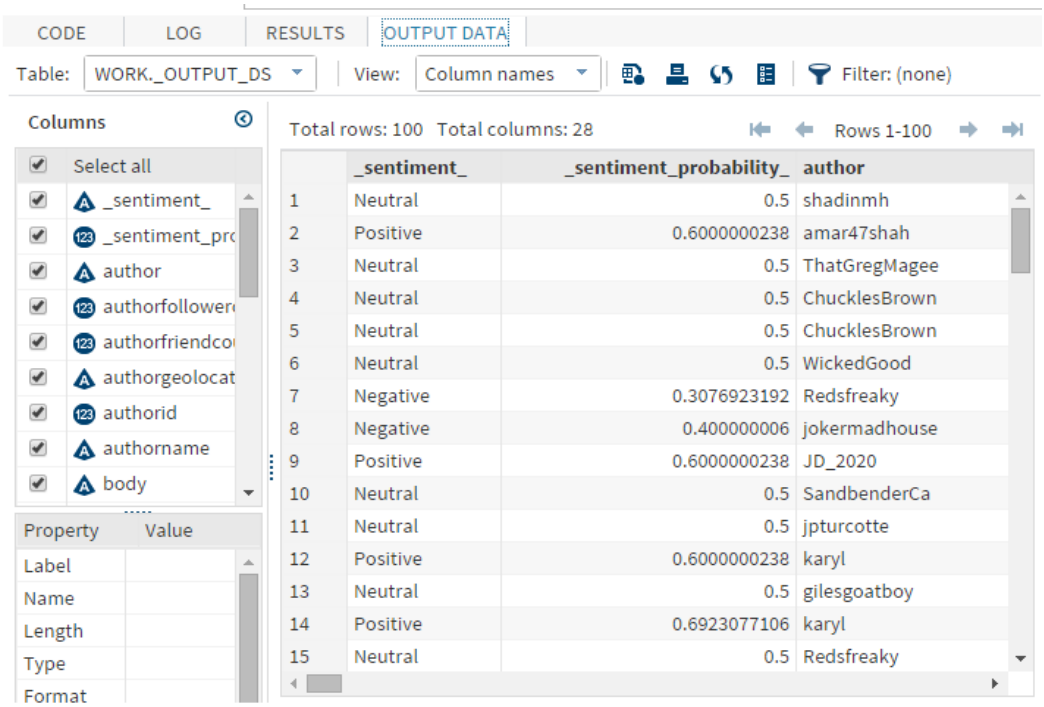
Output for Concept Model

	document_id ▲	name	start_offset	end_offset	term
1	107	NOUN_GROUP	135	154	Male Character trope
2	107	INTERNET	53	100	http://thegeekess.co.za/2014/07/24/pirates-cove/
3	107	TITLE	132	133	Ms
4	107	NOUN_GROUP	140	154	Character trope
5	107	PERSON	4	10	Geekess
6	107	CURRENCY	21	27	2 cents
7	107	TITLE	0	2	Mr.
8	267	LOCATION	1	24	Spacekatgal Raccoon City
9	333	TIME_PERIOD	26	34	last time
10	369	TIME_PERIOD	32	39	30 years
11	377	NOUN_GROUP	100	117	custom environment
12	933	NOUN_GROUP	53	65	great podcast
13	933	DATE	4	9	Friday
14	3185	NOUN_GROUP	73	86	class citizens
15	3185	ORGANIZATION	1	19	Spacekatgal Society
16	8123	INTERNET	69	89	http://bit.ly/19uQTzj
17	8123	PERCENT	61	63	75%
18	8123	COMPANY	16	25	Daily Deal
19	8123	NOUN_GROUP	16	25	Daily Deal
20	8825	INTERNET	56	76	http://bit.ly/1kKjZk2
21	8825	NOUN_GROUP	45	54	first week
22	8825	PERCENT	33	35	20%

Display 2. SAS Studio Output after Scoring with a Concept Model

Output for Sentiment Model

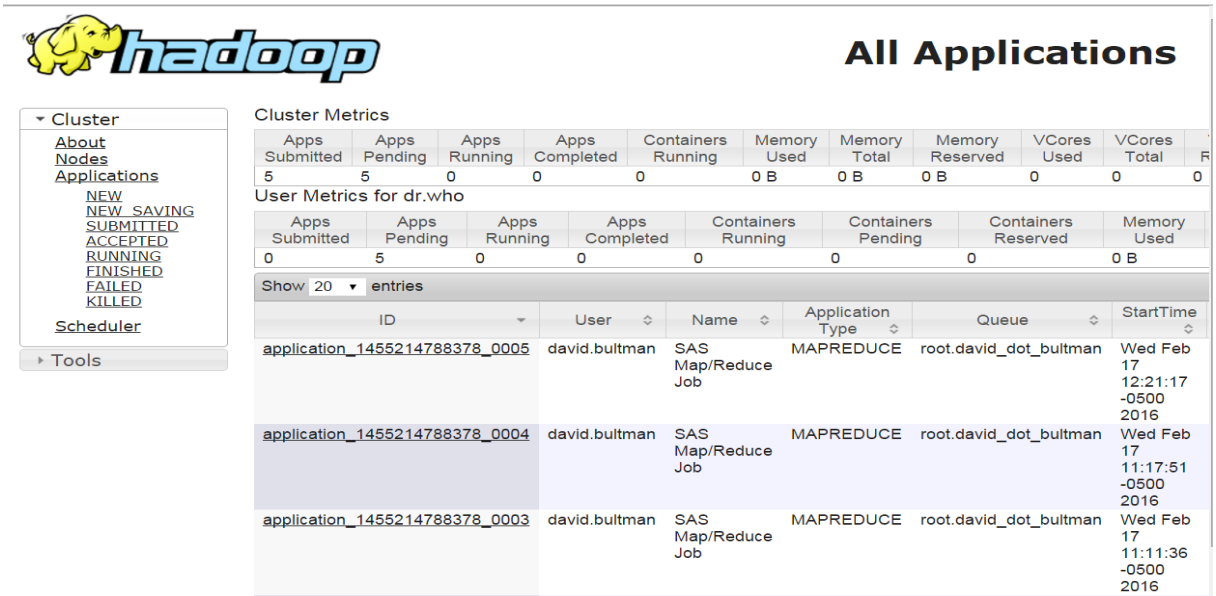
Sentiment models shipped with SAS Contextual Analysis can be run in Hadoop in the same way you would run the concept model. After you modify the DS2 source code to conform to your Hadoop configuration, install your sentiment model on your cluster and run. The output will look similar to Display 3.



	sentiment	_sentiment_probability_	author
1	Neutral	0.5	shadinmh
2	Positive	0.6000000238	amar47shah
3	Neutral	0.5	ThatGregMagee
4	Neutral	0.5	ChucklesBrown
5	Neutral	0.5	ChucklesBrown
6	Neutral	0.5	WickedGood
7	Negative	0.3076923192	Redsfreaky
8	Negative	0.400000006	jokermadhouse
9	Positive	0.6000000238	JD_2020
10	Neutral	0.5	SandbenderCa
11	Neutral	0.5	jpturcotte
12	Positive	0.6000000238	karyl
13	Neutral	0.5	gilesgoatboy
14	Positive	0.6923077106	karyl
15	Neutral	0.5	Redsfreaky

Display 3. SAS Studio Output after Scoring with a Sentiment Model

You can monitor your MapReduce jobs through a web user interface provided by YARN ResourceManager. The address is http://*hostname*:8088. You can monitor the status of the job, the user that submitted the job, the time the job started executing, the duration of the job, and other useful metrics, as shown in Display 4.



Display 4. Monitor MapReduce Jobs

BENEFITS OF SAS CONTEXTUAL ANALYSIS AND HADOOP AS YOUR TEXT ANALYTICS PLATFORM

There are many benefits of using SAS Contextual Analysis and Hadoop as a text analytics platform. Five important benefits apply to various stakeholders across the enterprise:

1. **Minimization of Data Movement (IT)**—Reduce network resource consumption.
2. **Automated Score Code Generation (IT & Data Science)**—There are two instances that require recoding of an algorithm before deployment is possible (1) Language incompatibility—the model is written in SAS or R or Python and needs to be in Java or C+ and (2) Algorithm complexity—the algorithm uses complex mathematics but needs to be deployed via SQL. SAS Contextual Analysis automates this process by using DS2, which is directly translatable in C, saving time and money.
3. **Score Code Portability (IT & Data Science)**—The automatically generated score code can be deployed in Hadoop (taking the model to the data) or deployed in a separate production environment in which SAS Contextual Analysis is not installed.
4. **Cost Effective for Model Development (CFO/CTO)**—Model building is an iterative process. With SAS Contextual Analysis, there are not onerous processing costs like those associated with hosted solutions. Run the code as many times as you like without stunting innovation.
5. **Ease of Use (Business Analysts)**—Writing linguistic rules in SAS Contextual Analysis can be as simple as writing formulas in Microsoft Excel, except much more powerful. This empowers analysts with no Hadoop experience to build models that can be deployed in complex Hadoop environments.

CONCLUSION

DS2 and SAS In-Database Code Accelerator for Hadoop technology provides the SAS user with a powerful framework to deploy SAS Text Analytics in Hadoop using familiar procedural syntax. In this paper, we demonstrate how a SAS user can modify the DS2 score code from SAS Contextual Analysis and leverage MapReduce to drive scoring in a partitioned, highly parallel data environment. We also discuss the benefits of doing so from the perspective of other stakeholders in the organization.

ACKNOWLEDGMENTS

Michael Carman, Sr. Solutions Architect, Cloudera Cluster Administrator, SAS Institute, Inc.

David Ghazaleh, Sr. Software Developer, Code Accelerator, SAS Institute, Inc.

Mike T. Johnson, Development Tester, Hortonworks Cluster Administrator, SAS Institute, Inc.

Tom Weber, Principal Software Developer, Platform R&D, SAS Institute, Inc.

David Zanter, Sr. Software Developer, Code Accelerator, SAS Institute, Inc.

Saratendu Sethi, Sr. Director, Text Analytics R&D, SAS Institute Inc.

Gail Goodling, Sr. Technical Writer, SAS Institute Inc.

RECOMMENDED READING

- Paper SAS5060-2016, “Exploring SAS® Embedded Process Technologies on Hadoop” by David Ghazaleh, SAS Institute, Inc.
- *SAS® Contextual Analysis: Administrator's Guide*
- *SAS® Content Categorization Single User Servers: Administrator's Guide*
- *SAS® Text Miner: Reference Help*
- *SAS® Enterprise Content Categorization Studio: User's Guide*

- *Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS®*
- *SAS® Encoding: Understanding the Details*

CONTACT INFORMATION

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

David Bultman
SAS Institute, Inc.
(919) 531-6875
David.Bultman@sas.com

Adam Pilz
SAS Institute, Inc.
(919) 531-2082
Adam.Pilz@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.