Paper SAS2269-2018

Catch at the Speed of Light: Analytics on Live-Streaming Data Using SAS® Event Stream Processing

Murali Krishna Pagolu, SAS Institute Inc.

ABSTRACT

Two of the most important dimensions of data, variety and volume, have challenged the world of data analytics for quite some time. While volume dictated how data needs to be stored and accessed, variety resulted in innovative techniques to mine non-traditional data types. The third but most important dimension of data, velocity, has significantly changed the paradigm of what constitutes big data. We are indeed living in interesting times from the perspective of analytics as we look to mine data on the verge of its creation. Performing analytics on samples of stored historical data has its due share of importance but that is not enough now. Insights at the speed of light—what is happening right now at this second can be of tremendous value. The timelines of data to insights and insights to actions are shrinking due to rapidly changing dynamics in the way businesses operate now. SAS® Event Stream Processing perfectly caters to this need for capturing and capitalizing streaming data. With the growing use of social media channels, consumers are increasingly sharing their activities, likes and dislikes, opinions, choices, and interests. Live-streaming data can be of significant value if rightly tapped using proper analytics. In this paper, we showcase the combined power of analytics with SAS Event Stream Processing to build a real-time processing engine that can generate insights on the fly on SAS® Viya®.

INTRODUCTION

Today, big data is an indispensable area of interest for businesses, technologists, and scientists alike. But what is big data? What factors constitute and qualify any data as big data? Figure 1 illustrates the three Vs of big data. We cannot term any data as “big” data unless it has volume, variety, and velocity. Organizations are witnessing phenomenal growth in data volumes. Research studies indicate there will be 35 zettabytes of data stored across the globe by 2020, a 50-fold increase since 2010. As volume increases, so does the variety of data stored. Data is available in wide varieties such as text, image, and audio and video files, apart from the traditional structured data. Another study shows that 80% of the data stored is unstructured, which is massive by any measure.

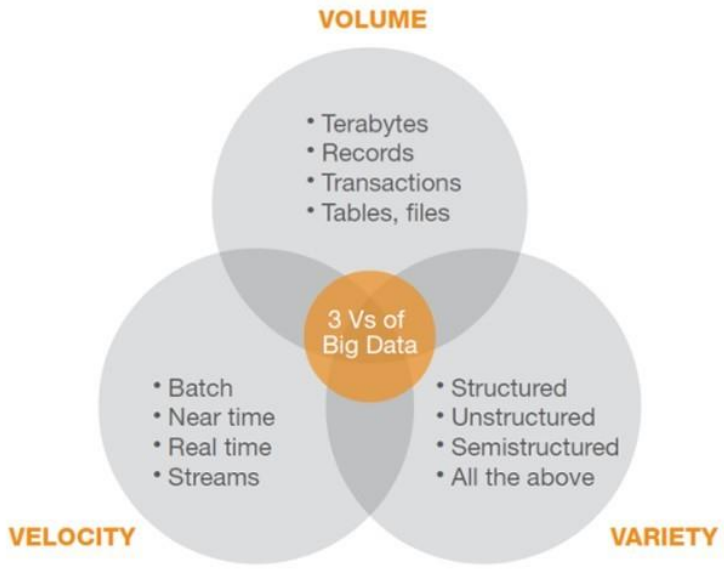

Figure 1 Three Vs of Big Data

The first two Vs, volume and variety, play important roles in the way organizations the world over conduct business operations and manage and study their own data for insights. The third V, velocity, is a particularly hot topic of interest in relation to the idea of connected devices or Internet of Things (IoT). Velocity refers to the speed at which data emanates from the source and is ported over to the destination systems. Figure 2 illustrates examples of applications for Internet of Things in some industries.

Sources are typically devices such as sensors, smart phones, instruments that are reading measurements such as energy consumption, monitors, or systems connected to several components of machines such as cars. These sources transmit the data recorded at a specific point in time, sometimes as frequently as every millisecond or microsecond. This level of detailed information can accumulate very quickly and reach sizes in the range of several gigabytes or terabytes within no time. Because of this, it is sometimes difficult to store data with high velocities into a permanent data sink. With this challenge comes the opportunity to leverage the technology that can analyze, summarize, and infer insights from the data as it transits, without even having to permanently store it. This field of study is particularly known in the industry as event stream processing, and SAS Event Stream Processing is the product offering meant to fill this need.

Figure 2 Examples of Applications for Internet of Things in Some Industries

In the utilities sector, a simple example of implementing internet of things is to have sensors from all the office/meeting rooms, hallways and open areas in your office building generate readings such as temperature, humidity, air pressure etc. in real time along with information captured from various components of Air Handling Units. This helps you to stream all the connected data into a common stream whereby you can perform analytics to understand demand vs supply, factors causing or associated with areas in the building which aren't necessarily performing optimally at a specific point in time. In a connected car transportation scenario, information generated by sensors connected to car's devices and instruments in real time which transmit them back to the server in a data stream. All the information from various drivers across the fleet can be analyzed for anomalies and corresponding alerts can be sent back to the drivers about exceeding speed limits, dangerous weather conditions, possible urgent maintenance etc. all done in real time. This type of applications can help monitor the health of vehicles as well as driver's driving patterns to save cost by preventing accidents or unwarranted downtimes for vehicles.

SAS EVENT STREAM PROCESSING – OVERVIEW

SAS Event Stream Processing enables you to tap into streaming data generated from various sources, and synthesizes the information to make it useful for understanding the underlying patterns or insights. It is particularly useful for deployments where a large volume of data flows in short spans of time, which requires low-latency and faster processing for results. Figure 3 provides a high-level overview of the model hierarchy for SAS Event Stream Processing. At the highest level is an event stream processing engine. Every model is associated with only one instance of the engine. An engine can contain one or more projects that run in their dedicated thread pool. Each project can contain one or more continuous queries. A continuous query is somewhat like a flow chart that indicates the direction of data flow. The only restriction in a continuous query is that there should be no parallel or closed loop flows. Each node in a continuous query is called a window, and there should be a source window in every continuous query. All windows are connected by directional arrows called edges. All windows other than the source window are referred as derived windows. The purpose of derived windows is to perform various types of data transformations and analysis. All layers in an SAS Event Stream Processing project are in XML format in the background. Therefore, it is easy to export and import SAS Event Stream Processing projects from and to SAS Event Stream Processing Studio, which is the visual interface for managing projects.

A data stream is referred to as a publish event stream when it is in bound into the source window of the event stream processing model. When a data stream from any window (either source or derived) of the event stream processing model is outbound toward a target application or data sink, it is referred as a subscribe event stream. The term event is specifically used for streaming data that is comparable to either a row or observation in a stored data table. An event block is a set of events. Each event has a set of fields that are akin to columns for an observation in a stored data table. Connectors can either publish or subscribe event streams, and they run synchronously within the event stream processing engine. Adapters can also either publish or subscribe event streams, but they are usually asynchronous processes that are potentially running independently of the event stream processing engine. In the following sections, you will see examples where we first initiate an event stream processing engine, and then inject events into a source window using an adapter. In cases such as this example, the continuous queries keep running, looking for a handshake with an adapter on order to make a connection and inject events. The publish/subscribe API can facilitate subscribing to an event stream window or publishing event streams into an event stream processing model source window.

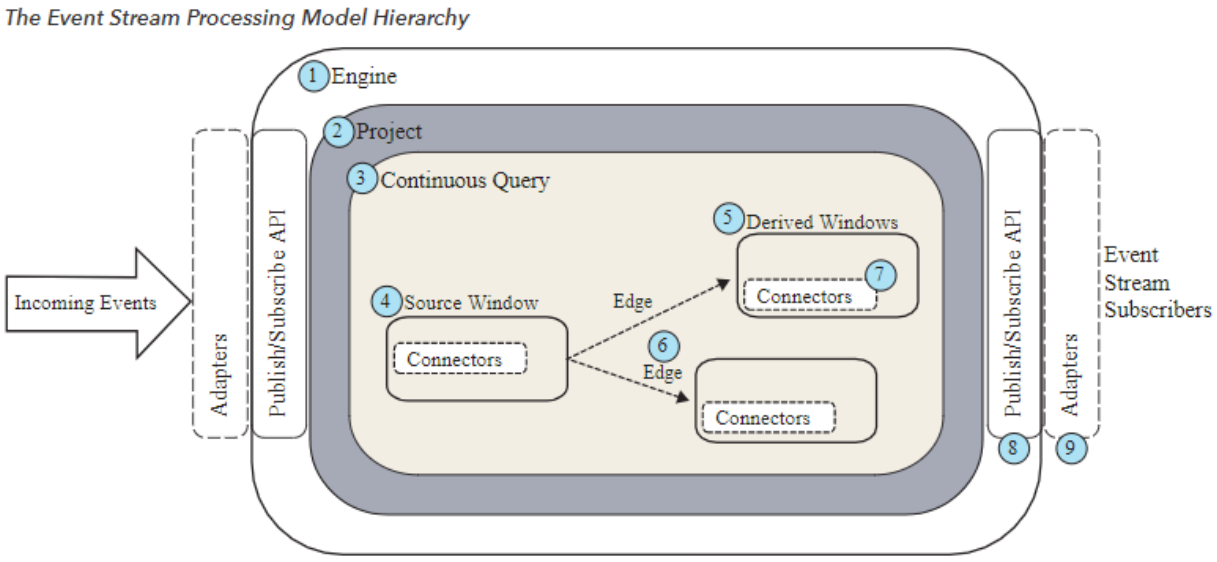

Figure 3 SAS Event Stream Processing – Overview

In SAS Event Stream Processing, you can perform Analytics in two different ways on the streaming data. Analytics that are performed on streaming data using models developed offline (outside of the event

stream processing engine) are called streaming analytics offline. If an analytical model is developed at the time when the data is streamed and is constantly updated as more data is available then it is called streaming analytics online. SAS Event Stream Processing 5.1 comes with several analytical modeling algorithms for use cases where both online and offline streaming analytics is relevant. The complete list of algorithms available for performing analytics is listed below in Table 1.

Mode	**Algorithm**	**Window(s)**	**Required File**
Online	Streaming K-Means Clustering Streaming DBSCAN Clustering Streaming Linear Regression Streaming Logistic Regression Streaming Support Vector Machines	Train and Score	-NA-
Online	Streaming Summary (Univariate Statistics) Streaming Pearson's Correlation Segmented Correlation Streaming Distribution Fitting Short-Time Fourier Transform Compute Fit Statistics Compute ROC Information Calculate a Streaming Histogram Streaming Text Tokenization Streaming Text Vectorization Image Processing Algorithm Moving Relative Range	Calculate	-NA-
Offline	Support Vector Data Description Random Forest Gradient Boosting Tree Support Vector Machine Factorization Machine Robust Principal Components Analysis Deep Neural Networks Convoluted Neural Networks Bayesian Network	Model Reader	.ASTORE
Offline (Text Analytics)	Content Categorization	Text Category	.mco
	Sentiment Analysis	Text Sentiment	.sam
	Entity Extraction	Text Context	.li
	Topic Extraction	Text Topic	.ASTORE

Table 1 List of online and offline algorithms available in SAS Event Stream Processing

Train window gives you the ability to use online training through any of the available algorithms and at the same time leveraging the developed model to score in real time using the score window. You can choose to perform the scoring either on the same data stream passing through the train window or another data stream with similar fields. Calculate window usually generates an output based on the input fields coming through the data stream and the appropriate algorithm you choose for performing analytics. A model reader window helps you to read models developed offline (outside SAS Event Stream Processing) using a binary file format proprietary to SAS called ASTORE which stands for Analytic Store. Additionally, there are text analytics windows specific to content categorization, entity extraction, topic extraction and sentiment analysis with dependency on relevant binary files. In the next sections, will discuss two examples which demonstrate the power of using offline or online analytics while streaming data through SAS Event Stream Processing.

EXAMPLE 1: STREAMING DATA ANALYTICS (TEXT ANALYTICS)

Twitter is a prime example of an active social media platform where conversations and updates tend to bring a lot attention to the public as well as to corporations. Events such as product or service launches, presidential elections, and major economic or political news have proven to make an impact on this microblogging service platform. Tapping such real-time data and analyzing it on the fly can have its own benefits. Companies can quickly understand the market pulse and public sentiment around product launches real time. Usually, field representatives collect, analyze, and provide consolidated feedback through traditional reporting and customer feedback channels, but this feedback generally takes several months from the time of launch. SAS Event Stream Processing comes with a Twitter Publisher Adapter that you can configure to pull sampled tweets from Twitter based on several parameters.

SAS EVENT STREAM PROCESSING ENGINE

As a starting point, you need to start an instance of the ESP server, specifying the http and pubsub ports as shown in the command below. In this command, -http represents the port for http REST API calls and -pubsub indicates the publish/subscribe port.

```
$DFESP_HOME/bin/dfesp_xml_server -http 5556 -pubsub 5555 <<-model url>>
```

You can use the -model parameter if you want to run the ESP server on a specific event stream processing model stored as an XML file. The url denotes the full path of the model file (file://path). $DFESP_HOME is the environment variable that points to the installation directory for SAS Event Stream Processing. After the ESP server is successfully started, you should see a message like this in the console.

```
2018-03-01T14:41:59,744; INFO ; 00000030; DF.ESP; (dfESPengine.cpp:827);
[Engine0007] dfESPengine::initialize() dfESPengine version 5.1 completed
initialization
2018-03-01T14:41:59,750; INFO ; 00000031; DF.ESP; (Esp.cpp:675);
[XMLServer0001] esp engine started, version 5.1, pubsub: 5555
2018-03-01T14:41:59,750; INFO ; 00000032; DF.ESP; (Esp.cpp:859);
[XMLServer0001] starting esp server
2018-03-01T14:41:59,750; INFO ; 00000033; DF.ESP; (Http.cpp:560);
[XMLServer0001] starting HTTP server on port 5556, type=admin
```

SAS EVENT STREAM PROCESSING STUDIO

SAS Event Stream Processing Studio, shown in Figure 4, provides the convenience of a visual interface for users who are comfortable with point and click actions for creating projects, continuous queries, and workflows. As previously mentioned, windows, continuous queries, and projects are all in XML file format under-the-hood. Therefore, you have the luxury of either using the drag and drop interface to set up your workflows with the relevant windows or directly importing the XML model file. After you gain a good understanding of the embedded XML structure of the layers, it is easy to programmatically create the XML files rather than relying on the user interface. In this example, we are interested in analyzing the Twitter content that is published from the Twitter public API and injected into the SAS Event Stream

Processing model. “sourceTwitter” is the source window that is the inlet for the streaming data to come in from the Twitter public API. This source window must have a definite schema (which is a set of fields and their properties) as specified in the topic “Using the Twitter Publisher Adapter” in *SAS Event Stream Processing 5.1*. After the Twitter publisher adapter pushes the streaming data, it moves through the windows in the flow. We calculate averages and counts on certain key measures either as directly coming as fields from the Twitter public API or derived using the analytical models that we use within the flow. In this example, we are using three types of text analytic models for analyzing the Twitter content (tweets) in real time and generating the summarized output (count or average). These three windows are described in detail below:

- **tweetSentiment**: This is a Text Sentiment text analytics window that is used to identify the sentiment mentioned in the analyzed content and classify it to either “positive”, “negative” or “neutral” sentiment. It requires a .sam binary file, which is a model file compiled from a sentiment analysis project developed using SAS® Contextual Analysis.
- **ContentCategorization**: This is a Text Category text analytics window that is used to classify the content into one or more categories based on the context and linguistic rules that are built in a taxonomy structure. It helps to summarize the entire streaming content into buckets for efficient reporting and consolidation. It requires a .mco binary file, which is a model file compiled from a content categorization project developed using SAS Contextual Analysis or SAS® Visual Text Analytics.
- **entityExtraction**: This is a Text Context window that is used to extract contextually relevant matches for entities or facts as they are available in the content. Examples of entities or facts are names of people, titles, locations, events, organizations, and measures. It requires a .li binary file, which is a model compiled from a contextual extraction project developed using SAS Contextual Analysis or SAS Visual Text Analytics.

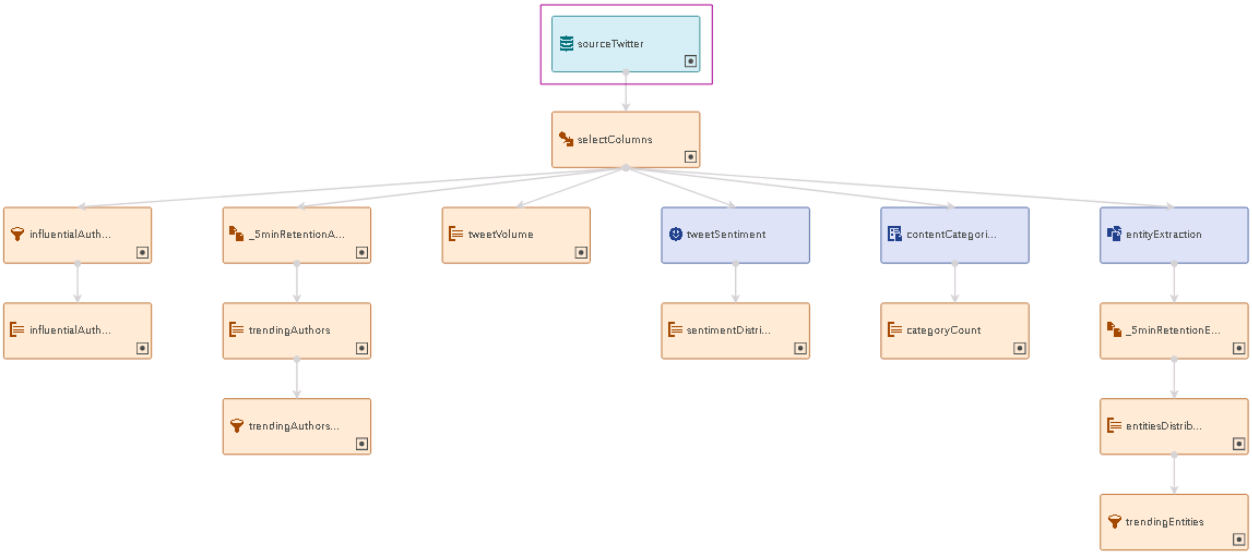

Figure 4 SAS Event Stream Processing Studio - Project Interface

These windows also have a role in this analysis.

- **influentialAuthorsFilter**: This is a Filter transformations window that is used to filter out content based on a logical or numerical expression. In this case, we filter out tweets where the author of the tweet has fewer than 10K followers on Twitter.
- **influentialAuthors**: This is an Aggregate transformations window that is used to aggregate the filtered content from the previous window. In this case, we added up the number of tweets generated by influential authors based on their number of followers on Twitter.
- **_5minRetentionAuthors**: This is a Copy transformations window that is typically used for subsetting the number of fields coming from the previous window. In this case, we also set the event retention to five minutes. This specifies that the events flowing through the previous window are retained in memory for a period of five minutes, which helps to develop aggregations in the following window using the retained block of events.
- **trendingAuthors**: This is an Aggregate transformation window that is used for aggregating the counts of authors who tweeted during the previous five-minute retention period.
- **trendingAuthorsFilter**: This is a Filter transformation window that is used for filtering out the aggregated information based on whether each author tweeted more than once in the last five minutes. The output that is generated shows authors who tweeted at least twice in the last five minutes.
- **tweetVolume**: This is an Aggregate transformation window that is used for aggregating the total volume of tweets within a time interval indicated by the tweet datetime stamp.
- **sentimentDistribution**: This is an Aggregate transformation window that is used for aggregating the total volume of tweets that are classified into positive, negative, or neutral categories.
- **categoryCount**: This is an Aggregate transformation window that is used for summarizing tweet categories using the categorization weighting score generated from the binary categorization model.
- **_5minRetentionEntities**: This is a Copy transformations window that is typically used for sub-setting the number of fields that are coming from the previous window. In this case, we also set the event retention to five minutes. This specifies that the events flowing through the previous window are retained in memory for a period of five minutes, which helps to develop aggregations in the following window using the retained block of events.
- **entitiesDistribution**: This is an Aggregate transformation window that is used for aggregating the counts of extracted entities from the tweets that were retained in the previous five minutes.
- **trendingEntities**: This is a Filter transformation window that is used to filter out the aggregated counts of extracted entities in the previous window. Only the entities or terms that occur more than once in the previous last five minutes period are retained. The others are omitted.

TWITTER PUBLIC API – ACCESS SETUP

The Twitter adapter is provided with SAS Event Stream Processing, but you must perform additional steps to run the adapter with the event stream processing engine. First, you must download the two twitter4j JAR files **twitter4j-core-4.0.4.jar** and **twitter4j-stream-4.0.4.jar** from http://twitter4j.org/en/index.html and copy the files to the lib directory located under the SAS Event Stream Processing installation directory. You must also ensure that the DFESP_TWITTER_JARS environment variable is set to this value.

```
/opt/sas/viya/home/SASEventStreamProcessingEngine/4.2.0/lib/twitter4j-core-
4.0.4.jar:/opt/sas/viya/home/SASEventStreamProcessingEngine/4.2.0/lib/twitter
4j-stream-4.0.4.jar
```

Next, you must ensure that the Twitter adapter configuration file twitterpublisher.properties, which is in the /etc directory under the SAS Event Stream Processing installation directory, is up-to-date with the

required consumerkey (API Key), consumersecret (API Secret), Access Token, and Access Token Secret. These keys should not be generated using your individual Twitter handle, but through the official SAS Event Stream Processing registered Twitter App ID. Refer to the topic “Using the Twitter Publisher Adapter” in *SAS Event Stream Processing 5.1* to find the steps for registering a Twitter account for the SAS Event Stream Processing Twitter adapter and generating access tokens.

SAS EVENT STREAM PROCESSING – TWITTER PUBLISHER ADAPTER

After the required access tokens (twitterpublisher.properties) and necessary JAR files are in place, you can start the SAS Event Stream Processing Twitter publisher adapter. It injects events (tweets) that are pulled from the sampled Twitter public API into the event stream processing model we have set up in the SAS Event Stream Processing Studio. Run this command to make the connection and let the data flow through the event stream processing model.

```
$DFESP_HOME/bin/dfesp_twitter_publisher -u "dfESP://<<Server
Name>>:5555/TwitterTextAnalyticsDemo/cqTwitter/sourceTwitter" -m filter -k
"iphone" -a "en" -b 20
```

- **<<Server Name>>**: Replace this with the fully qualified name of the server where SAS Event Stream Processing is installed.
- **TwitterTextAnalyticsDemo** is the name of the event stream processing project.
- **CqTwitter** is the continuous query that is in the “TwitterTextAnalyticsDemo” event stream processing project.
- **sourceTwitter** is the source window for the continuous query “CqTwitter”.
- **-m filter** indicates the method of accessing the Twitter publisher API. The filter will take a random sample of the full public API stream.
- **-k “iphone”** is the parameter passed to further filter the sampled tweets based on a keyword search in the tweets.
- **-a “en”** is the parameter to specify the language (en stands for English).
- **-b 20** is the parameter for setting the number of events (tweets) to be pulled in an event block (set to 20 here).

After the command successfully runs, you should see the connection established, with it receiving a status stream and other relevant messages indicating a successful flow of tweets from the public API into SAS Event Stream Processing. One of the advantages of SAS Event Stream Processing Studio is that you can test how various windows work and how they provide the output as the data is streaming live from the Twitter public API into the SAS Event Stream Processing model. Figure 5 shows data streaming in real time through the specific window we chose to view in the test process. Similarly, you can switch to any other selected window for testing the output and see the data streaming in real time.

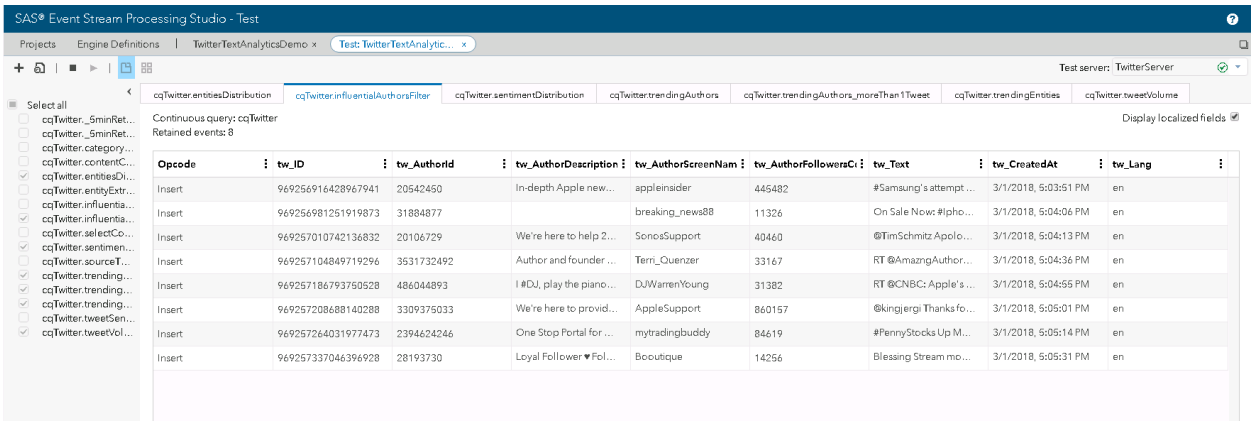

Figure 5 SAS Event Stream Processing Studio - Test Interface

SAS EVENT STREAM PROCESSING STREAMVIEWER

What is the fun in doing all this but having nothing to visualize? SAS Event Stream Processing Streamviewer just does that, providing the ability to subscribe to any of the windows in the event stream processing model and the capability to provide visualizations on the subscribed stream of data in either updating or streaming mode. Ideally, you would want to leverage the aggregate and filter windows that we specifically created to monitor trends on tweet volumes, authors, and so on. Figure 6 is a dashboard that is constantly updated as the data stream comes through from the Twitter public API, is injected into our XML model, and is eventually pushed to the SAS Event Stream Processing Streamviewer target application in subscription mode.

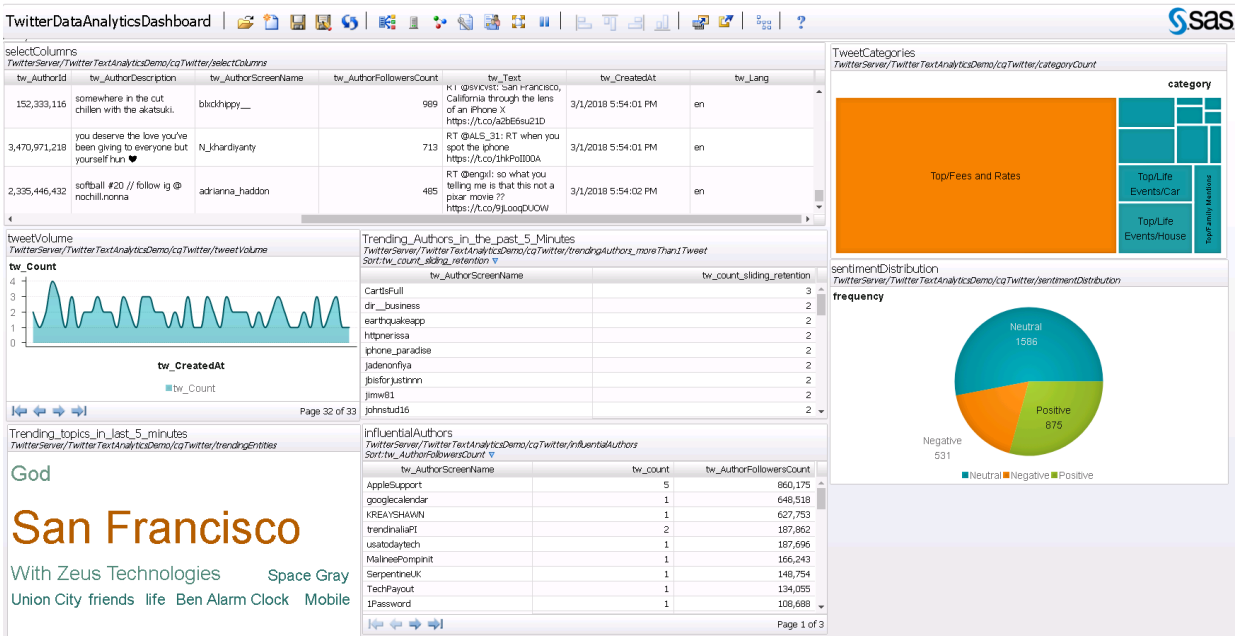

Figure 6 SAS Event Stream Processing Streamviewer

EXAMPLE 2: STREAMING DATA ANALYTICS (K-MEANS CLUSTERING)

The ability to build models while the data is at rest is the traditional means of model development. However, SAS Event Stream Processing enables you to build models as the data is streaming. This unique and powerful feature in SAS Event Stream Processing caters to the current market needs, because some applications such as stock market price feeds, credit card fraud detection, cyber security, and system health monitoring might require you to build or update models as data accumulates without

much time delay between model refinement and deployment. SAS Event Stream Processing supports K-means clustering, DBSCAN clustering, linear regression, logistic regression, and support vector machines with training and scoring online as data streams. Figure 7 shows a simple model that is configured to simultaneously train K-means clustering on the data streaming through the “w_source” window and score the data.

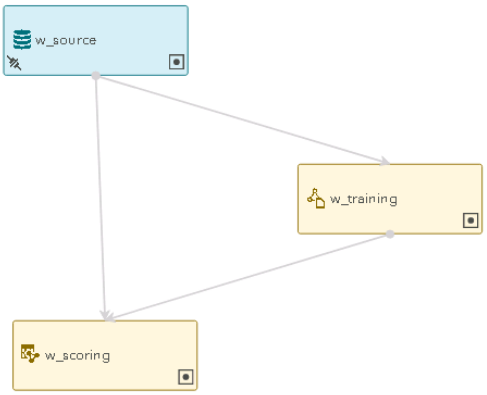

Figure 7 Online Streaming Analytics – k-means Clustering Project Diagram

The K-means clustering algorithm is available as a streaming analytics online algorithm for SAS Event Stream Processing. As the source window receives the incoming stream of data, it is passed on to both the training and scoring windows. The training window produces a k-means model, and keeps updating it as more observations stream through the source window. In this example, the source data contains only three fields: id, x_c, and y_c, which stand for identification, x-coordinate and y-coordinate respectively. Figure 8 shows the output generated from the training window, displaying the number of models iteratively developed by the k-means algorithm as the data stream passed through the training window. The field **model_id** identifies each individual model. You can see a total of 18 models were built by the k-means algorithm. The field **model_addr** identifies the location where the individual model is stored and referenced.

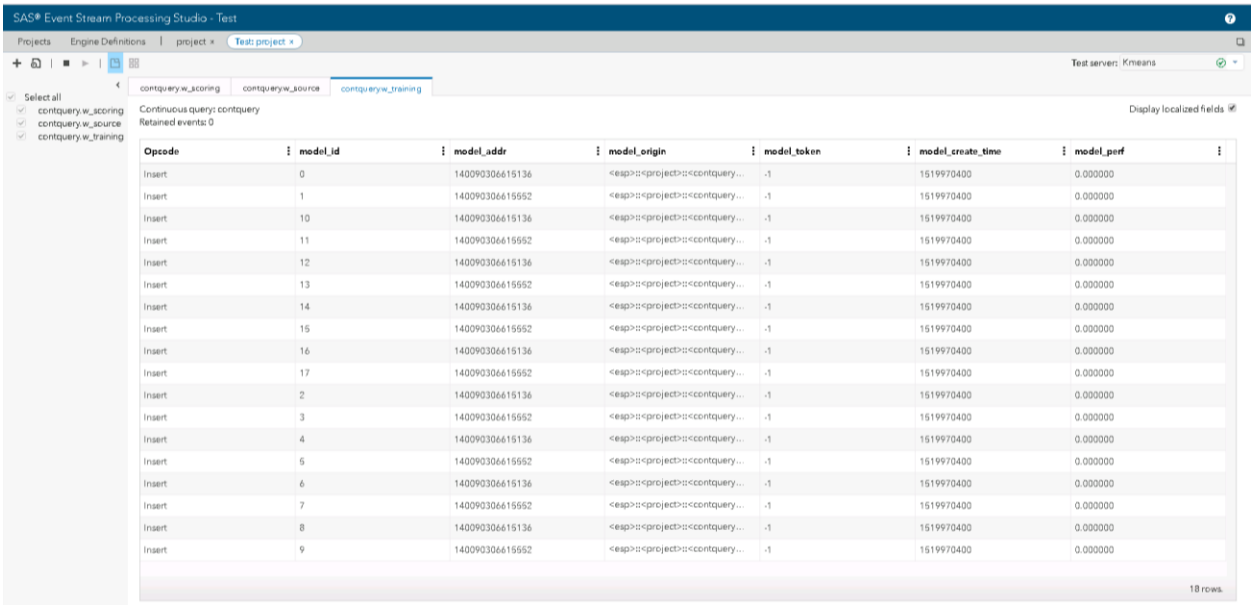

Figure 8 Online Streaming Analytics – k-means Clustering (Training)

Figure 9 shows the output generated from the scoring window. We find the original three fields, id, x_c, and y_c, along with the scoring output based on the model available from the training window at the time of the event passing through the scoring window. The field **seg** corresponds to cluster membership. In this case, there are only two clusters, numbered 0 and 1. The field **min_dist** specifies the distance to the nearest cluster for that event. The field **model_id** is the individual model used for scoring the event in the scoring window. You can also cross-reference this field with the **model_id** field in the output from the training window.

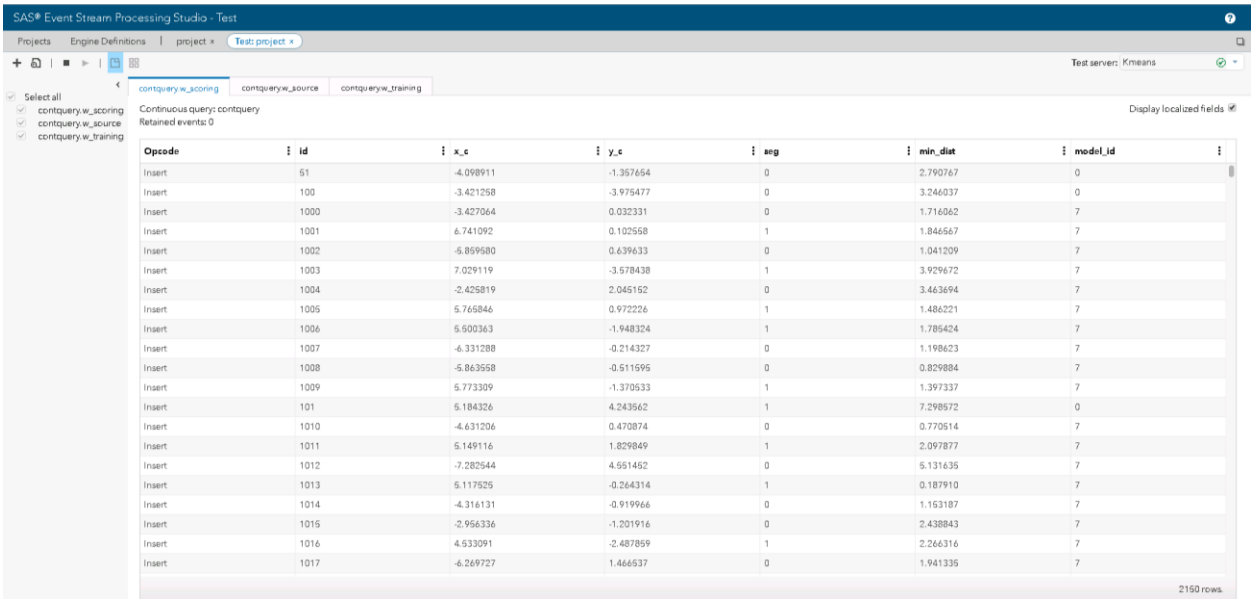

Figure 9 Online Streaming Analytics - k-means Clustering (Scoring)

CONCLUSION

SAS Event Stream Processing is an excellent fit for solving business problems that require you to perform analytics on the fly and to generate insights as data streams even without storing the data. SAS Event Stream Processing provides an abundant variety of advanced analytical modeling algorithms, both supervised and unsupervised, that are capable of processing structured data and unstructured data (such as text and images) along with the ability to use models developed both offline and online.

REFERENCES

SAS Institute Inc. 2017. “Using the Twitter Publisher Adapter.” In *SAS Event Stream Processing 5.1*. Cary, NC: SAS Institute Inc. Available: http://go.documentation.sas.com/?cdcId=espcdc&cdcVersion=5.1&docsetId=espca&docsetTarget=p1wqds40k1qcggn1l1mmkhfbpp44.htm&locale=en (accessed March 2, 2018)

Walker, Michael. “Data Veracity.” Available https://www.datasciencecentral.com/profiles/blogs/data-veracity. Accessed on March 2, 2018.

ACKNOWLEDGMENTS

I would like to thank Nicolas Robert, Steve Sparano, Steven Enck, Xiangqian Hu, Shunping Huang, Scott Pope, and Evan Guarnaccia at SAS for their guidance and support, which helped me develop the content for this paper.

RECOMMENDED READING

SAS Institute Inc. 2017. *SAS® Event Stream Processing 5.1: Users Guide*. Cary NC: SAS Institute Inc. Available:

http://go.documentation.sas.com/?cdcId=espcdc&cdcVersion=5.1&docsetId=espov&docsetTarget=home.htm&locale=en

CONTACT INFORMATION

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

Murali Pagolu
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
murali.pagolu@sas.com
http://www.sas.com