WILL THEY BLEND

The Blog Post Collection

KNIME®

Rosaria Silipo (Ed.)

This book has been updated for **KNIME 3.5**.

For information regarding permissions and sales, write to:

KNIME Press
Technoparkstr. 1
8005 Zurich
Switzerland

knimepress@knime.com

Table of Contents

Introduction

Data scientist has been named as the sexiest job of the 21st century, according to an article on Harvard Business Review (https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century).

One evening in September 2016, when some colleagues and I were sitting around, amiably cleaning up data – as one does as a data scientist - we started a discussion about what the sexiest job of the 21st century actually entails. In addition to machine learning and parallel computing, it also involves data cleaning, data integration, data preparation, and other more obscure tasks in the data science discipline.

Data integration or, as it is called nowadays, data blending is a key process to enrich the dataset and augment its dimensionality. And since more data often means better models, it is easy to see how data blending has become such an important part of data science. Data blending is an integration process and, like all integration processes, its problem is the diversity of the players involved.

So bring on the players! In a first experiment, we need to get data from a database. The next experiment might require the same data but from another database, which means speaking a slightly different SQL dialect. Including some data from a REST service API could be useful too and when we're talking REST APIs, we need to be able to parse and integrate XML and JSON formatted data structures. Of course, we can't leave out the omnipresent Excel file. But what about a simple text file? It should be possible to integrate the data whatever its original format. The most advanced among us might also need to connect to a big data platform which takes us to the question of deciding which one - as they all rely on a slightly different version of Hive and/or Spark.

Integrating data of different types is another thing we data scientists have to take care of: structured and unstructured data, data from a CRM database with texts from a customer experience tool, or perhaps documents from a Kindle with images from a public repository. In these cases, more than with data blending we are dealing with type blending. And mixing data of different types, like images and text frequency measures, can be non-intuitive.

Time blending is another issue. For those of us who are somewhat vestige of an older analytics era, we often have to blend current data with older data from legacy systems. Migrations are costly and resource intensive. So legacy tools and legacy data easily ignore hypes and modern technology.

Lab leaders might dream of having a single data analytics tool for the whole lab, but this is rarely a reality. Which quickly takes us from data blending to tool blending. Legacy tools need to interact with a few contemporary tools either operating in the same sectors or in slightly different sectors. Tool blending is the new frontier of data blending.

After so much discussion, my colleagues and I came to the conclusion that a series of blog posts to share experiences on the blending topic would help many data scientists who are running a real-life instance of the sexiest job of the 21st century.

Digging up blending memories on YouTube (https://youtu.be/lBUJcD6Ws6s), we decided to experiment the craziest blending tasks in a "Will they blend?" blog post series. All posts from the series have been collected in this book to pass data blending know-how to the next generation of data scientists.

I hope you will enjoy the stories as much as we did.

1. IBM Watson meets Google API

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/01_Data_Access/05_REST_Web_Services/02_IBM_Watson_News–Google_News
Posted on: November 7, 2016

The Challenge

It's said that you should always get your news from several, different sources, then compare and contrast to form your own independent opinion. At the point of this writing we are only days away from the US election and all the news we could find are articles about the election race between Hillary Clinton and Donald Trump. Our question then is: What has Obama been doing?

So what about blending IBM Watson's News Service with Google News to find out?

My free subscription for IBM Watson is running out in a few days, so I'd better hurry up to experiment how I can use it inside KNIME Analytics Platform. Google News on the other side is a free limited service.

Let's see what happens when we blend IBM Watson News and Google News within KNIME Analytics Platform. Shall we?

Topic. Barack Obama in the news.

Challenge. Extract and combine news headlines from Google News and IBM Watson News.

Access Mode. REST service for both Google API and IBM Watson.

The Experiment

Querying IBM Watson News Service

1. Create an account with IBM Watson Console at https://console.ng.bluemix.net (free for 30 days) and enable it for the Alchemy category, since the News service is part of the Alchemy category.

2. Identify the API Key that was produced with your registration.

3. Define the template REST request for IBM Watson. In the case of the News service, it would look something like this:
https://access.alchemyapi.com/calls/data/GetNews?apikey=**<API_Key>**&return=enriched.url.title&start=1477008000&end=now&q.enriched.url.cleanedTitle=**InsertTitle**&count=100&outputMode=xml

   Where:
   Access.alchemyapi.com/calls/data/GetNews is the REST service
   <API_Key> is the API key you got at registration
   Return specifies which part of the news we get back, in this case just the title
   Start and *end* indicate respectively start and end of time window (in UTC seconds) for news search
   q.enriched.url.cleanedTitle is the topic string (InsertTitle is just a placeholder. It would be Barack+Obama in our case)
   Count is the maximum number of returned results
   outputMode is the response format: XML or JSON

4. Notice that <API_Key> and InsertTitle are just placeholders for the real values. <API_Key> would be the API Key you got at registration and InsertTitle would be the search topic, in our case Barack+Obama.

5. We used Quickform String Input nodes to define two flow variables: the search topic and the API Key.

6. The template REST request was written into a Table Creator node and passed to a String Manipulation node to override the search topic and API Key with the corresponding flow variable values.

7. The request is then sent to IBM Watson News through the GET Request node.

8. The response is received back in XML format as selected in the query and parsed with an XPath node.

Querying Google API News Service

Now we'll send a similar request to the Google News API service.

1. Create an account with Google Console at https://console.developers.google.com and enable it for the Custom Search category. There used to be a Google News API but this seems not to be available anymore. So we will query the Custom Search service for a specific topic on the Google News page.

2. Identify the API Key that was produced with your registration.

3. Create a custom search engine at https://cse.google.com to be used in the Custom Search query and remember its engine ID or its cx parameter.

4. Define the template REST request for Google News. In the case of the custom search service on the Google News page, it would look something like this:
   https://www.googleapis.com/customsearch/v1?q=InsertTitle&cref=News+google&cx=<Your-cx-id.number>&key={YOUR_API_KEY}

   Where:

   googleapis.com/customsearch/v1 is the REST service
   key is the API key you got at registration
   q is the topic string (InsertTitle is just a placeholder. It would be Barack+Obama in our case)
   cref is the name of the custom search engine we created
   cx is the code for the custom search engine we created

   Notice that {YOUR_API_KEY} and InsertTitle are just placeholders for the real values. {YOUR_API_KEY} would be the API Key value you got and InsertTitle would be the search topic, in our case Barack+Obama.

   Notice also that you need to insert your own cx value in the template REST request.

5. We used Quickform String Input nodes to define two flow variables: the search topic and the API Key.

6. The template REST request was written into a Table Creator node and passed to a String Manipulation node to override the topic and the API Key with the corresponding flow variable values

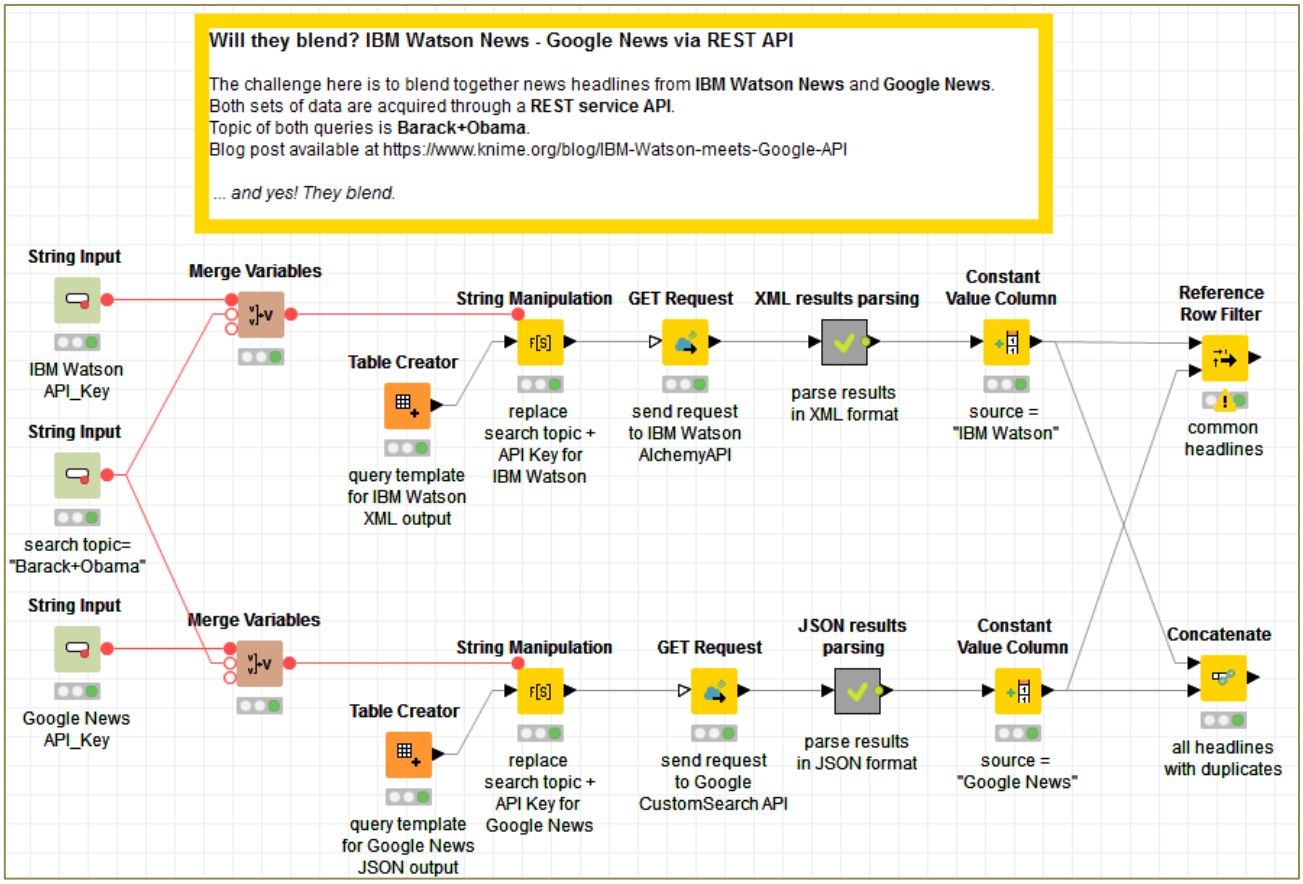
7. The request is then sent to the Google API service through the GET Request node

8. The response is received back in JSON format and parsed with the help of a JSON Path and of an Ungroup node.

Blending News from IBM Watson with News from Google API: final Workflow

The results from both queries are then labeled with their source value (Google News or IBM Watson News) and concatenated together to form a new data table.

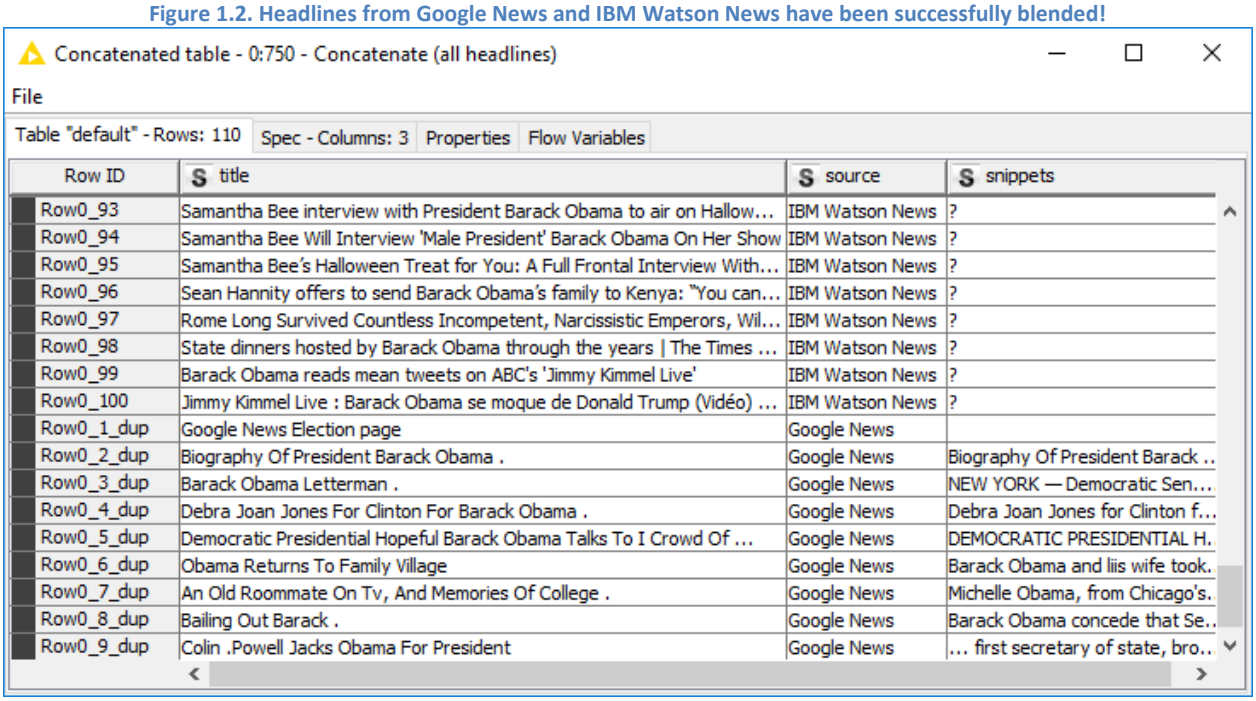
Figure 1.1. This workflow successfully blends news headlines from Google News and IBM Watson News.



Hint: If you had a KNIME Partner Productivity tool (https://www.knime.org/knime-partner-productivity) you would be able to encapsulate and encrypt the String Input nodes containing the API keys. In this way, the end user would be able to run the workflow and query the two REST services, but could not use the API keys for anything else. Since most of you will not have such a KNIME tool, the workflow has been uploaded without encryption and without API keys to the EXAMPLES server under 01_Data_Access/05_REST_Web_Services. The API keys will be your responsibility.

Note: In this workflow we also blend data in XML format with data in JSON format. But this will be the topic for another "Will they blend?" challenge.

Figure 1.2. Headlines from Google News and IBM Watson News have been successfully blended!



Row ID	title	source	snippets
Row0_93	Samantha Bee interview with President Barack Obama to air on Hallow...	IBM Watson News	?
Row0_94	Samantha Bee Will Interview 'Male President' Barack Obama On Her Show	IBM Watson News	?
Row0_95	Samantha Bee's Halloween Treat for You: A Full Frontal Interview With...	IBM Watson News	?
Row0_96	Sean Hannity offers to send Barack Obama's family to Kenya: "You can...	IBM Watson News	?
Row0_97	Rome Long Survived Countless Incompetent, Narcissistic Emperors, Wil...	IBM Watson News	?
Row0_98	State dinners hosted by Barack Obama through the years | The Times ...	IBM Watson News	?
Row0_99	Barack Obama reads mean tweets on ABC's 'Jimmy Kimmel Live'	IBM Watson News	?
Row0_100	Jimmy Kimmel Live : Barack Obama se moque de Donald Trump (Vidéo) ...	IBM Watson News	?
Row0_1_dup	Google News Election page	Google News	
Row0_2_dup	Biography Of President Barack Obama .	Google News	Biography Of President Barack ..
Row0_3_dup	Barack Obama Letterman .	Google News	NEW YORK — Democratic Sen...
Row0_4_dup	Debra Joan Jones For Clinton For Barack Obama .	Google News	Debra Joan Jones for Clinton f...
Row0_5_dup	Democratic Presidential Hopeful Barack Obama Talks To I Crowd Of ...	Google News	DEMOCRATIC PRESIDENTIAL H.
Row0_6_dup	Obama Returns To Family Village	Google News	Barack Obama and liis wife took.
Row0_7_dup	An Old Roommate On Tv, And Memories Of College .	Google News	Michelle Obama, from Chicago's.
Row0_8_dup	Bailing Out Barack .	Google News	Barack Obama concede that Se..
Row0_9_dup	Colin .Powell Jacks Obama For President	Google News	... first secretary of state, bro..

The workflow is available, without the API keys, on the KNIME EXAMPLES server under *01_Data_Access/05_REST_Web_Services/02_IBM_Watson_News–Google_News*.

The Results

Yes, they blend! All in all, we got back 100 headlines from IBM Watson News as the maximum number we had specified, and 10 headlines from Google News - the maximum number the Google API free service allows us to retrieve.

From those headlines we can see that President Barack Obama was very active around Halloween, he can apparently predict the outcome of the NBA tournament, and of course he is actively campaigning with celebrities and politicians. Some of the articles are actually about his wife, Michelle. A somewhat surprising article is the one that talks about Obama's dietary habits "How to eat like Barack Obama". Given that he weathered his years as president remarkably well, maybe it's worth a look?

If we check the intersection of headlines from IBM Watson News and Google News with a Reference Row Filter node, we find that none of the 10 headlines recovered by Google News were also present in the results of the IBM Watson News. It seems, that diversifying news sources in the digital era still makes sense.

But the most important conclusion is: Yes, they blend!

2. Open Street Maps meets CSV Files & Google Geocoding API

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/03_Visualization/04_Geolocation/06_Google_Geocode_API-OSM-text_file
Posted on: November 14, 2016

The Challenge

Today's challenge is a geographical one. Do you know which cities are the most populated cities in the world? Do you know where they are? China? USA? By way of contrast, do you know which cities are the smallest cities in the world?

Today we want to show you where you can find the largest and the smallest cities in the world by population on a map. While there is general agreement from trustworthy sources on the web about which are the most populated cities, agreement becomes sparser when looking for the smallest cities in the world. There is general agreement though about which ones are the smallest capitals in the world.

We collected data for the 125 world's largest cities in a CSV text file and data for the 10 smallest capitals of equally small and beautiful countries in another CSV text file. Data includes city name, country, size in squared kilometers, population number, and population density. The challenge of today is to localize such cities on a world map. Technically this means:

- To blend the city data from the CSV file with the city geo-coordinates from the Google Geocoding API into KNIME Analytics Platform
- Then to blend the ETL and machine learning from KNIME Analytics Platform with the geographical visualization of Open Street Maps.

Topic. Geo-localization of cities on a world map.

Challenge. Blend city data from CSV files and city geo-coordinates from Google Geocoding API and display them on a OSM world map.

Access Mode. CSV file and REST service for Google Geocoding API,

Integrated Tool. Open Street Maps (OSM) for data visualization.

The Experiment

1. First we need to import the data about the top largest cities and top smallest capital cities from the respective CSV text files.

   Data about the 10 smallest capitals in the world were collected from the web site http://top10for.com/top-10-smallest-towns-world/, while data about the world's largest cities were collected from http://www.citymayors.com/statistics/largest-cities-population-125.html and saved to a CSV file. Both CSV files are read using a File Reader node and resulting contents are concatenated together to form a single data table.

2. For each city we build the request to retrieve its latitude and longitude from the Google Geocoding REST API service. Google Geocoding REST API service is free and does not need any kind of authentication. The request will look something like this https://maps.googleapis.com/maps/api/geocode/json?address=City,Country

Where:

maps.googleapis.com/maps/api/geocode/json is the REST service with JSON formatted output
City is the city for which we want to extract longitude and latitude
Country is the country where the city is located

The above request is saved as a template in a Constant Value Column node and appended to each city data row. Each request is then customized through a String Manipulation node. Here "City" and "Country" column values from the input data table replace "City" and "Country" placeholders in the template request.

3. The request is sent to the Google API REST service through the GET Request node.

4. The response is received back in JSON format and parsed with the help of a JSON Path node. At the output of the JSON Path node we find the original city data plus their latitude and longitude coordinates.

5. After cleaning and defining a few graphical properties, such as color, shape, and size of the points to be displayed, city data are sent to an OSM Map View node. This node is part of the KNIME Open Street Map integration and displays points on a world map using their latitude and longitude coordinates.

Figure 2.1. World Map showing the location of the largest (blue squares) and smallest (red triangles) cities in the world as generated by the KNIME Open Street Map Integration.

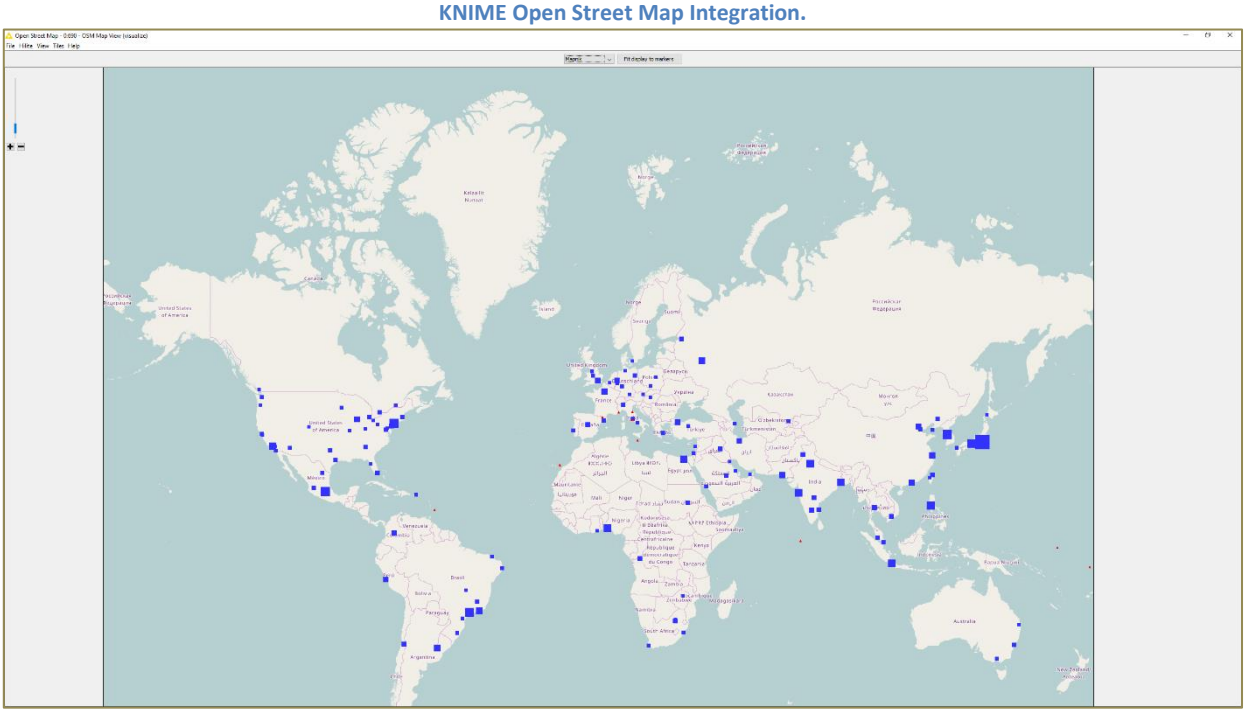


Figure 2.1 shows the world map with the largest (blue squares) and smallest (red triangles) cities in the world. The dot sizes are proportional to the city populations.

The workflow implemented to blend the data from the original CSV files with the responses of the Google Geocoding API and to blend KNIME Analytics Platform and Open Street Map is shown in figure 2.2.

The workflow is available on KNIME EXAMPLES server under
03_Visualization/04_Geolocation/06_Google_Geocode_API-OSM-text_file.

The Results

Yes, they blend!

The largest cities on Earth in terms of population are Tokyo, New York, Sao Paulo, Seoul, and Mexico City. Most of the smallest capitals are located on islands or … in Europe. Europe has the highest number of smallest capital cities of equally small countries, no doubt due to historical reasons. The smallest capital of all is San Marino with only 25000 inhabitants.

Notice the strange case of Rome, included among the largest cities but also including one of the smallest capitals, the Vatican.

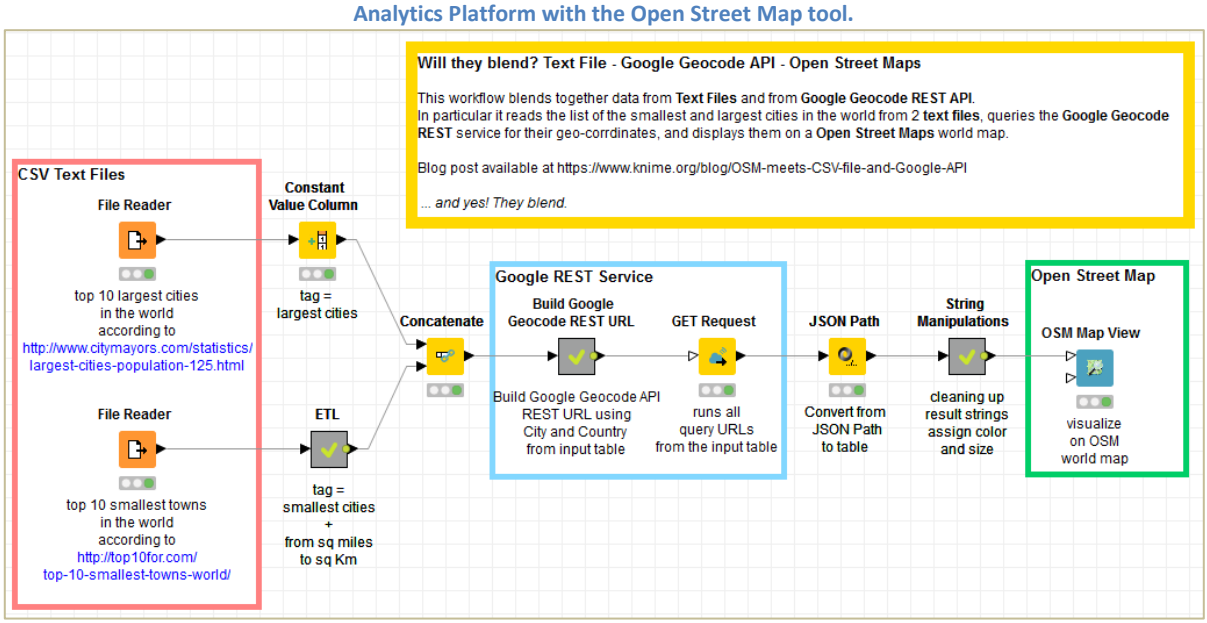
The final data set is the result of merging data from CSV file with data from the Google Geocoding REST API. The most important conclusion then is: Yes, the data blend!

This post, however, describes another level of blending: the tool blending between KNIME Analytics Platform and Open Street Map.

The second conclusion then is: Yes, the two tools also blend!

If we also consider the blending of smallest and largest cities, we could add a third blending level to this post. I am getting carried away.

Figure 2.2. This workflow successfully blends data from CSV files with responses from Google Geocoding API. It also blends the KNIME Analytics Platform with the Open Street Map tool.



3. Hadoop Hive meets Excel. Your Flight is boarding now!

Author: Vincenzo Tursi, KNIME AG
Posted on: November 22, 2016

The Challenge

Today's challenge is weather-based - and something we've all experienced this ourselves while traveling. How are flight departures at US airports impacted by changing weather patterns? What role do weather / temperature fluctuations play in delaying flight patterns?

We're sure the big data geeks at the big airlines have their own stash of secret, predictive algorithms but we can also try to figure this out ourselves. To do that, we first need to combine weather information with flight departure data.

On one side we have the whole archive of US flights over the years, something in the order of millions of records, which we saved on a big data platform, such as Hadoop Hive. On the other side, we have daily US weather information downloadable from https://www.ncdc.noaa.gov/cdo-web/datasets/ in the form of Excel files. So, Hadoop parallel platform on one side and traditional Excel spreadsheets on the other. Will they blend?

Topic. Exploring correlations between flights delays and weather variables

Challenge. Blend data from Hadoop Hive and Excel files

Access Mode. Connection to Hive with in-database processing and Excel file reading

The Experiment

1. We have data for the years 2007 and 2008 of the "airline data set" (http://stat-computing.org/dataexpo/2009/the-data.html) already stored on an Apache Hive platform in a cluster of the AWS (Amazon Web Services) cloud. The "airline data set" has been made available and maintained over the years by the U.S. Department of Transportation's Bureau Transportation of Statistics and it tracks the on-time performance of US domestic flights operated by large air carriers.

2. To access the data in the Apache Hive platform, we use the KNIME nodes for in-database processing. That is, we start with a Hive Connector node to connect to the platform; then we use a Database Table Selector node to select the airline data set table and a Database Row Filter node to extract only Chicago O'Hare (ORD) as the origin airport. The Hive Connector node, like the all other big data connector nodes for Impala, MapR, Hortonworks, etc …, is part of the KNIME Performance Extension and requires a license (https://www.knime.org/knime-performance-extensions).

3. The goal here is to assess the dependency of flight departure delays from the local weather. Weather daily data for all US major locations are available at https://www.ncdc.noaa.gov/cdo-web/datasets/ in the form of Excel files. Weather data for ORD airport only have been downloaded to be joined with the flight records extracted from the Apache Hive data set in the previous step.

4. At this point, we have two options: we upload the climate data into Apache Hive and perform an in-database join on the big data platform or we extract the flight records from Apache Hive into the KNIME Analytics platform and perform the join operation in KNIME. Both options are viable, the only difference being in the execution time of the joining operation.

Option 1. In-database join.

4a. A Hive Loader node imports the weather data from Excel into Hive. For this operation, you can use any of the protocols supported by the file handling nodes, e.g. SSH/SCP or FTP. We chose SSH and used an SSH Connection node.

4b. Thus the Database Joiner node joins by date the weather data for Chicago International Airport with the flight records of the airline dataset.

4c. Now that we have all data in the same table, the Hive to Spark node transfers the data from Hive to Spark. There we run some pre-processing to normalize the data, to remove rows with missing values, and to transform categories into number.

4d. Finally, the Spark Statistics node calculates a number of statistical measures, and the Spark Correlation Matrix node computes the correlation matrix for the selected input columns on the Spark cluster. The resulting table is transferred back for further analysis into the KNIME Analytics Platform.

Option 2. In-KNIME join.

4a. Selected flight data is imported from Hive into the KNIME Analytics Platform using a Database Connection Table Reader node. The Database Connection Table Reader node executes in-database the SQL query at its input port and imports the results into the KNIME Analytics Platform.

4b. Then a Joiner node joins by date the flight data for Chicago Airport with the weather data from the Excel file.

4c. At this point, the same preprocessing procedure as in item 4c is applied.

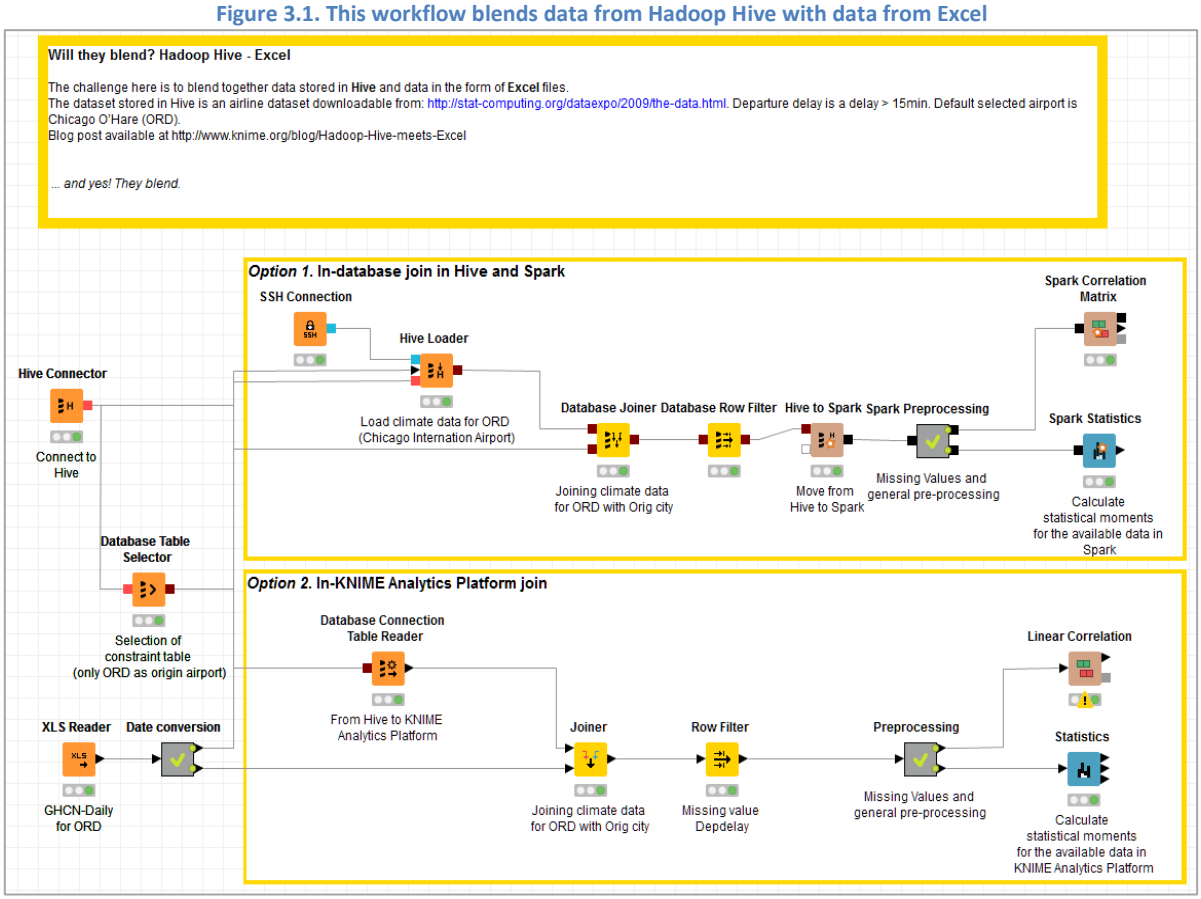
4d. Finally, the Statistics node and the Linear Correlation node calculates the same statistical measures and correlations between departure delays and weather related variables.

The KNIME workflow including a branch for each one of the two options is shown in the figure below.

While the approach described in option 1 is faster, since it takes advantage of parallel computation, the approach described in option 2 is simpler to implement.

In conclusion, you can always choose the best compromise of Hadoop, Spark, and KNIME nodes that fits your problem at hand!

Figure 3.1. This workflow blends data from Hadoop Hive with data from Excel



The Results

Yes, they blend!

The correlation matrix shows some correlation (0.12) between snow alert codes and departure delays in Chicago. The correlation level is not so high, since snow is not the only cause for flight departure delays. Rain, for instance, is also correlated. This is hardly surprising.

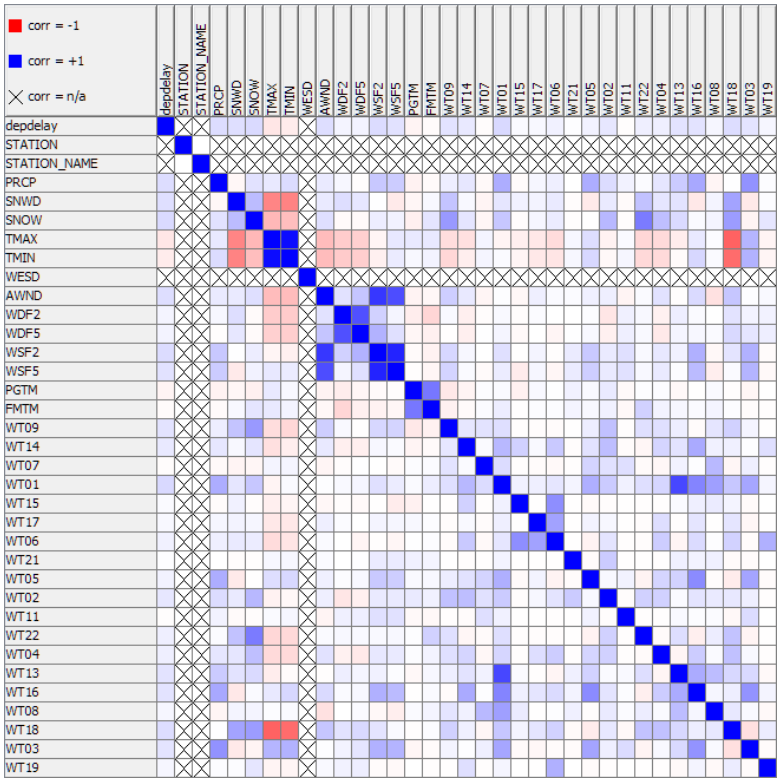
However, what was surprising in this experiment was the easiness to blend data from a modern Hadoop platform (any really) with a traditional Excel file!

Indeed, by just changing the connector node at the beginning of the workflow, the experiment could have run on any other big data platform. Notice that in this case only the connector node would change, the in-database processing part would remain unaltered.

Another surprising conclusion from this experiment is the flexibility of the KNIME analytics Platform and of its Performance Extension. Indeed, mix and match is allowed and it can take you to the optimal degree of Hadoop, Spark, and KNIME, for the best compromise between execution complexity and execution performance.

So, even for this experiment, involving Hadoop Hive on one side and Excel files on the other side, we can conclude that … yes, they blend!

Figure 3.2. Correlation Matrix of flight and weather data



4. Twitter meets PostgreSQL. More than idle Chat?

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/*08_Other_Analytics_Types/04_Social_Media/01_Twitter_meets_PostgreSQL*
Posted on: November 28, 2016

The Challenge

Today we will trespass into the world of idle chatting. Since Twitter is, as everybody knows THE place for idle chatting, our blending goal for today is a mini-DWH application to archive tweets day by day. The tweet topic can be anything, but for this experiment we investigate what people say about #KNIME through the word cloud from last month's tweets.

Now, if you connect to Twitter with a free developer account, you will only receive the most recent tweets about the chosen topic. If you are interested in all tweets from last month for example, you need to regularly download and store them into an archive. That is, you need to build a Data WareHousing (DWH) application to archive past tweets.

As archival tool for this experiment, we chose a *PostgreSQL* database. The DWH application at the least should download and store yesterday's tweets. As a bonus, it could also create the word cloud from all tweets posted in the past month. That is, it should combine yesterday's tweets from Twitter and last month's tweets from the archive to build the desired word cloud.

Summarizing, on one side, we collect yesterday's tweets directly from Twitter, on the other side we retrieve past tweets from a PostgreSQL database. Will they blend?

Topic. What people say about #KNIME on Twitter.

Challenge. Blend together tweets from Twitter and tweets from PostgreSQL database and draw the corresponding word cloud.

Access Mode. Twitter access and Database access.

The Experiment

Defining the *Time Frame*.

1. The metanode named "Time Parameters" creates a number of flow variables with the date of today, yesterday, the day before yesterday, and one month ago. These parameter values will be used to extract the tweets for the word cloud.

Accessing Twitter.

1. First of all, we created a Twitter developer account at https://apps.twitter.com/. With the creation of the account, we received an API Key with API Key secret and an Access Token with Access Token secret. Let's remember those, because we will need them to connect to Twitter and download the tweets. Details on how to create a Twitter developer account can be found in this past KNIME blog post https://www.knime.org/blog/knime-twitter-nodes.
2. In a new workflow, we created a Twitter API Connector node to connect to Twitter using the API Key and the Token Access we got. We then used a Twitter Search node to download all tweets around a given topic: the hashtag #KNIME. This topic string is generated by the String Input Quickform node as a flow variable and passed to the Twitter Search node. After execution, this last node produces at its output port the most recent tweets on that topic, with tweet body, time, user, location, and other related information.

3. After some time format conversion operations and a Row Filter node, only yesterday's tweets are kept and saved in a table named "tweets" in a PostgreSQL database.

Accessing PostgreSQL database (or any database for that matter).

1. PostgreSQL is one of the many databases with a dedicated connector node in KNIME Analytics Platform. This made our job slightly easier. To connect to a PostgreSQL database we used the PostgreSQL Connector node, where we provided the host URL, the database name, and the database credentials.

2. The recommended way to provide the database credentials is through the Workflow Credentials. In the KNIME Explorer panel, right-click the workflow and select Workflow Credentials. Provide a credential ID, username, and password. Then in the PostgreSQL Connector node enable option "Use Credentials" and select the ID of the just created credential.

3. Once the connection to the database had been established, we used a series of in-database processing nodes to build the SQL query for the data extraction, such as the Database Table Selector, to select table "tweets", and two Database Row Filter nodes, to select the tweets in the archive posted between one month ago and the day before yesterday.

4. The last node, Database Connection Table Reader node, runs the query on the database and exports the results into KNIME Analytics Platform.

Blending data from Twitter and PostgreSQL database.

1. Now the workflow has yesterday's tweets from Twitter and last month's tweet from the PostgreSQL archive. The Concatenate node puts them together.

2. After some cleaning and relative frequency calculation, the word cloud is built by the Tag Cloud node.

Scheduling on KNIME Server

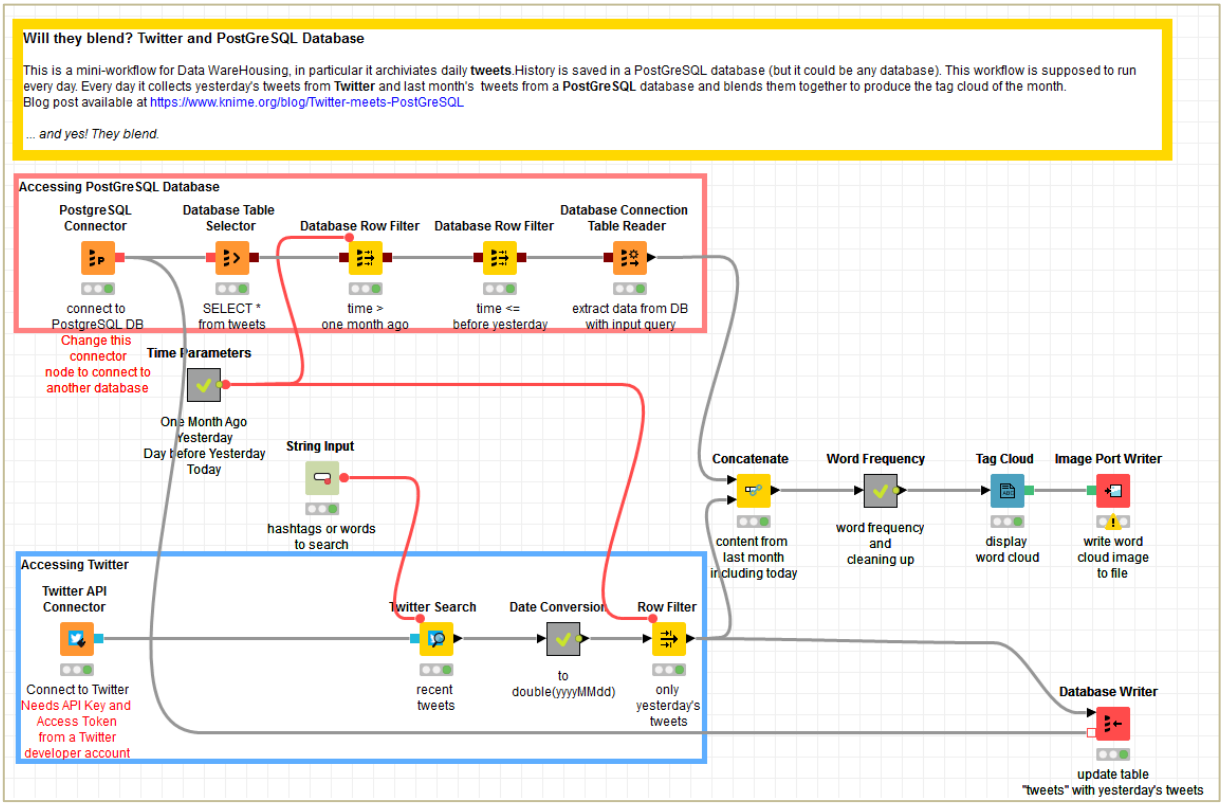
1. When the workflow was finished, it was moved onto the KNIME Server (just a drag & drop action) and there it was scheduled to run every day once a day.

Note. If you are using *any other database*, just change the connector node. In the category Database/Connector in the Node Repository, you can search for the connector node dedicated to your database of choice. If you use a not so common or an older version database and a dedicated connector is not available, use the generic Database Connector node. This should connect to all databases, provided that you have the right JDBC driver file.

The workflow that blends data from Twitter and PostgreSQL and builds this mini DWH application is displayed in figure 4.1 and is downloadable from the KNIME EXAMPLES server under
08_Other_Analytics_Types/04_Social_Media/01_Twitter_meets_PostgreSQL.

Hint. Did you see the round connections? This s a preview of how connections between nodes could look like in the new KNIME Analytics Platform 3.3 to be released on Dec 6th. Are you ready? In case you are not, join our webinar "What is new in KNIME Analytics Platform 3.3" on Dec 13, 18:00 CET.

Figure 4.1. This workflow implements a mini-DWH application that blends yesterday's tweets directly from Twitter and last month's tweets from a PostgreSQL database and builds the corresponding word cloud.



The Results

Yes, they blend!

The resulting word cloud for tweets posted between October 31st and November 24th 2016 and containing the hashtag #KNIME, is shown in figure 4.2. Looks familiar? If you have been at the KNIME Spring Summit 2016 in Berlin, this word cloud, built from the tweets containing hashtag #KNIMESummit2016, was projected during the event pauses. By the way, we will use this same workflow to generate the word cloud at the upcoming KNIME Spring Summit 2017 in Berlin on March 15-16, this time around the hashtag ***#KNIMESummit2017***.

In the word cloud we notice the predominance of the hashtags *#datascience*, *#data*, *#bigdata*, and *#datablending*, which describe the whole span of the KNIME Analytics Platform realm.

Surprising are Microsoft Azure username and hashtag. They come from the tweets about the Microsoft Roadshow, which travelled across Europe in October and November and saw KNIME as one of the involved partners.

Another unexpected username is OReilly. This originates from tweets about the newly released and announced O'Reilly media course "Introduction to Data Analytics with KNIME".

Figure 4.2. Word cloud generated from tweets posted between Oct 31st and Nov 24th 2016 with hashtag #KNIME.

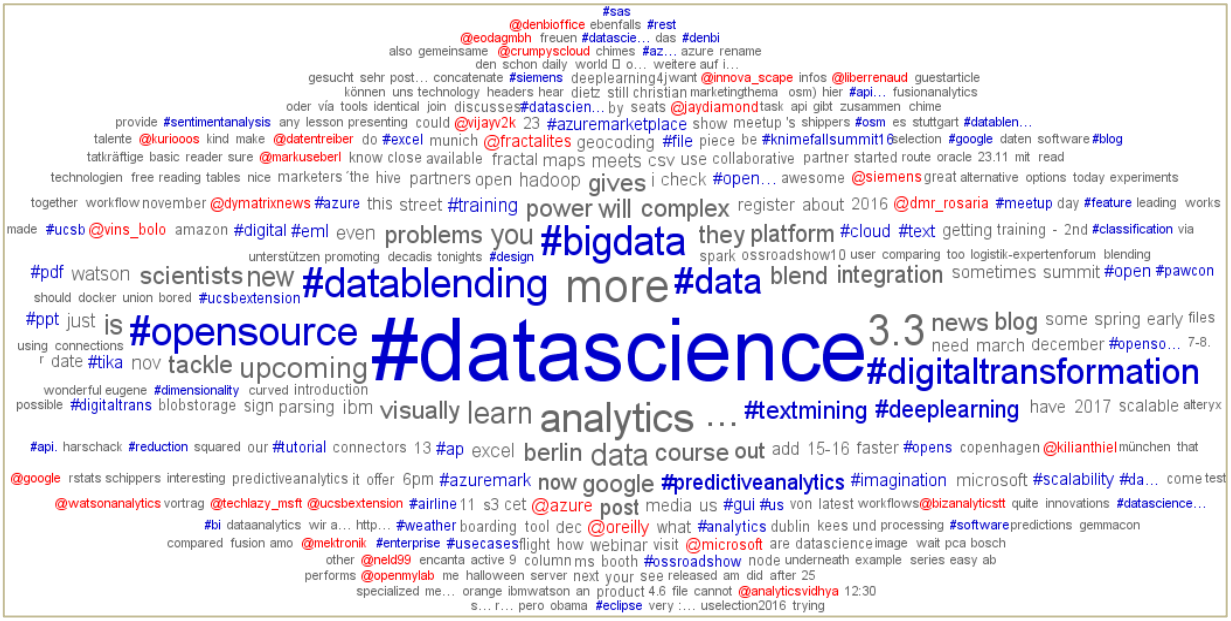
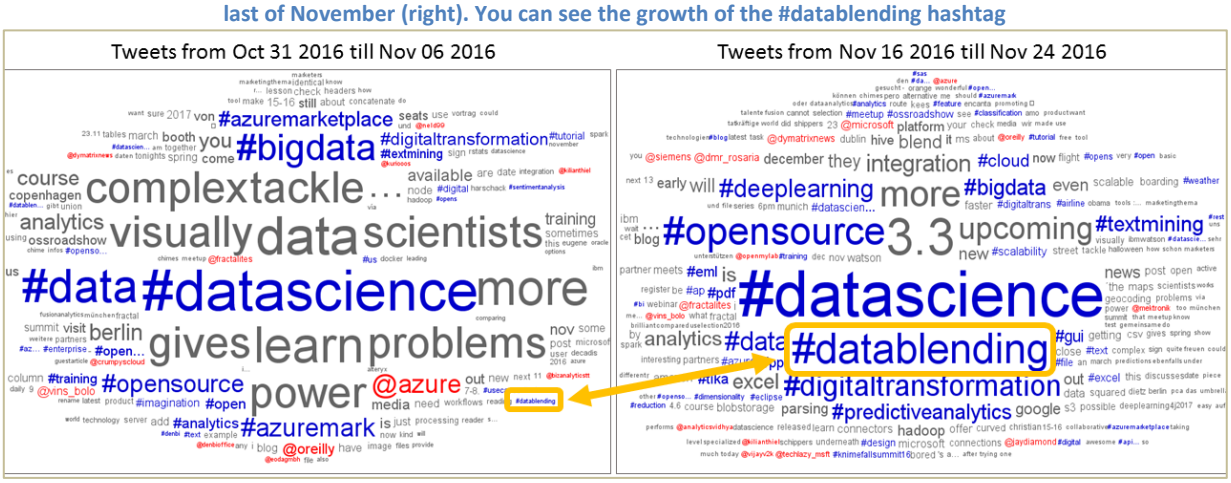


Figure 4.3. The word cloud generated from tweets with hashtag #KNIME posted in the first week of November (left) and in the week before last of November (right). You can see the growth of the #datablending hashtag



One word about the *#datablending* hashtag. It has not always been that prominent up there. Let's compare, for example, the tag cloud of the first week in November – when the "Will they blend?" blog post series had just started – with the tag cloud of the week before last again in November, as in figure 4.3. You can see the growth of the hashtag *#datablending*, which directly reflects the impact of this blog series. Thanks for tweeting!

Any more words that surprise you in this word cloud? Did you find your username?

This word cloud is the proof of the successful blending of data from Twitter and data from a PostGreSQL database. The most important conclusion of this experiment is: Yes, they blend!

5. A Cross-Platform Ensemble Model. R meets Python and KNIME. Embrace Freedom in the Data Science Lab!

Author: Vincenzo Tursi, KNIME AG
Workflow in: EXAMPLES/*04_Analytics/13_Meta_Learning/ 04_Cross-Platform_Ensemble_Model*
Posted on: December 13, 2016

The Challenge

Today's challenge consists of building a cross-platform ensemble model. The ensemble model must collect a Support Vector Machine (SVM), a logistic regression, and a decision tree. Let's raise the bar even more and train these models on different analytics platforms: R, Python, and of course KNIME.

A small group of three data scientists was given the task to predict flight departure delays from Chicago O'Hare (ORD) airport, based on the airline data set. As soon as the data came in, all data scientists built a model in record time. I mean, each one of them built a different model on a different platform! We ended up with a Python script to build a logistic regression, an R script to build an SVM, and a KNIME workflow to train a decision tree. Which one should we choose?

We had two options here: select the best model and claim the champion; embrace diversity and build an ensemble model. Since more is usually better than less, we opted for the ensemble model. Thus, we just needed to convince two out of the three data scientists to switch analytics platform.

Or maybe not.

Thanks to its open architecture, KNIME can easily integrate R and Python scripts. In this way, every data scientist can use his/her preferred analytics platform, while KNIME collects and fuses the results.

Today's challenge has three main characters: a decision tree built on KNIME Analytics Platform, an SVM built in R, and a logistic regression built with Python. Will they blend?

Topic. Flight departure delays from Chicago O'Hare (ORD) Airport.

Challenge. Build a cross-platform ensemble model, by blending an R SVM, a Python logistic regression, and a KNIME decision tree.

KNIME Extensions. Python and R Integrations.

The Experiment

Data Access.

1. We used data from the "airline data set" (available at: http://stat-computing.org/dataexpo/2009/the-data.html) for the years 2007 and 2008. The data were previously enriched with external information, such as cities, daily weather (https://www.ncdc.noaa.gov/cdo-web/datasets/), US holidays, geo-coordinates, and airplane maintenance records. The Table Reader node is used to read the data set.

Preprocessing.

1. *Binning*. Some columns, such as distance and departure and arrival times, were binned into only a few segments.
2. *Missing Values*. Data rows with no Departure Delay value were removed. Other missing values were substituted with specific knowledge-based fixed values. Then, all string values were converted into numbers.
3. *Partitioning*. Data rows referring to year 2007 were used as training set; data rows referring to year 2008 were used as the test set.
4. *Normalization*. SVM requires normalized input values in [0,1].
5. DepDelay column was selected as target to train the models.

Model Training.

1. A **Decision Tree Learner** node was used to train a decision tree on KNIME Analytics Platform.
2. An **R Learner** node was implemented to run the R script that trains a Support Vector Machine in R. Please notice that the R svm() function has scalability issues and therefore a Row Sampling node is used to reduce the training set size to 10000 randomly extracted rows.
3. A **Python Learner** node was introduced to execute the Python script to train a logistic regression model using Python Scikit library.
4. The three nodes were placed inside a metanode to produce a KNIME PMML model, an R model, and a Python model.
5. Next, the subsequent metanode runs the appropriate Predictor nodes to evaluate the models on the test set and joins the prediction columns into a single data table. Here the **Prediction Fusion** node combines all predictions through the median. Other operators are also available.

The Prediction Fusion node is only one way to build an ensemble model. Another way would include transforming each model into a PMML data cell (using, for example, the R to PMML node and the JPMML-SkLearn Python library), collecting all models into a data table with a Concatenate node, transforming the data table into an ensemble model with a Table to PMML Ensemble node, and finally creating the ensemble predictions with a PMML Ensemble Predictor node. However, for this experiment we preferred to use the Fusion Prediction node, as it is more flexible for the integration of external models.

Figure 5.1. This workflow builds a cross-platform ensemble model by blending together three different models from three different analytics platforms, i.e. an SVM from R, a Logistic Regression from Python and a decision tree from KNIME. In this case, the ensemble predictions are calculated by a Prediction Fusion node on the basis of the predictions produced by the three single models.

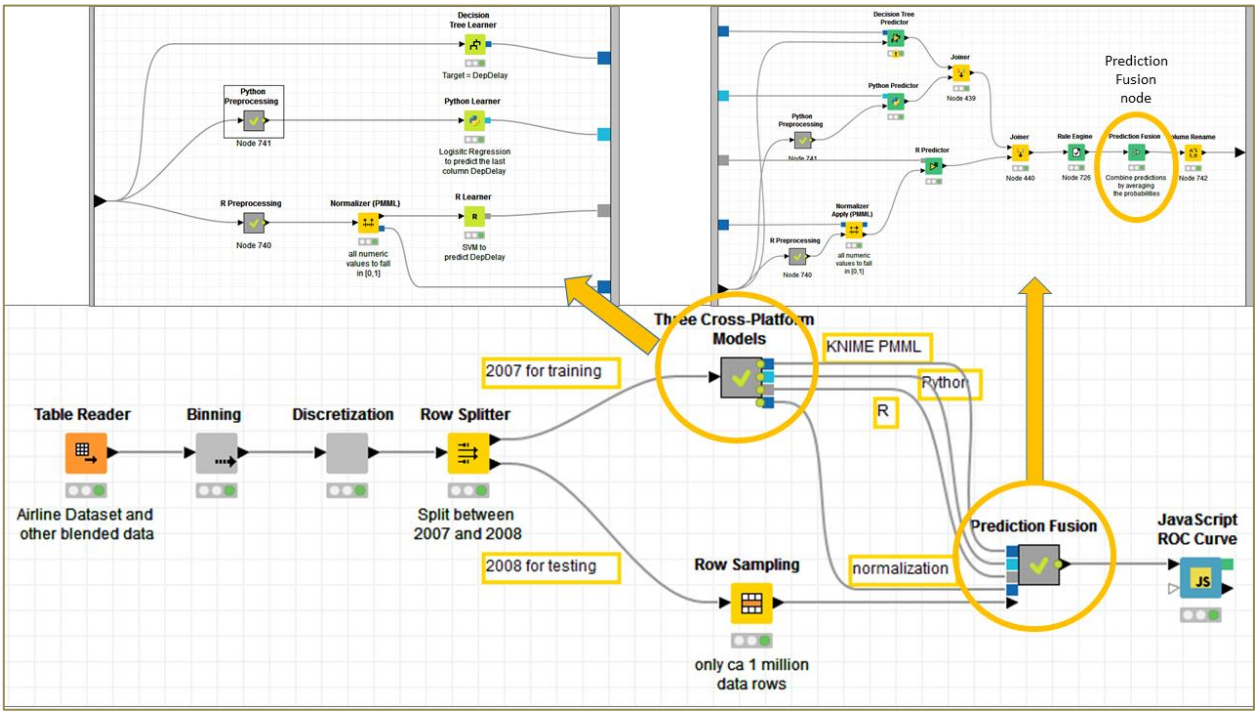
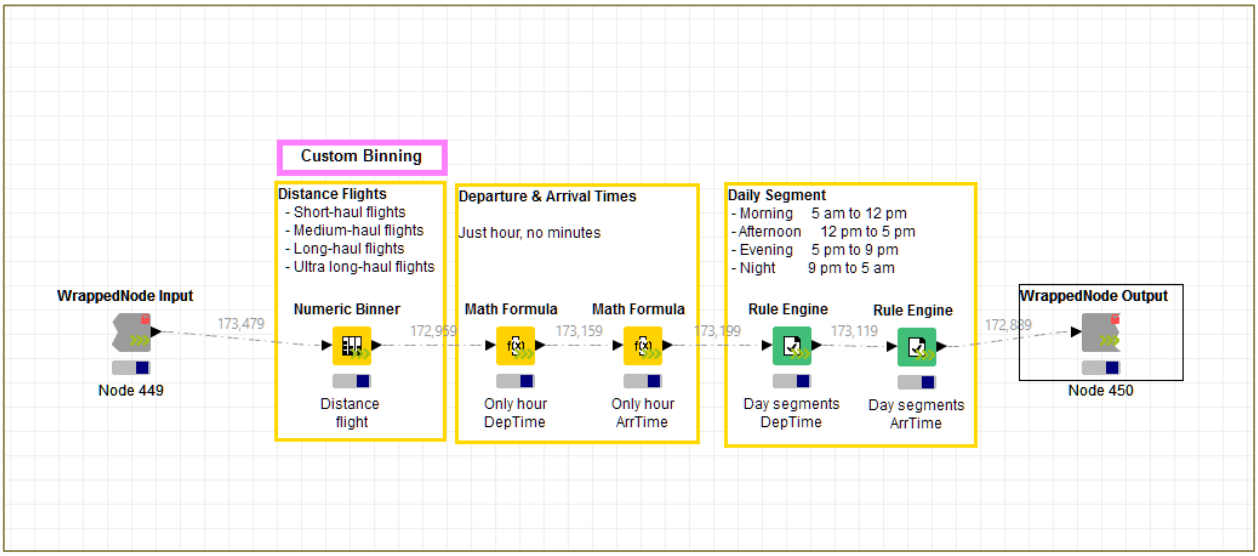


Figure 5.2. The Streaming Executor engine executes nodes concurrently inside the wrapped node.



Note. Have a look at the first wrapped node in the workflow, the one called “Binning”. Can you see the arrow in its lower right corner? This is a node that runs on the KNIME Streaming Executor. It executes its nodes in parallel as much as possible. When the first node has finished processing the current data package, the subsequent node can already start execution on the transmitted data. This cuts down on I/O and memory usage as only the few rows 'in transit' need to be taken care of, which should speed up calculations.

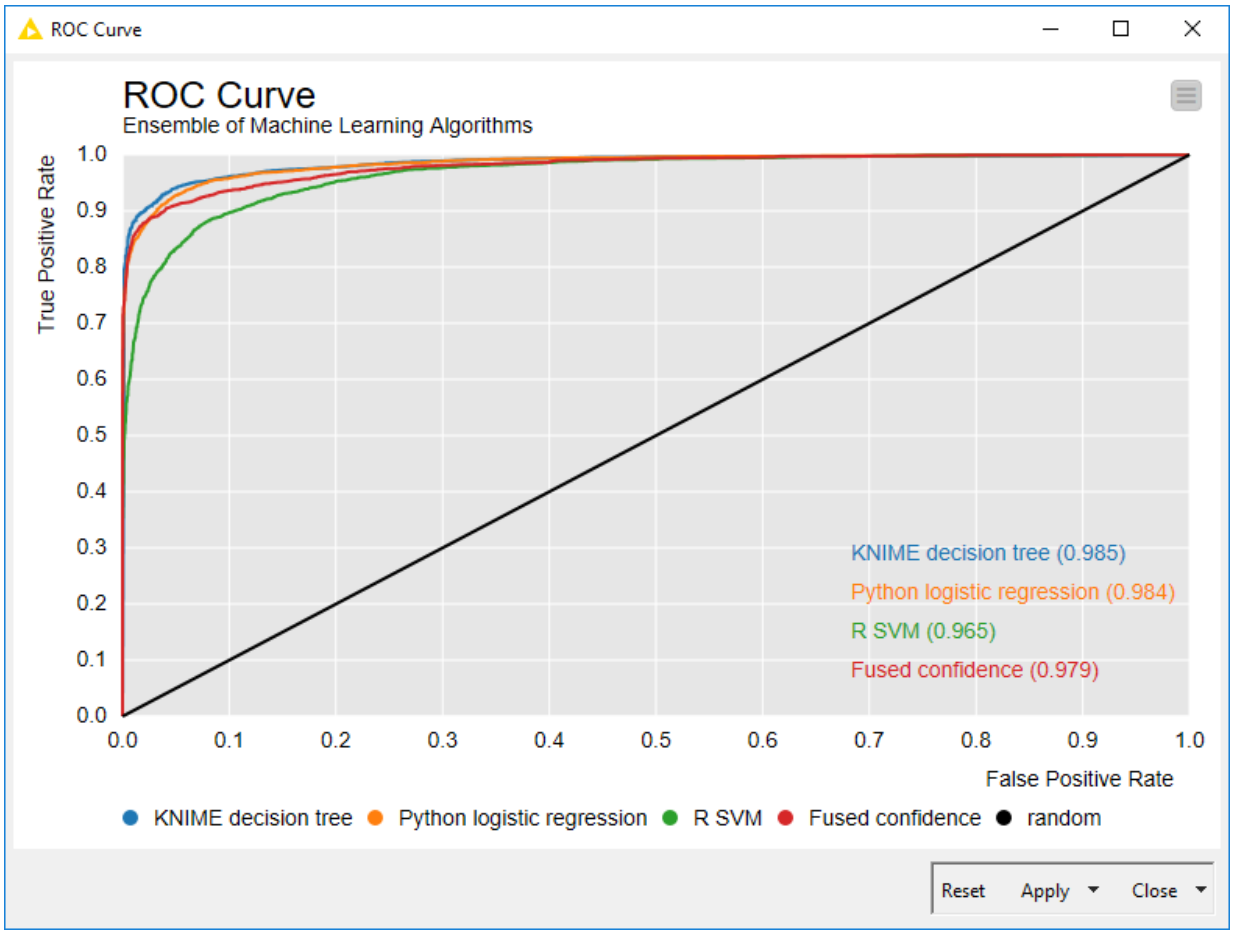
The Results

Yes, they blend!

Platform blending as well as model blending were successful!

The last step was to assess the quality of the ensemble model vs. the single models. To do this, we used the Java Script ROC Curve node. Our cross-platform ensemble model shows performances in between those of its single parts, as you can see from the ROC curves below.

Figure 5.3. ROC curves of the cross-platform ensemble model (Fused confidence) and of the single model components (KNIME decision tree, Python Logistic Regression, and R SVM).



With this experiment we proved that KNIME Analytics Platform enables freedom in the Data Science Lab by allowing data scientists to use their preferred analytics platform and by integrating models and results into a KNIME workflow.

The workflow used for this experiment is available on the KNIME EXAMPLES server under *04_Analytics/13_Meta_Learning/ 04_Cross-Platform_Ensemble_Model*

6. MS Word meets Web Crawling. Identifying the Secret Ingredient.

Author: Heather Fyson & Roland Burger, KNIME AG
Workflow in: EXAMPLES/*08_Other_Analytics_Types/01_Text_Processing/10_Discover_ Secret_Ingredient*
Posted on: December 20, 2016

The Challenge

It's Christmas again, and like every Christmas, I am ready to bake my famous Christmas cookies. They're great and have been praised by many! My recipes are well-kept secrets as I am painfully aware of the competition. There are recipes galore on the web nowadays. I need to find out more about these internet-hosted recipes and their ingredients, particularly any ingredient I might have overlooked over the years.

My recipes are stored securely in an MS Word document on my computer, while a very well regarded web site for cookie recipes is http://www.handletheheat.com/peanut-butter-snickerdoodles/. I need to compare my own recipe with this one on the web and find out what makes them different. What is the secret ingredient for the ultimate Christmas cookie?

In practice, on one side I want to read and parse my Word recipe texts and on the other side I want to use web crawling to read and parse this web-hosted recipe. The question as usual is: will they blend?

Topic. Christmas cookies

Challenge. Identifying secret ingredients in cookie recipes from the web by comparing them to my own recipes stored locally in an MS Word document.

Access Mode. MS Word .docx file reader and parser and web crawler nodes.

The Experiment

Parsing the Word document

The Text Processing Extension of KNIME Analytics Platform version 3.3 includes the Tika library with a Tika Parser node that can read and parse virtually any kind of document: docx, pdf, 7z, bmp, epub, gif, groovy, java, ihtml, mp3, mp4, odc, odp, pptx, pub, rtf, tar, wav, xls, zip, and many more!

The KNIME Text Processing extension can be installed from KNIME Labs Extension / KNIME TextProcessing. (A video on YouTube explains how to install KNIME extensions, if you need it.)

Reading the MS Word file

1. After installing the KNIME Text Processing extension, we take the Tika Parser node in KNIME Labs/Text Processing/IO to read and parse the content of the SnickerdoodleRecipes.docx file. The content of the file is then available at the node output port.
2. The Strings to Document node converts a String type column into a Document type column. Document is a special data type necessary for many Text Processing nodes. After that, a Sentence Extractor node splits the document into the different recipes in it.

Extracting the list of ingredients

1. Next, we extract the paragraph with the cookie ingredients. This is implemented through a series of dedicated String Manipulation nodes, all grouped together in a metanode called “Ingredients only”.
2. The “Text Processing” metanode works on the ingredient sections after they have been transformed into Documents: Punctuation Erasure, Case Conversion, Stop Word Filtering, Number Filtering, and N Char Filtering.
3. Inside the same “Text Processing” metanode, a POS Tagger node tags each word according to its part of speech. Thus the Tag Filter node keeps only nouns and discard all other parts of speech. Finally, the Stanford Lemmatizer node reduces all words to their lemma, removing their grammar inflections.
4. The data table at the output port of the “Text Processing” metanode now contains only nouns and is free of punctuation, numbers, and other similarly uninteresting words (i.e. non-ingredients). We now use the Bag of Words node to move from the Document cells to their list of words.
5. Duplicate items on the list are then removed by a GroupBy node in the Unique List metanode.

Crawling the web page

Here we use a KNIME Community Extension, the Palladian extension, to crawl the web. It’s available under KNIME Community Contributions – Other / Palladian for KNIME. (A video on YouTube explains how to install KNIME extensions.) After installing it, we proceed to crawl the page http://www.handletheheat.com/peanut-butter-snickerdoodles/.

Crawling the web page.

1. First a Table Creator node is introduced to contain the URL of the web page.
2. Then the HttpRetriever node from the Palladian extension is applied to retrieve the content of the web page.
3. The HtmlParser node uses the Validator.nu HTML Parser to extract the content of an HTML document and to store it in an XML data cell.
4. Two XPath nodes extract the recipe from the XML document.

Now that we have the recipe text, we repeat the same steps described above in “Extracting the list of ingredients”.

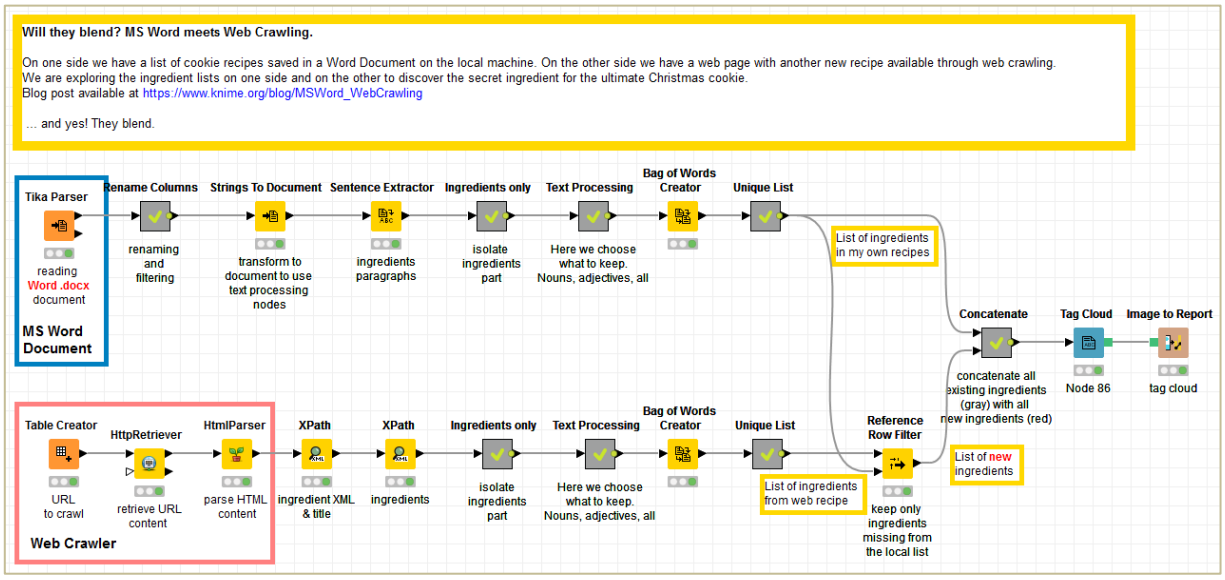
Comparing the two lists of ingredients

Two lists of ingredients have emerged from the two workflow branches. We need to compare them and discover the secret ingredient – if any – in the new recipe as downloaded from the web.

1. A Reference Row Filter node compares the new list of ingredients with the list of ingredients from the existing recipes in the Word document. The ingredients from the old recipes are used as a reference dictionary. Only the new ingredients, missing from the previous recipes but present in the web-derived recipe, are transmitted to the node output port.
2. New and old ingredients are then labeled as such and concatenated together to produce a color-coded word cloud with the Tag Cloud node. Here, new ingredients are highlighted in red while old ingredients are gray.
3. Finally, the word cloud image is exported to a report, which is shown in figure 6.2.

The workflow used for this experiment is available for download from the EXAMPLES server as usual under *08_Other_Analytics_Types/01_Text_Processing/10_Discover_ Secret_Ingredient.*

Figure 6.1. This workflow successfully blends ingredients from recipes in a Word document with ingredients from a recipe downloaded from the Web.



The Results

Yes, they blend!

The main result is that data from a MS Word document and data from web crawling do blend! We have also shown that ingredients from recipes in a Word document and ingredients from web-based recipe also blend!

But what about the secret ingredient for the ultimate cookie? Well, thanks to this experiment I have discovered something I did not know: peanut butter cookies. If you check the final report in figure 6.2, you can see the word "peanut" emerging red from the gray of traditional ingredients. I will try out this new recipe and see how it compares with my own cookie recipes. If the result is a success I might add it to my list of closely guarded secret recipes!

Happy Holidays everyone!

Figure 6.2. The report resulting from the workflow above, identifying peanut (in red) as the secret ingredient of the cookie competition.

Credit for the web hosted recipe goes to Tessa.
More infos at http://www.handletheheat.com/peanut-butter-snickerdoodles/

7. Local vs. Remote Files. Will Blending overcome the Distance?

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/*01_Data_Access/06_ZIP_and_Remote_Files/03_Local_vs_remoteHTTP*
Posted on: January 23, 2017

The Challenge

Today's challenge is distance: physical, geographical distance … between people and between compressed files.

Distance between people can easily be solved by any type of transportation. A flight before Christmas can take you back to your family just in time for the celebrations. What happens though if the flight is late? Better choose your airline carrier carefully to avoid undesired delays!

Distance between compressed files can easily be solved by KNIME. A few appropriate nodes can establish the right HTTP connection, download the file, and bring it home to the local files.

The goal is to visualize the ratio of departure delays in Chicago airport by carrier through a classic bar chart. We will take the data from the airline dataset (http://stat-computing.org/dataexpo/2009/the-data.html) and we will focus on two years only: 2007 and 2008. I worked on this dataset for another project and I already have the data for year 2008 zipped and stored locally on my laptop. I am missing the data for year 2007 but I can get them via the URL of the original web site http://stat-computing.org/dataexpo/2009/2007.csv.bz2.

So on the one hand I have a ZIP file with the 2008 data from the airline data set here on my laptop. And on the other side I have a link to a ZIP file with the 2007 data on some server in some remote location, possibly close to the North Pole. Will KNIME fill the distance? Will they blend?

Topic. Departure delays by carrier

Challenge. Collect airline data for 2007 and 2008 and display departure delay ratio by carrier from Chicago airport

Access Mode. One file is accessed locally and one file is accessed remotely via an HTTP connection

Integrated Tool. JavaScript based visualization.

The Experiment

Access the LOCAL file for year 2008 of airline data

Airline data for year 2008 have already been downloaded from http://stat-computing.org/dataexpo/2009/the-data.html onto my machine a few weeks ago for a previous experiment. Data was still zipped. So, I used:

- an Unzip Files node to unzip the file content into the knime.workspace folder
- a classic File Reader node to read the content of the unzipped file and import it into the KNIME workflow.

Access the REMOTE file via HTTP connection for year 2007 of airline data

The data for 2007 still have to be downloaded. They were still available on the original URL http://stat-computing.org/dataexpo/2009/2007.csv.bz2 so I could download them via an HTTP connection.

All nodes that deal with remote files can be found in IO/File Handling/Remote in the Node Repository. This sub-category contains nodes to upload, download, delete, change files in a remote location. From KNIME Analytics Platform 3.3 you'll also find connectors for Amazon S3 and Microsoft Blob Store files.

In this case:

- We first established an HTTP connection to the server URL (http://stat-computing.org) through an HTTP Connection node
- We then downloaded the required file using the Download node
- The downloaded file was compressed. So we used the Unzip Files node to extract it to a local location
- Finally, we used a classic File Reader node, as in item number 1, to read the file content and import it into a KNIME data table

Blend the two data sets

Now, the lower branch of the workflow (Fig. 7.1) deals with the 2008 airline data from the local file, while the upper branch handles the 2007 airline data from the remote file. After removing all cancelled flights on both sides, we used a Concatenate node to put both data sets into a single data table.

The following metanode defines departure delay, isolates flights originating in Chicago airport, and calculates the ratio of departure delays by carrier, which the JavaScript Bar Chart node then interactively visualizes (Fig. 7.2).

The workflow is available on the KNIME EXAMPLES server under
0*1_Data_Access/06_ZIP_and_Remote_Files/03_Local_vs_remoteHTTP*.

The Results

Yes, they blend!

By looking at the chart we can see that if you had taken an ExpressJet (EV) flight from Chicago in 2007 you would have been delayed at departure one out of two times. Things would have looked better though one year later in 2008. Delta and North West seemed to be the most reliable airlines when departing from Chicago O'Hare airport respectively in 2007 and 2008.

In this post we can safely conclude that KNIME has overcome the distance problem between two compressed files and successfully blended them to build a bar chart about airline departure delay ratios.

Again the most important conclusion is: Yes, they blend!

Figure 7.1. This workflow successfully blends data from a local and a remote file location. The remote file is downloaded through an HTTP connection and then unzipped and read like the local file.

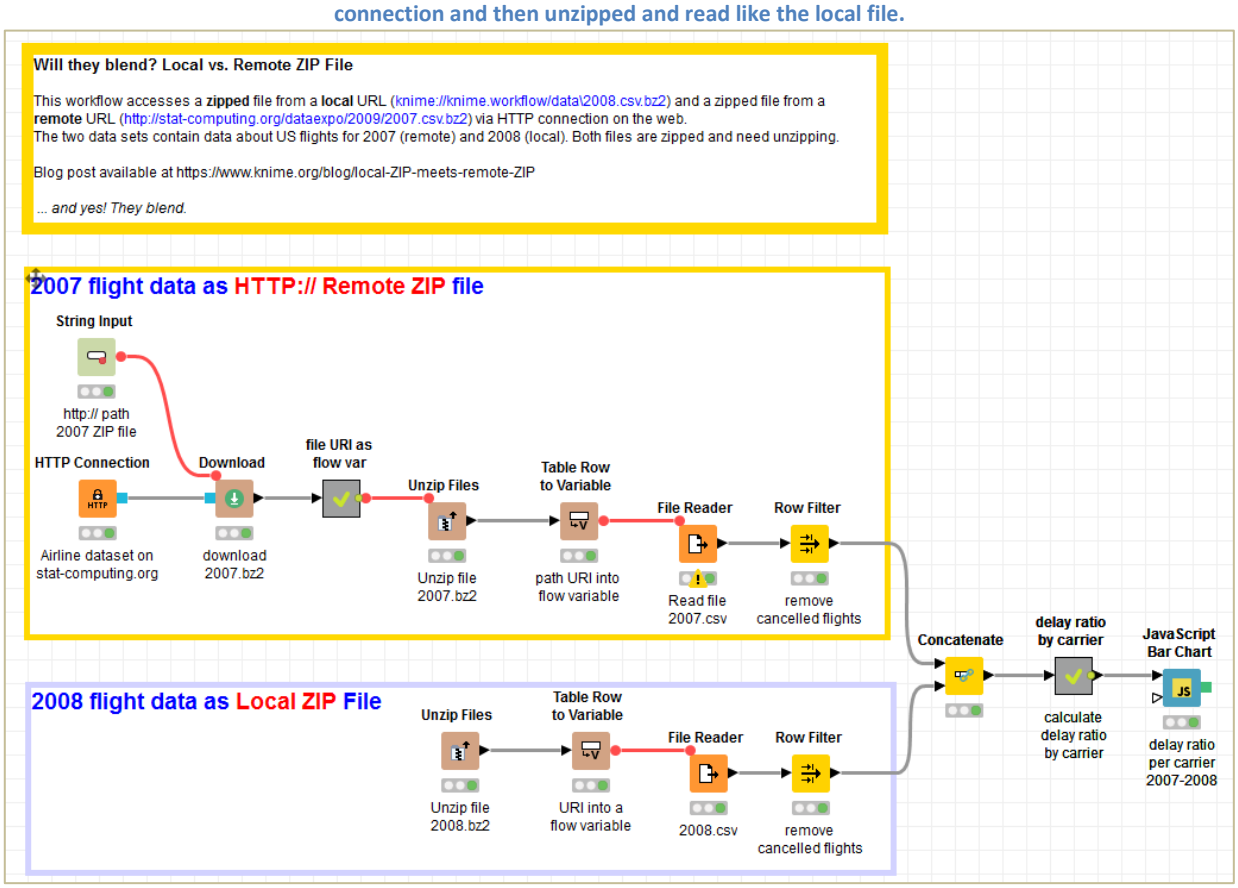
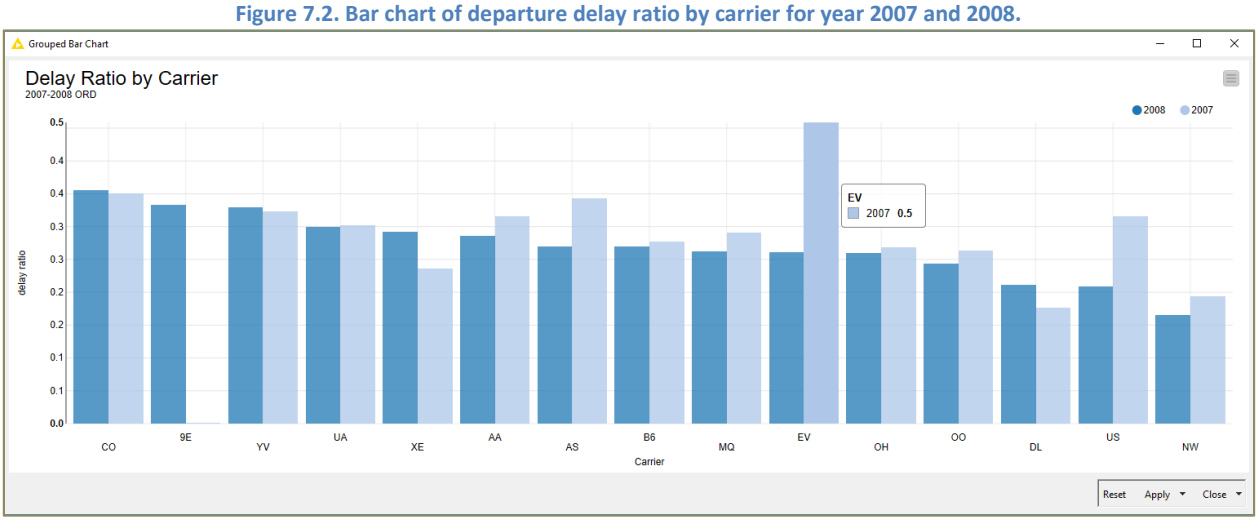


Figure 7.2. Bar chart of departure delay ratio by carrier for year 2007 and 2008.



8. Amazon S3 meets MS Azure BlobStorage. A Match made in the Clouds.

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/*01_Data_Access/06_ZIP_and_Remote_Files/04_AmazonS3-MSBlobStorage_Census_Data*
Posted on: January 30, 2017

The Challenge

Today let's leave the ground and move into the clouds! When we talk about clouds, two options come immediately to mind: the Amazon cloud and the MS Azure cloud. Both clouds offer proprietary bulk repositories (data lakes) to store data: Amazon cloud uses the S3 data repository and MS Azure has loaded the Blob Storage data repository.

Let's suppose now that for some unpredictable twist of fate, we have ended up with data on both clouds. How could we make the two clouds communicate, so as to collect all data in a single place? It is a well-known fact that clouds rarely talk to each other.

Today's challenge is to force the Amazon cloud and the MS Azure cloud to communicate and exchange data. That is, we want to blend data stored in an S3 data lake on the Amazon cloud with data stored in a Blob Storage data lake on the MS Azure cloud. Will they blend?

In the process, we will also put a few Excel files into the blender, just to keep our feet on the ground: after all every data analytics process has to deal with an Excel file or two.

Topic. .Analyze the commuting time of Maine workers from the new CENSUS file

Challenge. Blend together CENSUS file ss13hme.csv about homes in Maine and hosted on S3 on the Amazon cloud with file ss13pme.csv about people in Maine and hosted on Blob Storage on the MS Azure cloud.

Access Mode. Connection to Amazon S3 and connection to MS Blob Storage.

The Experiment

The key nodes here are the Connection and the File Picker nodes; respectively the Amazon s3 Connection node and the Amazon File Picker node for one cloud service and the Azure Blob Store Connection node and the Blob Store File Picker node for the other cloud service.

1. ***From Amazon S3***
    a. First, the Amazon S3 Connection node connects to the Amazon S3 service using the credentials and hostname provided in its configuration window.
    b. Next, the Amazon S3 File Picker node builds the full path of the resource selected on S3 and exports it as a flow variable. The selected file was ss13hme.csv.
    c. Finally, a File Reader node uses the content of that flow variable to overwrite the URL of the file to be read, therefore accessing the file on the Amazon S3 platform.
2. ***From MS Azure Blob Storage (same procedure with dedicated nodes as for Amazon S3)***
    a. First, the Azure Blob Store Connection node connects to the Azure Blob Storage service using the credentials and hostname provided in its configuration window.

b. Next, the Azure Blob Store File Picker node builds the full path of the resource selected on Blob Storage and exports it as a flow variable. Selected file was ss13pme.csv.

c. Finally, a File Reader node uses the content of that flow variable to overwrite the URL of the file to be read, therefore accessing the file on the Azure Blob Storage platform.

3. **Now both data sets are available as data tables in the KNIME workflow.**

   a. ss13pme.csv contains a number of randomly interviewed people in the state of Maine, which should correspond to 1% of the Maine population.

   b. ss13hme.csv contains information about the house in which the interviewed people live.

   c. In both files, the interviewed people are uniquely identified by the SERIAL NO attribute.

   d. The 2 data sets are then joined on SERIAL NO.

4. We focused on the ENG, SEX, and JWMNP attributes. ENG is a code that represents fluency in the English language; SEX is a code for the person's sex; and JWMNP contains the travel time to work. The idea is to see if there is any interesting pattern across such features. A full explanation of all attributes contained in the dataset can be downloaded from:
   http://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMSDataDict15.pdf.
   The new CENSUS dataset is publicly available and can be downloaded from http://www.census.gov/programs-surveys/acs/data/pums.html . More big data sources can be found in this article by Bernard Marr: "Big data: 33 Brilliant and Free data Sources for 2016".

5. Missing values in SEX and ENG are substituted with special code 99; JWMNP is renamed as "Travel Time to Work".

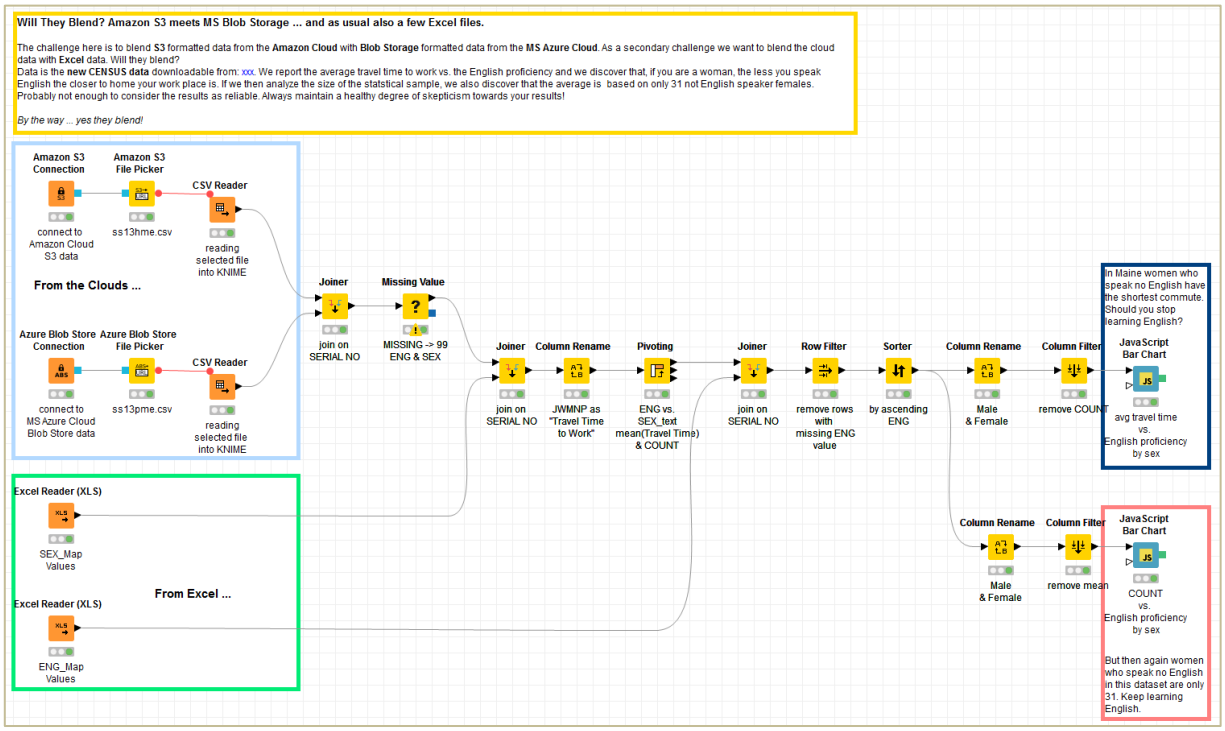
6. A Pivoting node builds the two matrices: avg(JWMNP) of ENG vs. SEX and count of ENG vs. SEX.

7. SEX codes (1, 2) are mapped respectively to Male and Female, while ENG codes (1, 2, 3, 4) are mapped to text Strings describing fluency in the English language. Look up tables are stored in local Excel files.

8. Finally, a Javascript Bar Chart node displays the average travel time for females and males depending on their English proficiency. Another Javascript Bar Chart node displays the number of people for each group of males/females and their fluency in the English language.

This workflow blends data from Amazon S3 and MS Azure Blob Storage and it is downloadable from the KNIME EXAMPLES server under:
01_Data_Access/06_ZIP_and_Remote_Files/04_AmazonS3-MSBlobStorage_Census_Data

Figure 8.1. This workflow blends data from Amazon S3 and data from MS Azure Blob Storage. As added bonus, it also throws in some Excel data as well.



The Results

Yes, they blend!

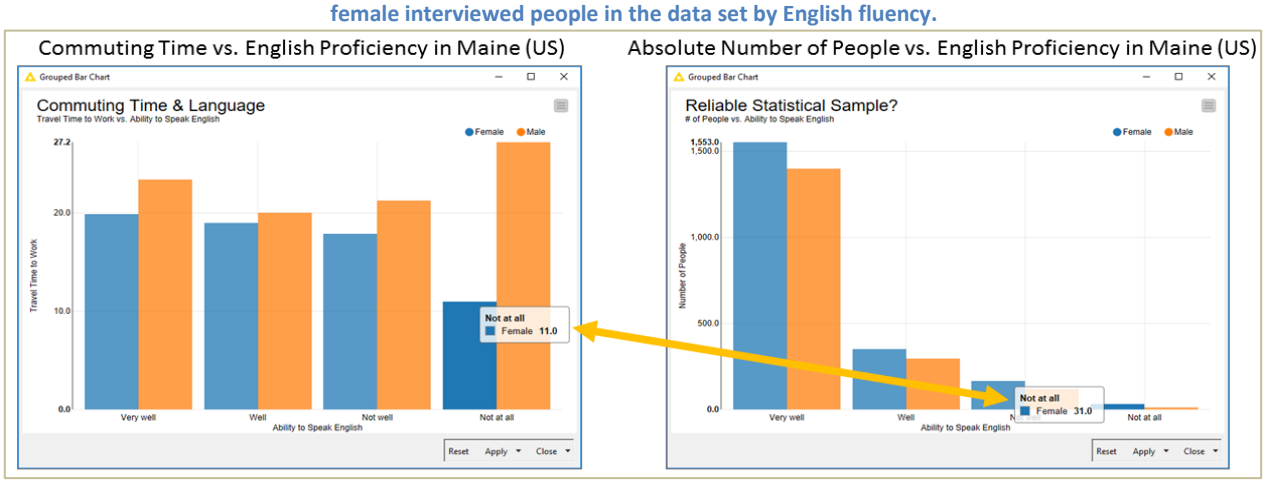
We were able to blend together the data coming from the ss13hme.csv file from Amazon S3 and the data from the file ss13pme.csv from Azure Blob Storage. We were also able to isolate different groups of people based on their gender and their fluency in English. Results have been displayed in the bar plot on the left in figure 8.2.

According to these results, we can see that if you are a woman in Maine and speak no English, your commute to work seems to be much shorter than the commute of other women who speak better English. If you are a woman who lives in Maine and speaks no English yet, my advice would be not to learn it farther if you want to keep your commute short in the future. Wait a moment … this does not make much sense!

Just to double-check, let's see how big each one of those groups is. The number of interviewed people by sex and English knowledge is reported in figure 8.2 on the right. Well, the number of non-English speaking women consists of only 31 subjects. Not really the biggest statistical sample I have ever seen. Maybe, just maybe, the conclusions we drew above are not that reliable after all!

Anyway, our conclusions might have unsubstantiated, but the experiment was still successful, because … yes they blend!

Figure 8.2. On the left: average Travel Time to Work by English proficiency for female and male people; on the right: number of male and female interviewed people in the data set by English fluency.



Advice. As this story has been teaching us, always keep a healthy degree of skepticism about the results produced by your analysis. There is always the possibility that a mistake or a too small data group even in a very large data set might have invalidated your results.

If you have believed the first conclusions, do not worry! Everybody can make this mistake. A full web site, named Spurious Correlations http://www.tylervigen.com/spurious-correlations, is indeed dedicated to all such impossible ridiculous conclusions that could be easily unmasked just by using some common sense.

If you want to know more about best practices in the data analytics world, to avoid the rookie mistakes, you might be interested in Dean Abbott's talk and workshop at the upcoming KNIME Spring Summit 2017 in Berlin on March 15-16.

9. MS Access meets H2. Test your Baseball Knowledge.

Author: Vincenzo Tursi, KNIME AG
Workflow in: EXAMPLES/01_Data_Access/02_Databases/06_MSAccess_meets_H2
Posted on: February 6, 2017

The Challenge

Today's challenge is sport related. How well do you know Major League Baseball? Do you know who had the best pitching and batting statistics in the decade 1985-1990? Do you know who has been the highest- paid baseball player of all times?

Baseball has been for a long time, and arguably still is, the most data focused sport. The most famous usage of data analytics in Major League Baseball is for sure documented in the Moneyball movie (see the "Breaking Biases" scene), but there have been many other cases.

For this challenge, we used batting and pitching statistics for all players active from 1985 to 2015 in the two baseball leagues, i.e. the National League and the American League. Such data has been made publicly available through the Sean Lahman's website. We would actually like to use this chance to thank all the site contributors for making this standard baseball encyclopedia publicly available. The Lahman Database stores player statistics as well as data about managers, birthdates, awards, all-star games, and much more.

In most companies every department owns specific data, sometimes even using different separated databases. For instance, salaries and demographic data are often owned by HR, while performance metrics are owned by Operations. In this experiment, we mimic the HR Department to host salaries and demographics data on a MS Access database and the Operations Department to host the performance metrics (batting and pitching stats) on a H2 database.

MS Access is part of the Microsoft Office package and therefore available on most Windows based PCs. H2 is a relatively new open source database downloadable for free at http://www.h2database.com/html/download.html. Therefore both databases are quickly accessible and commonly used in single departments or small-to-medium businesses.

Today's technical challenge is to attempt a data blending from a MS Access database and an H2 database. Will they blend?

Afterwards, on the blended data, we will take to a short guided tour on the KNIME WebPortal, to detect the best-paid and/or best-performing baseball players for each decade.

Topic. Best paid and best performing baseball players from 1985 to 2015.

Challenge. Blend data from MS Access and H2 databases and guide the users through the analytics on a web browser.

Integrated Tool. Database Connectors and KNIME WebPortal.

The Experiment

Accessing MS Access database

1. To access the MS Access database we rely on the UCanAccess driver. UCanAccess is a pure Java JDBC Driver implementation, which allows Java developers and JDBC client programs to read/write from and to Microsoft Access databases (.mdb and .accdb files). Download the driver file from http://ucanaccess.sourceforge.net/site.html.
2. Copy the required jar files (*ucanacces-xxx.jar*, *jackcess-xxx.jar*, *commons-lang-xx.jar*, *commons-logging-xxx.jar*, *hsqldb.jar*) into <KNIME_INSTALL>\jre\lib\ext
3. Register the driver jar (*<KNIME_INSTALL>\jre\lib\ext\ucanacces-xxx.jar*) into File → Preferences → Databases in KNIME Analytics Platform.The driver should then be visible in the database driver list in the Database Connector node.
4. In the workflow, we used a List Files node and a String Manipulation node to create the UCanAccess JDBC URL as *"jdbc:ucanaccess://",$Location$, ";showSchema=true"*.
5. The UCanAccess JDBC URL is then fed to the Database Connection node through the Table Row to Variable node.
6. Then, two Database Table Selector nodes select the Salaries table and the Master table, containing respectively the players' salaries and demographics information.
7. Finally, two Database Connection Reader nodes export the data into KNIME Analytics Platform.

Accessing H2 database

1. To connect to the H2 database, we used the H2 Connector node.
2. After connecting to the database, two Database Table Selector nodes select the Batting table and the Pitching table.
3. Finally, two Database Connection Reader nodes export the data into KNIME Analytics Platform.

Blending data from MS Access and H2 databases

After some transformation and cleaning in "Preprocessing and joining data" wrapped node, such as computation and normalization of the average stats for each player, we created a sequence of 4 webpages to display on the KNIME WebPortal:

a. **Page 1** visualizes the average salary for baseball players of the two major baseball leagues, i.e. the National League and the American League, over time.

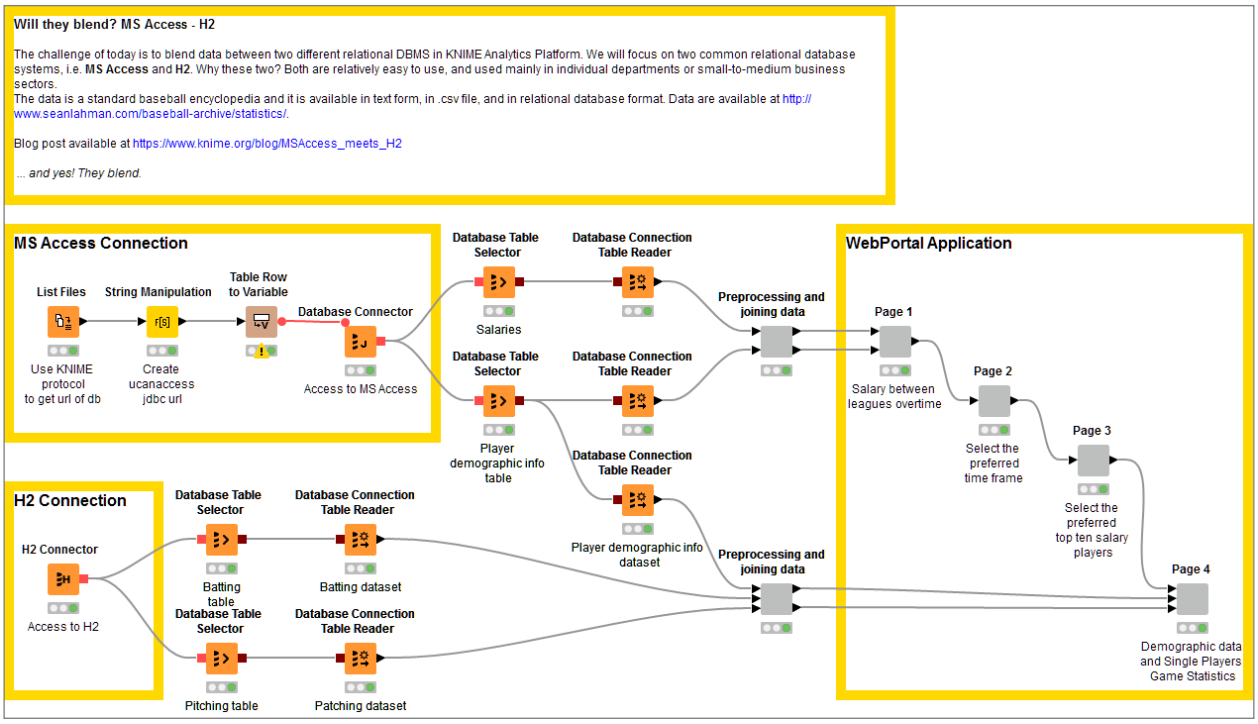
b. **Page 2** allows the user to select a 5-year time frame between 1986 and 2015.

c. **Page 3** displays the top 10 best paid players for the selected time frame in both National and American League and allows you to select some of them.

d. **Page 4** numerically describes the selected players through average statistics and radar plots.

The final workflow is available for download on the EXAMPLES server under *knime://EXAMPLES/01_Data_Access/02_Databases/06_MSAccess_meets_H2*.

Figure 9.1. This workflow blends data from MS Access database and H2 database, on the left. After blending, the last 4 wrapped nodes on the right implement 4 web pages to guide data selection and description on the KNIME WebPortal.



Note. For any other database, just change the connector node. In the category, Database/Connector in the Node Repository, a number of dedicated connector nodes are available. If you cannot find yours, you can always resolve to the generic Database Connector node. The Database Connector node can connect to all databases, if the database JDBC driver file is provided.

The Results

Yes, they blend! The database blending was successful. The question though is: are the players worth the money?

The scatter plot in page 1 on the WebPortal (Fig. 9.2) shows the growth of the average salary over the years. That is quite some growth! Was it worth it?

Let's select a time frame on page 2, for example 1985 to 1990.

On page 3, let's select 3 of the top 10 best paid baseball players active between 1985 and 1990: Frank Viola, Orel Hershiser and Cal Ripken. Were they pitching or batting phenomena? Or both?

The answer to our question is in page 4. The tables in Fig. 9.3 and 9.4, extracted from page 4, summarize the batting and pitching stats in a radar plot for each player. Clearly, Cal Ripken was paid for his batting skills, while Orel Hershiser and Frank Viola exhibited fantastic pitching stats.

Now it's your turn! Who is your favorite baseball player? Using our workflow, you can check out his batting and pitching stats, directly in KNIME or on a web browser!

This WebPortal application is the proof of the successful blending of data from MS Access and H2 database. The most important conclusion of this experiment is again: Yes, they blend!

The workflow is available on the KNIME EXAMPLES server under *knime://EXAMPLES/01_Data_Access/02_Databases/06_MSAccess_meets_H2*.

Figure 9.2. Page 1 on the WebPortal. This scatter plot shows the growth of the average salary in baseball from 1985 to 2015

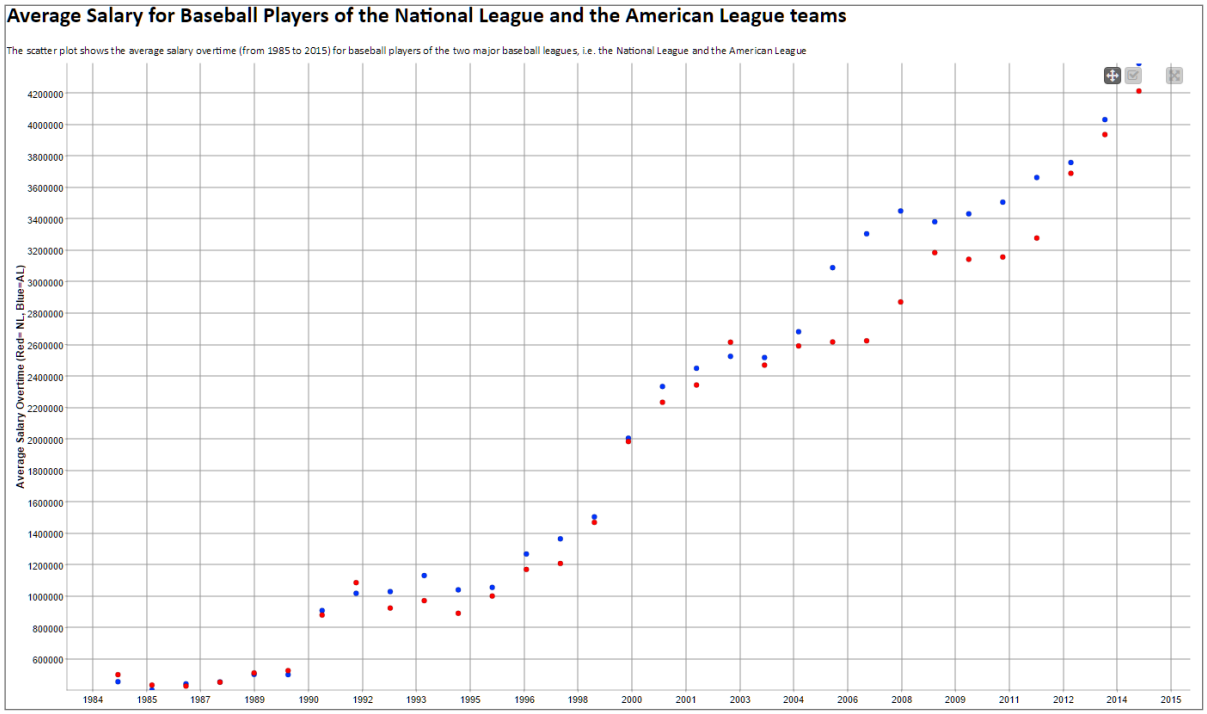
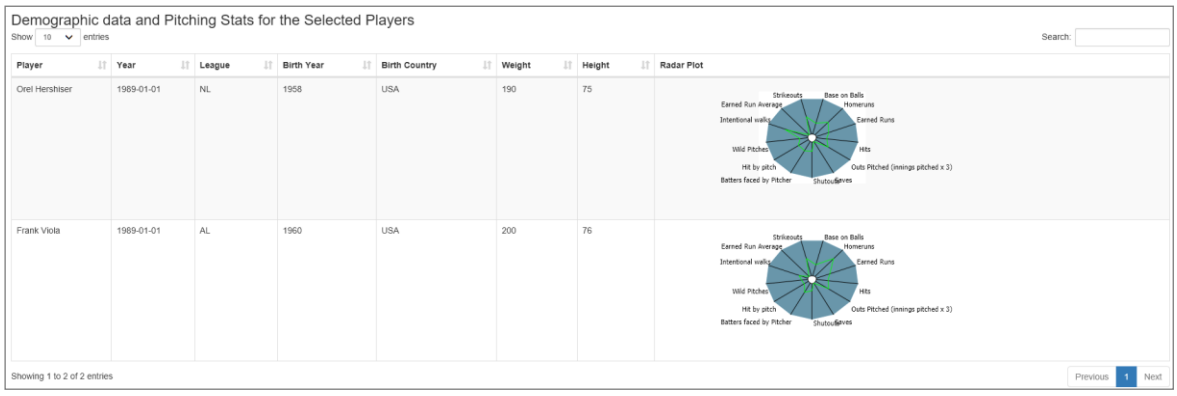


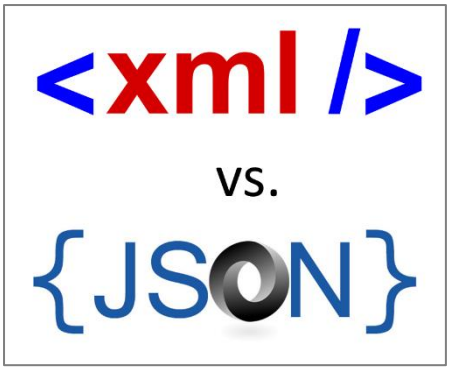
Figure 9.3. The table shows demographic data and batting stats for the selected players

Figure 9.4. The table shows demographic data and pitching stats for the selected players



10. XML meets JSON.

Author: Rosaria Silipo, KNIME AG
Workflow in: EXAMPLES/04_Analytics/13_Meta_Learning/ 04_Cross-Platform_Ensemble_Model
Posted on: February 20, 2017



The Challenge

Do you remember the first post of this series? Yes, the one about blending news headlines from IBM Watson News and Google News services about Barack Obama? (https://www.knime.org/blog/IBM-Watson-meets-Google-API). Blending the news headlines involved a little side blending – i.e. blending JSON structured data – the response of Google News – with XML structured data – the response from IBM Watson News.

Today, the challenge is to parse and blend XML structured data with JSON structured data. Recycling part of the original blog post workflow, we query IBM Watson AlchemyAPI News service for the first 100 news headlines on Barack Obama and the first 100 news headlines on Michelle Obama. We want the response for Barack Obama to be received in XML format and the response for Michelle Obama in JSON format. Two data sets: one for Barack Obama (XML) and one for Michelle Obama (JSON). Will they blend?

Topic. News Headlines on Barack Obama and separately on Michelle Obama from October 2016.

Challenge. Blend the two data response data sets respectively in XML and JSON format.

Access Mode. JSON and XML parsing for IBM Watson AlchemyAPI News REST service response.

The Experiment

1. We build a request for the IBM Watson AlchemyAPI REST service GetNews, as described in the previous post https://www.knime.org/blog/IBM-Watson-meets-Google-API inserting the API Key, the topic, and the output format as parameters.

   Notice that we encapsulated the whole REST service request part in two identical wrapped metanodes, both named “Querying IBM Watson”. The advantage of the wrapped metanode is that the Quickform parameters are shown in its configuration window. This means it is easy to change the topic in the metanode configuration window without needing to access the internal content of the wrapped metanode. The same parameters shown in the wrapped node configuration window are shown in a web browser through KNIME WebPortal. Finally, since the metanodes are identical except for the parameter values, they could actually be replaced with links to wrapped node templates located on some KNIME Server, making maintenance of the workflow and its metanodes much easier.

   The output format is set to XML for Barack Obama and to JSON for Michelle Obama.

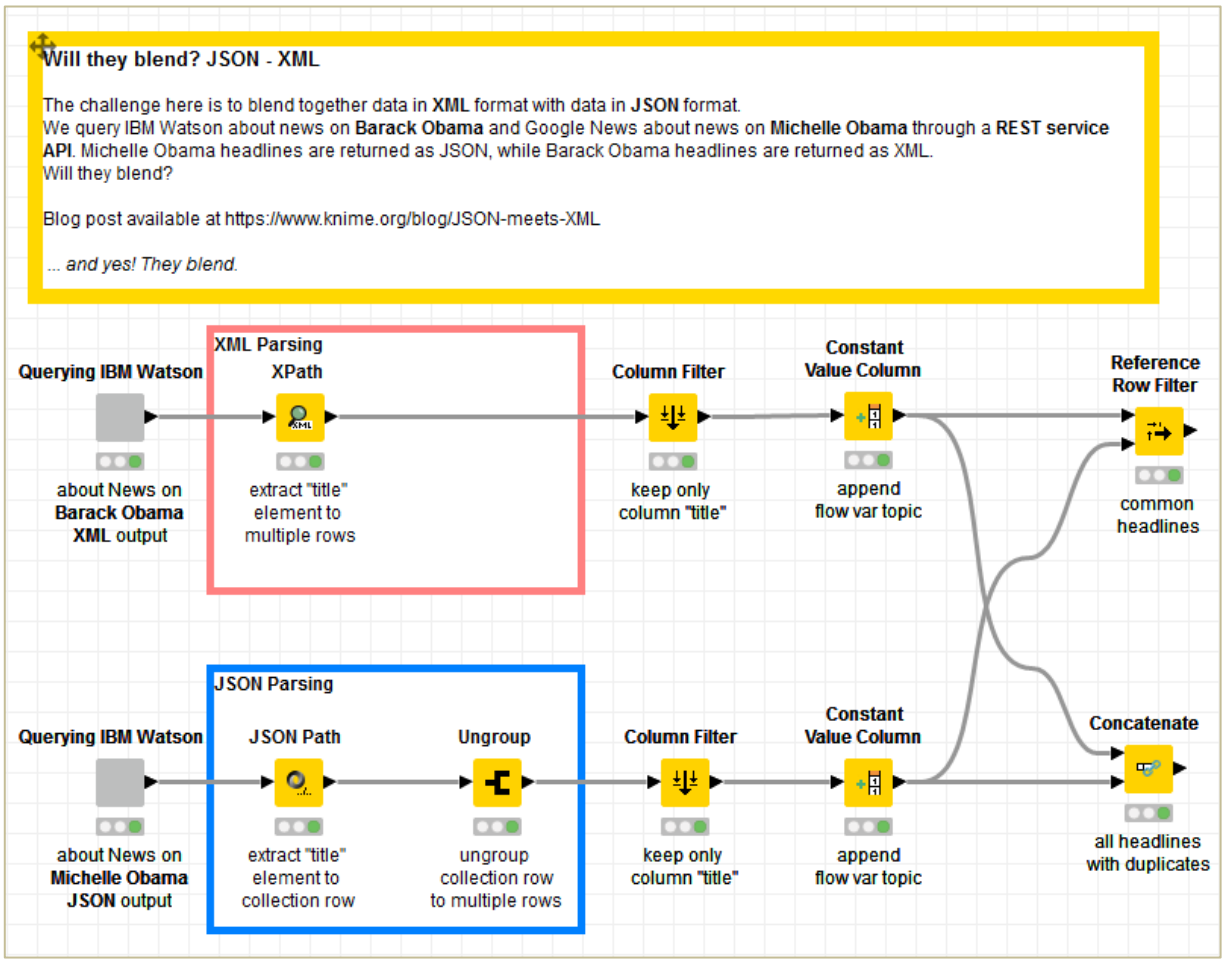
2. The XPath node uses the following query /results/result/docs/element/source/enriched/url/title to extract all news titles from the XML response. To process the array of news headlines properly, it is important to set the return type to String (Cell) and to allow for multiple rows in the Multiple Tag Options frame in the configuration window of the XPath node.

3. The JSON Path node uses the following query $['result']['docs'][*]['source']['enriched']['url']['title'] to extract all news headlines into a collection-type data cell. Since the XPath node does not have the "multiple rows" option, we use an Ungroup node to transform the collection cell into a number of separate data rows.

4. The experiment is practically finished with 100 headline titles in the upper branch for Barack and 100 in the lower branch for Michelle. We can now concatenate the results for further processing or we could, for example, extract the common headlines reporting about both Obamas.

The workflow that blends Barack's XML-structured headlines and Michelle's JSON-structured headlines in KNIME Analytics Platform is shown in figure 10.1. A version of this workflow reading IBM Watson responses from a file is available on the KNIME EXAMPLES server under *01_Data_Access/04_Structured_Data/03_XML_meets_JSON*.

Figure 10.1. This workflow successfully blends JSON-structured news data about Michelle Obama with XML-structured news data about Barack Obama.



The Results

Yes, they blend!

The experiment was run in November 8, 2016. 100 news headlines were extracted between October 22 and November 8 for each of the Obamas. At that time, the Obamas had only one common headline at the output of the Reference Row Filter node.

The workflow successfully blended XML-structured data for Barack with JSON-structured data for Michelle. Again, the most important conclusion is: Yes, they blend!

11. SAS, SPSS, and Matlab meet Amazon S3. Setting the Past free.

Author: Phil Winters, CIAgenda
Workflow in: EXAMPLES/01_Data_Access/06_ZIP_and_Remote_Files/05_SAS_SPSS_MATLAB_meet_S3
Posted on: February 27, 2017

The Challenge

I am an old guy. And old guys grew up with proprietary data formats for doing cool data science things. That means I have literally 100s of SAS, SPSS and MATLAB files on my hard disk and I would still love to use that data - but I no longer have the cash for the yearly licenses of those venerable old packages.

I know I can read all those files for free with KNIME. But what I REALLY want to do is move them into an open format and blend them somewhere modern like Amazon S3 or Microsoft Azure Blob Storage. But when I check out the various forums, like the SAS one, I find only horrendously complicated methods that - oh by the way - still requires an expensive license of the appropriate legacy tool. So it's KNIME to the rescue and to make it easy, this example pulls files of all three legacy types and allows you to either move them or first convert them to an open source format before moving them to - in this example - Amazon S3.

It's easy, it's open, it's beautiful…. and it's free.

The Experiment

The Setup

First, you will need your credentials or Access Key/Secret Key for Amazon S3. If you don't have one or are new to Amazon S3, the folks have made it easy for you to sign up and try for free. Just follow the instructions here: www.amazon.com/S3.

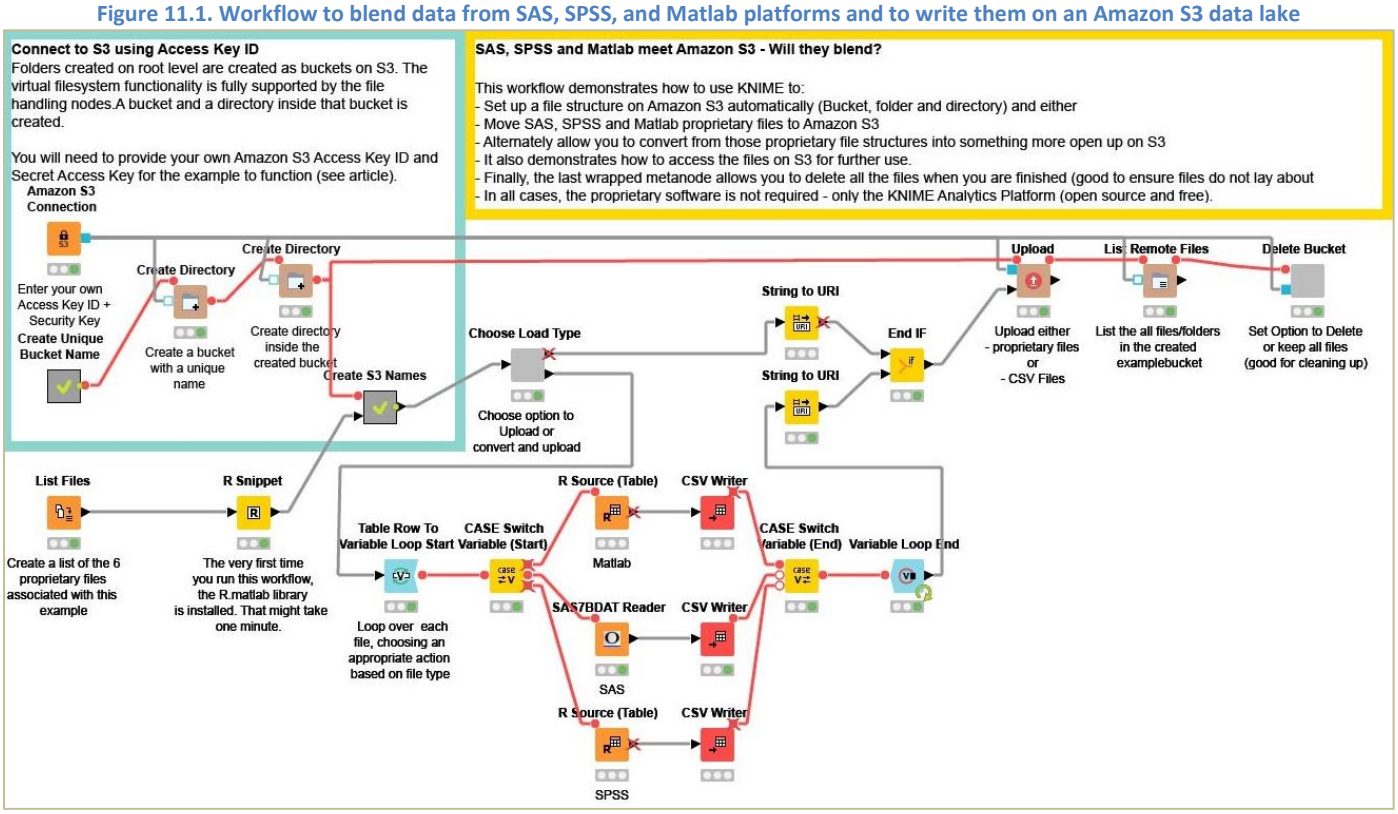
By the way, to be totally inclusive, KNIME also supports Microsoft Azure Blob Storage, and they also offer a free trial account. So sign up for both and compare them. For purposes of keeping this blog short, I will restrict myself to S3.

R is used to read the SPSS and MATLAB files. If you do not have the KNIME R extensions, you will want to install them via the File/Install pulldown within KNIME. By the way, the MATLAB package is not automatically included, but the workflow takes care of this one-time installation for you.

The Workflow

An overall view of the workflow is here:

Figure 11.1. Workflow to blend data from SAS, SPSS, and Matlab platforms and to write them on an Amazon S3 data lake



In the upper left hand corner, I first enter my Amazon S3 credentials into the Amazon S3 Connection node. This establishes a valid connection to S3. By default, there is just "space" on S3 and I need to define what is known as a Bucket and then establish a Directory within that Bucket for all my files. I do that in the upper left hand corner by randomly generating a name for the Bucket. This is helpful as I may do a lot of testing and playing with this workflow, and I want to see the various flavors of files that I produce. For a production application, this section would be replaced with appropriate established String values.

I then want to find the SAS (.SASB7DAT), SPSS (.SAV) and Matlab (.MAT) files. In this example workflow, I've provided a directory with a small variety of each file type actually contained within the workflow location. That List Files node creates that list for me. In preparation for uploading, I need to convert local PC filenames into more generic URI names (which are recognized by S3) and that is what I do in the small metanode named "Create S3 Names".

I then have a choice to make. I can either move the proprietary files directly onto S3, or I can first convert each of those files into an open format before moving them to S3. I do that with the small wrapped metanode named "Choose Load Type". Simply by right clicking that node, I get a specific configuration box (made with Quickforms) that allows me to choose.

If I choose "Proprietary", then the top branch is chosen, the appropriate files name are used and then KNIME does a bulk-load of the files into my Amazon S3 directory. Extremely easy and efficient but, at the end of the day, still in that proprietary form.

If I choose "Open (CSV Format)", I first read each of the proprietary files with the appropriate KNIME node. In this case, we have a native node for SAS7BDAT files and two R nodes that use the appropriate R package to read SPSS and MATLAB files. I then use CSV Writer nodes to write each file. Note the LOOP and CASE SWITCH construct, which takes each row and - based on the file type - automatically uses the appropriate READ node.

When the conversion is finished, all those now Open Format CSV files are bulk loaded into S3. At the end of the flow, I do a simple LIST Directory to show which files are there.

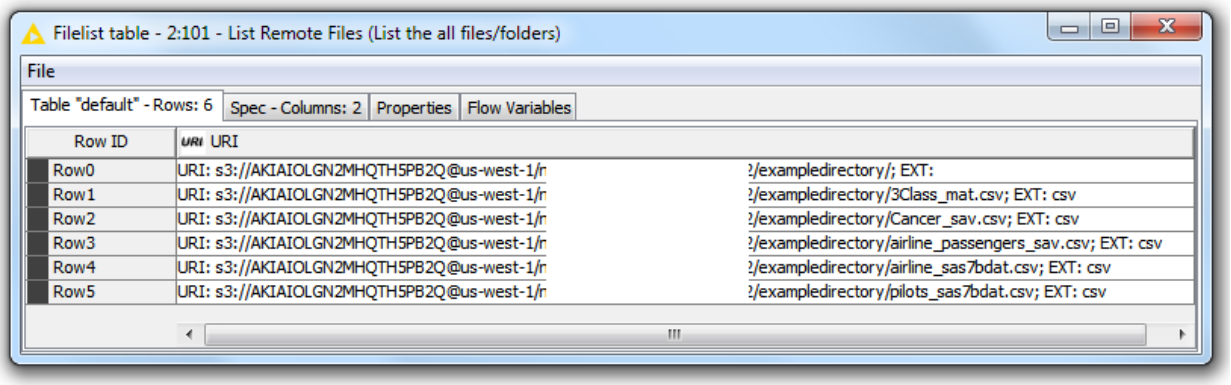
As a very last step, I have a small wrapped metanode that allows me to choose whether to delete the Bucket I created - very handy when cleaning up after my experiments.

NOTE. If you end your KNIME session and want to come back to this example, you will each time need to reset the workflow then rerun (so that you get a NEW S3 Bucket and Directory each time).

The Results

Will they blend? Absolutely. In the figure below, you can clearly see the uploaded .csv files in the list of remote files from the cloud.

Figure 11.2. The blended data on the Amazon data lake. For obvious reasons, the Amazon server URL has been covered.



It is actually fun to play with this example KNIME workflow. You can easily point to your own files or even change the File Readers to your favorite other type of data source (proprietary and old or otherwise). What I find extremely powerful about this blending workflow: you can also change the output data type. Do you fancy NoSQL, Redshift, XML, Excel or any other format? Simply change to the appropriate Writer (either native, ODBC based or R based). And if you want to check out Microsoft AZURE, you can easily modify this workflow by changing the Amazon S3 Connection node to an Azure Blob Store Connection node, with the appropriate parameters, and the workflow should work on Azure as well.

Although these sample files are small, this workflow will work for extremely large files as well since - after all - that is what the cloud is for.

Indeed, in KNIME Analytics Platform you can create streamed wrapped metanodes that subset and manipulate your data "on the fly" creating fantastic opportunities to efficiently use KNIME on those ever growing Data Lakes within the cloud.

Note. If you use a supported **streaming** data types, such as CSV or KNIME Table, you do not need to bulk-download the S3 file but can transfer the data via streaming. For examples, search the KNIME Examples server with the keyword S3 or Azure.

This workflow is available, without the API keys (of course), on the KNIME EXAMPLES server under *01_Data_Access/06_ZIP_and_Remote_Files/05_SAS_SPSS_MATLAB_meet_S3.*

12. Kindle epub meets Image JPEG. Will KNIME make peace between the Capulets and the Montagues?

Author: Heather Fyson and Kilian Thiel, KNIME AG

Workflow in: EXAMPLES/*08_Other_Analytics_Types/01_Text_Processing/18_epub_JPEG_Romeo_Juliet*

Posted on: March 13, 2017

The Challenge

"*A plague o' both your houses! They have made worms' meat of me!*" said Mercutio in Shakespeare's "Romeo and Juliet" – in which tragedy results from the characters' inability to communicate effectively. It is worsened by the fact that Romeo and Juliet each come from the feuding "two households": Romeo a Montague and Juliet, a Capulet.

For this blog article, we decided to take a look at the interaction between the characters in the play by analyzing the script – an epub file – to see just who talks to who. Are the Montagues and Capulets really divided families? Do they really not communicate? To make the results easier to read, we decided to visualize the network as a graph, with each node in the graph representing a character in the play and showing an image of the particular character.

The "Romeo and Juliet" e-book is downloadable for free in a number of formats from the Gutenberg Project web site. For this experiment, we downloaded the epub file. epub is an e-book file format used in many e-reading devices, such as Amazon Kindle for example (for more information about the epub format, check https://en.wikipedia.org/wiki/EPUB).

The images for the characters of the Romeo and Juliet play have been kindly made available by Stadttheater Konstanz in a JPEG format from a live show. JPEG is a commonly used format to store images (for more information about the JPEG format, check https://en.wikipedia.org/wiki/EPUB).

Unlike the Montague and the Capulet families – will epub and JPEG files blend?

Topic. Analyzing the graph structure of the dialogs in Shakespeare's tragedy "*Romeo and Juliet*"

Challenge. Blending epub and JPEG files and combining text mining and network visualization

Access Mode. epub parser and JPEG reader

The Experiment

Reading and Processing the Text in the epub file

1. The Tika Parser node reads the "Romeo and Juliet" epub file, downloaded from the Gutenberg Project site. This node integrates the Apache Tika Parser library into KNIME Analytics Platform and can therefore read a very large number of file formats, such as epub, pdf, docx, eml, zip, odp, ppt, and many, many more. The

output of the Tika Parser node is a number of String cells containing e-book information, such as title, author, content, etc...

2. The "content" column is the column with the full e-book text. It is converted from String into Document type using the Strings to Document node in the wrapped meta-node, "Tag Characters". The Strings to Document node is part of the KNIME Text Processing extension, which needs to be installed to use the node.

3. In this experiment we are not interested in what the characters say but in how they interact with each other, e.g. in how often they talk to each other. The aim is to identify the speaker of each paragraph and tag him or her accordingly. Identifying and tagging characters in the text is the job of the Wildcard Tagger node, still inside the wrapped meta-node, "Tag Characters". The list of the main characters in the play from the Table Creator node represents the tagging dictionary; we remove all other words to keep only the character names.

4. Now, we count the term frequencies and the co-occurrences of all characters using the TF and Term Co-occurrence Counter node. Co-occurrences are normalized pair-wise by creating a pair ID for each unique co-occurrence. Normalization is necessary to count the pairs independently on their occurrence order, e.g. A-B and B-A. This is all handled inside the wrapped meta-node, "Mentions & Interactions".

5. We now set about building the network of interaction among the characters. For each term (=character), the Object Inserter node creates a point in the network. This node is provided by the Network Mining extension. The Object Inserter node also creates an edge between the nodes for each pair of characters. The edges are weighted depending on the number of dialog-based co-occurrences. The term-frequencies of the characters are also inserted into the network as features. This all takes place in the wrapped meta-node, "Build Network". The output is a network with a node for each character and an edge between characters proportionally thick to the number of interactions between the two characters.

Loading the JPEG image files

1. JPEG images of the play's characters are stored locally on the hard disk of the machine. The URL to the image file for each character is manually inserted in a Table Creator node together with the reference character. The Table Creator node allows the user to manually input data in an Excel-like fashion.

2. The list of image URLs feeds the wrapped meta-node, "Load Images". Inside this wrapped node, an Image Reader (Table) node reads all of the JPEG image files in the list. The Image Reader (Table) node is part of the KNIME Image Processing community extension, which needs to be installed for the node to work.

3. The Image Reader (Table) node creates ImagePlus type cells, which have to be converted to regular KNIME image cells to be used for visualization in the network later on. The ImgPlus to PNG Images node, in the Load Images wrapped meta-node, takes care of this task. The output of the meta-node is a table containing the list of character names and the corresponding PNG image.

Blending Text and Images in the Dialog Graph

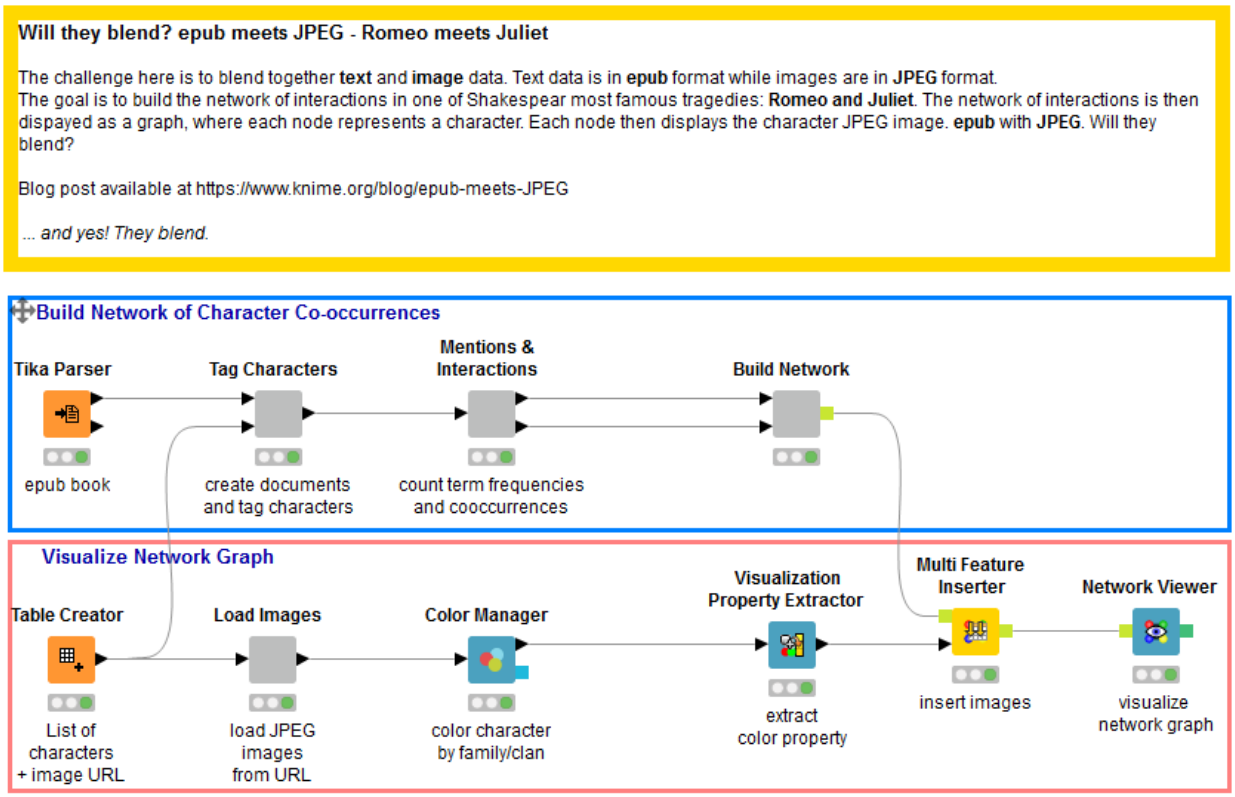
1. It's time to blend! The character interaction network and the character images can now be blended. This is done by the Multi Feature Inserter node. Each character in the play is assigned a color based on the family affiliation. PNG images and colors are subsequently inserted into the network as features.

2. The Network Viewer node finally builds a view of the network, where each node is rendered by the character image, sized by the term (=character) frequency, and border-colored by the family assignment.

The final workflow shown in figure 12.1 is available on the KNIME EXAMPLES server under *08_Other_Analytics_Types/01_Text_Processing/18_epub_JPEG_Romeo_Juliet*

The final graph showing the interaction degree between the different characters of the play is depicted in figure 12.2.

Figure 12.1. This workflow successfully blends a Kindle epub file of the play "Romeo and Juliet" with the JPEG images of the play's characters in a live show in a dialog graph.



The Results

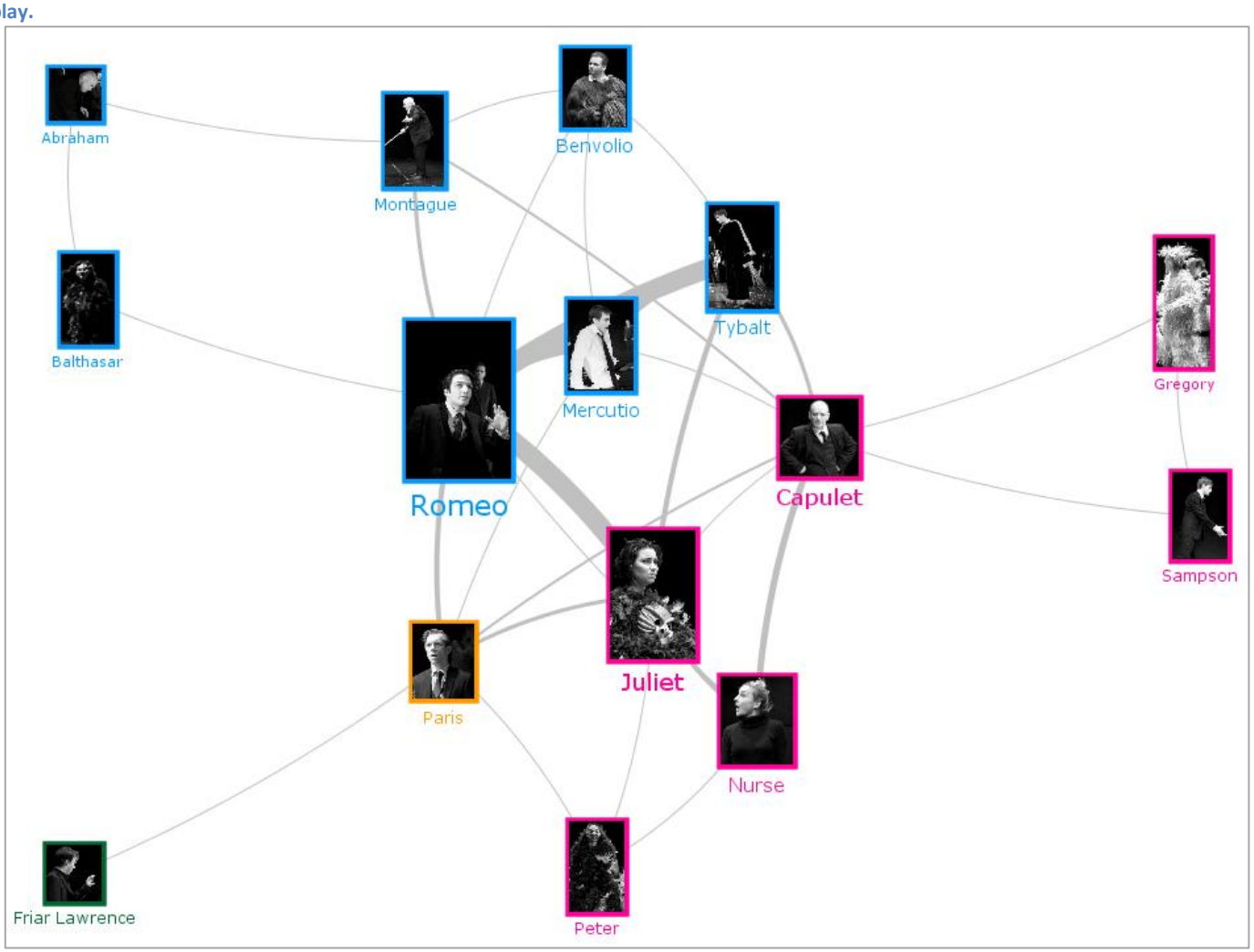
Yes, they blend!

The graph that comes out of this experiment shows the clear separation between the two families quite impressively. All Montagues in green are on one side; all Capulets in red on the other. The two families only interact with each other through a small number of characters and, not surprisingly, most of the interaction that does take place between the families is between Romeo and Juliet. The separation is so neat that we are tempted to think that Shakespeare used a graph himself to deploy the tragedy dialogs!

In this experiment, we've managed to create and visualize the network of interaction between all characters from "Romeo and Juliet", by parsing the epub text document, reading the JPEG images for all characters, and blending the results into the network.

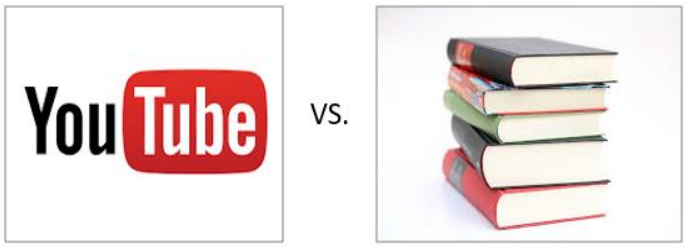
So, even for this experiment, involving epub and JPEG files, text mining and network visualization, we can conclude that … yes, they blend!

Figure 12.2. Interaction network of characters from "Romeo and Juliet". The border color of the nodes indicates the family assignment and the node size reflects the term-frequency of the character in the epub document. Edge thickness between two characters reflects the term-frequency of the character in the epub document. Edge thickness between two characters reflects their interaction degree throughout the play.



13. YouTube Metadata meet WebLog Files. What will it be Tonight – a Movie or a Book?

Author: Rosaria Silipo, KNIME AG
Workflow in: *EXAMPLES/01_Data_Access/07_WebLog_Files/01_Example_for_Apache_Logfile_Analysis*
Posted on: March 27, 2017



The Challenge

Thank God it's Friday! And with Friday, some free time! What shall we do? Watch a movie or read a book? What do the other KNIME users do? Let's check!

When it comes to KNIME users, the major video source is YouTube; the major reading source is the KNIME blog. So, do KNIME users prefer to watch videos or read blog posts? In this experiment we extract the number of views for both sources and compare them.

YouTube offers an access REST API service as part of the Google API. As for all Google APIs, you do not need a full account if all you want to do is search; a key API is enough. You can request your own key API directly on the Google API Console. Remember to enable the key for the YouTube API services. The available services and the procedure to get a key API are described in these 2 introductory links:
https://developers.google.com/apis-explorer/?hl=en_US#p/youtube/v3/
https://developers.google.com/youtube/v3/getting-started#before-you-start

On YouTube the KNIME TV channel hosts more than 100 tutorial videos. However, on YouTube you can also find a number of other videos about KNIME Analytics Platform posted by community members. For the KNIME users who prefer to watch videos, it could be interesting to know which videos are the most popular, in terms of number of views of course.

The KNIME blog has been around for a few years now and hosts weekly or biweekly content on tips and tricks for KNIME users. Here too, it would be interesting to know which blog posts are the most popular ones among the KNIME users who prefer to read – also in terms of number of views! The numbers for the blog posts can be extracted from the weblog file of the KNIME web site.

YouTube with REST API access on one side and blog page with weblog file on the other side. Will they blend?

Topic. Popularity (i.e. number of views) of blog posts and YouTube videos.

Challenge. Extract metadata from YouTube videos and metadata from KNIME blog posts.

Access Mode. WebLog Reader and REST service.

The Experiment

Accessing YouTube REST API

We are using three YouTube REST API services:

- The video search service, named "***search***":
  https://www.googleapis.com/youtube/v3/search?q=KNIME&part=id&maxResults=50&videoDuration=any&key=<your-key-API>
  Here we search for videos that are tagged "KNIME" (q=KNIME), allowing a maximum of 50 videos in the response list (maxResults=50).

- The service for the video details, named "***videos***":
  https://www.googleapis.com/youtube/v3/videos?id=<videoID>&part=snippet,statistics,contentDetails&key=<your-key-API>
  We pass the video IDs we get from the "search" service (id=<videoID>). In return we obtain details of the video, such as duration, permission flags, and statistics, in terms of total number of views, likes, and other video related actions.

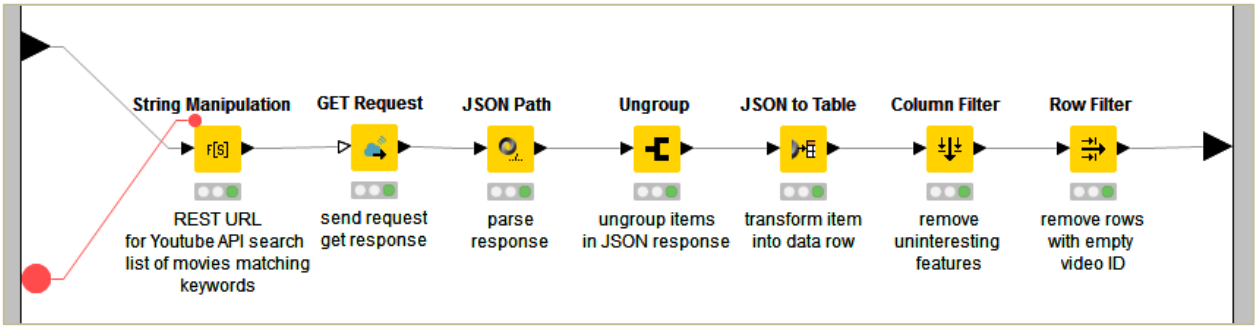
- The service retrieving the comments to the video, named "***commentThreads***":
  https://www.googleapis.com/youtube/v3/commentThreads?videoId=<videoID>&part=snippet,id&key=<your-key-API>
  Here we pass the video IDs we get from the "search" service (id=<videoID>). In return we obtain all comments posted for that video, including the comment text, author ID, and posting date.

A variation of the same metanode, with name starting with "YouTube API", is used to invoke all three YouTube REST API services. All metanodes have the same structure.

  - First the REST query is built as a String, as described above, through a String Manipulation node;
  - then the query is sent to the REST server through a GET Request node;
  - the list of videos or comments or details is extracted from the REST response with a JSON Path node;
  - the same list is ungrouped to place each JSON-structured item in a table row;
  - finally the interesting values are extracted from each JSON-structured item.

See figure 13.1 for the content of a "YouTube API…" metanode.

Figure 13.1. Sub-workflow in metanode to access YouTube REST API.



The upper branch of the final workflow has been built around such metanodes to access the YouTube REST API and to extract the videos related to the given keywords, their details, and the attached comments.

1. The YouTube branch of the workflow starts by creating the keyword for the search and passing the key API for the REST service to be run. Remember, you can get the key API at https://developers.google.com/youtube/v3/getting-started#before-you-start. For both tasks we use a String Input Quickform node. The 2 String Input nodes are contained in the first wrapped node, named "keywords and key API".

2. We now send the request to the first YouTube API REST service, the one named "search", with the keyword "KNIME" defined at the previous step. At the output of the dedicated metanode we get the list of 50 videos tagged "KNIME".

3. Then for each result video we send the request for statistics and comments to the YouTube services named "videos" and "commentThreads" respectively and are returned a list of properties and comments for each video.

4. *The Comments*

a. We now subject the comments to some text processing! Text processing includes language recognition, classical text clean up, bag of word creation, and frequency calculation - all contained in the metanode named "Text-Preprocessing".

> **Note**. Language recognition is performed by means of the **Tika Language Detector node**. Given a text, this node produces a language hypothesis and a confidence measure. We take only comments in English with confidence above 0.8.

The plurality of languages shows how widely KNIME is used around the world. However, we limited ourselves to English just for comprehension reasons.

b. Finally, the terms and frequencies of the video comments end up in a word cloud, built by means of the Tag Cloud node.

c. The word cloud image is then exported into the report connected to this workflow.

5. *The Video Statistics*

a. The video statistics in terms of view count, like count, and similar, are sorted by view count in descending order and the top 10 rows are selected; we extract the top most viewed videos on YouTube, tagged with the word "KNIME".

b. Next, a bar chart is built showing the view count for the top most viewed KNIME related YouTube videos.

Parsing the WebLog File

The KNIME blog is part of the general KNIME web site. All access details about the KNIME blog are available in the weblog file from the KNIME web site. Among those details, the access data for each blog post are available in the weblog file.

The lower branch of this experiment's workflow focuses on reading, parsing, and extracting information about the KNIME blog posts from the site weblog file.

1. KNIME Analytics Platform offers a dedicated node to read and parse weblog files: the Web Log Reader node. The lower workflow branch starts with a Weblog Reader node. In the configuration window of the Web Log Reader node, a few useful parsing parameters can be defined. Besides the file path, which is necessary of course, it is also possible to set the date format, time interval, line format, and locale you want to use. The button "Analyze Log" produces a rough analysis of the log file structure and the required settings. The Web Log Reader node can be found in the wrapped metanode named "Read Log Files".

2. The next process after importing the content of the weblog file involves parsing and filtering to make sure we separate all the blog post details from all of the other information.

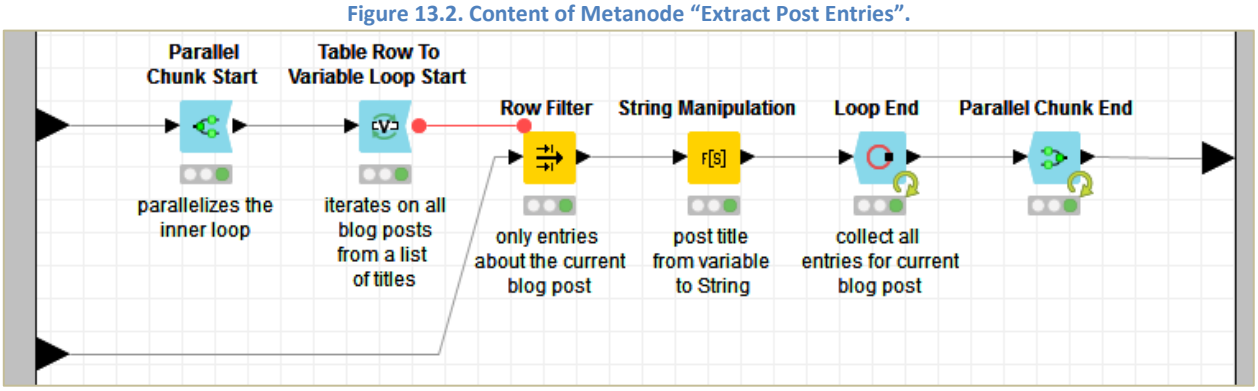
a. The list of titles of the blog posts published so far is made available, among other things, by the metanode "Get Blog Patterns".

b. This title list feeds a TableRow to Variable Loop Starts node, in metanode "Extract Post Entries". This node iterates through all post titles, one by one, and the following Row Filter node extracts the weblog content related to the post title in the current iteration. The loop collects all weblog entries for each one of the blog post titles.

> **Note**. Actually, as you can see from figure 13.2, the metanode "Extract Post Entries" exhibits 2 loops. The second loop loops around the blog post titles, one by one, as described above. The first loop, the **parallel chunk loop**, is just a utility loop, used to parallelize and speed up its loop body.

c. The last metanode, called “Aggregate & Sort”, counts the number of views for each blog post based on the number of related entries, and extracts the top 10 most viewed blog posts since the first publication.

Figure 13.2. Content of Metanode “Extract Post Entries”.



Data Blending and Final Report

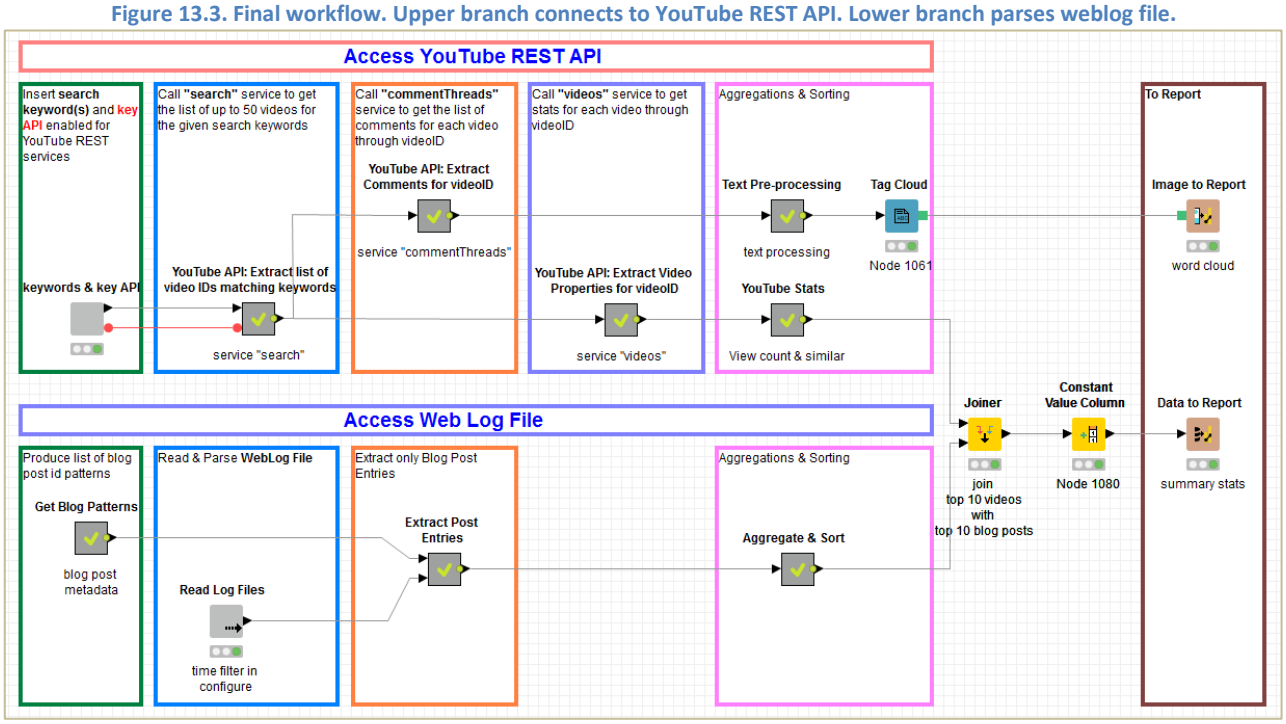
The upper branch has the data from YouTube, aggregated to show the number of views for the top 10 most viewed KNIME tagged videos.

The lower branch has the data from the weblog file, aggregated to show the number of views for the top 10 most read KNIME blog posts.

The two datasets are joined through a Joiner node and sent to the workflow report project.

The final workflow is shown in figure 13.3.

Figure 13.3. Final workflow. Upper branch connects to YouTube REST API. Lower branch parses weblog file.



For privacy reasons, we could not make this workflow available as it is on the KNIME EXAMPLES server.

However, you can find an example workflow for the WebLog Reader node on the EXAMPLES server under *01_Data_Access/07_WebLog_Files/01_Example_for_Apache_Logfile_Analysis.*

The upper part of this workflow can be found on the EXAMPLES server under:

01_Data_Access/05_REST_Web_Services/03_Access_YouTube_REST_API, without the key API information. You will need to get your own key API, enabled for the YouTube REST API, from the Google API Console.

The Results

The report created from the workflow is exported as pdf document in figure 13.4. On page 2 and 3, you will see two bar charts reporting the number of views for the top 10 most viewed YouTube videos and the top 10 most read blog posts, respectively.

Here are the top 10 YouTube videos:

- "An Introduction to KNIME" by KNIME TV
- "Learning KNIME through the EXAMPLES Server – Webinar" by KNIME TV
- "Introduction to the KNIME Data Mining System (Tutorial)" by Predictive Analytics
- "Text Mining Webinar" by KNIME TV
- "KNIME Workflow Introduction" by KNIME TV
- "Building a basic Model for Churn Prediction with KNIME" by KNIME TV
- "KNIME: A tool for Data Mining" by Sania Habib
- "Analyzing the Web from Start to Finish - Knowledge Extraction using KNIME - Bernd Wiswedel - #1" by Zürich Machine Learning and Data Science Meetup
- "Recordings of "Database Access with KNIME" Webinar" by KNIME TV
- "Tutorial about the new R Interactive nodes in KNIME" by KNIME TV

Here are the top 10 KNIME blog posts:

- "Sentiment Analysis" by K. Thiel
- "7 Techniques for Data Dimensionality Reduction" by R. Silipo
- "7 Things to do after installing KNIME Data Analytics Platform" by R. Silipo
- "Semantic Enrichment of Textual Documents" by J. Grossmann
- "From D3 example to interactive KNIME view in 10 minutes" by C. Albrecht
- "To Code or not to code – Is that the question?" by M. Berthold
- "Anomaly Detection in Predictive Maintenance with Time Series Analysis" by R. Silipo
- "Author ranking and conference crawling for gene editing technology CRISPR-Cas" by F. Dullweber
- "Market Basket Analysis and Recommendation Engines" by R. Silipo
- "The KNIME Server REST API" by J. Fuller

In both lists, we find abundant material for KNIME beginners. The readers of the KNIME blog posts seem to enjoy a post or two about some specific topics, such as gene editing technology or the KNIME Server REST API, but in general they also use the KNIME blog posts to learn new how-to procedures.

The last page of the pdf report document contains the word cloud of comments in English on the YouTube videos. We would like to take the opportunity in this blog post to thank the YouTube watchers for their kind words of appreciation.

In general, we have more views on the YouTube videos than on the blog post. It is also true that the KNIME TV channel started4 years ao, while the KNIME blog only 2 years ago. Since we have not set any time limits, the number of views are counted from each video/post uploading date. So, it is hard to conclude the proportion of KNIME users who prefer watching videos over reading posts.

Summarizing, in this experiment we tried to blend data from a weblog file with metadata from YouTube videos. Again, the most important conclusion is: Yes, they blend!

What about you? Which kind of KNIME user are you? A video watcher or a blog post reader?

News! If you are a video watcher type, we have a new treat for you. A full [e-learning course] to introduce new and old users to the secrets of KNIME Analytics Platform is now available on the KNIME web site.

This e-learning course consists of a series of short units. Each unit involves a brief YouTube video and often a dedicated exercise to allow data scientists to learn about KNIME Analytics Platform and ETL operations at their own pace.

So, if you are looking for something fun to do this evening, you can start exploring this new e-learning course. The [intro page has a lot of information].

Figure 13.4. Final Report as PDF document. At page 2 and 3, we find 2 bar charts with the number of views for the top most watched KNIME tagged YouTube videos and with the number of views of the most read KNIME blog posts respectively. In the last page, the flattering word cloud from the watchers comments on the YouTube videos.

14. Blending Databases. A Database Jam Session.

Author: Rosaria Silipo, KNIME AG
Workflow in: *EXAMPLES/01_Data_Access/02_Databases/08_Database_Jam_Session*
Posted on: April 10, 2017

The Challenge

Today we will push the limits by attempting to blend data from not just 2 or 3, but 6 databases!

These 6 SQL and noSQL databases are among the top 10 most used databases, as listed in most database comparative web sites (see DB-Engines Ranking, The 10 most popular DB Engines …, Top 5 best databases). Whatever database you are using in your current data science project, there is a very high probability that it will be in our list today. So, keep reading!

What kind of use case is going to need so many databases? Well, actually it's not an uncommon situation. For this experiment, we borrowed the use case and data sets used in the Basic and Advanced Course on KNIME Analytics Platform. In this use case, a company wants to use past customer data (behavioral, contractual, etc…) to find out which customers are more likely to buy a second product. This use case includes 6 datasets, all related to the same pool of customers.

1. **Customer Demographics (Oracle)**. This dataset includes age, gender, and all other classic demographic information about customers, straight from your CRM system. Each customer is identified by a unique customer key. One of the features in this dataset is named "Target" and describes whether the customer, when invited, bought an additional product. 1 = he/she bought the product; 0 = he/she did not buy the product. This dataset has been stored in an Oracle database.

2. **Customer Sentiment (MS SQL Server)**. Customer sentiment about the company has been evaluated with some customer experience software and reported in this dataset. Each customer key is paired with customer appreciation, which ranges on a scale from 1 to 5. This dataset is stored on a Microsoft SQL Server database.

3. **Sentiment Mapping (MariaDB)**. This dataset here contains the full mapping between the appreciation ranking numbers in dataset # 2 and their word descriptions. 1 means "very negative", 5 means "very positive", and 2, 3, and 4 cover all nuances in between. For this dataset we have chosen storage relatively new and very popular software: a MariaDB database.

4. **Web Activity from the company's previous web tracking system (MySQL)**. A summary index of customer activity on the company web site used to be stored in this dataset. The web tracking system associated with this dataset has been declared obsolete and phased out a few weeks ago. This dataset still exists, but is not being updated anymore. A MySQL database was used to store these data.

5. **Web Activity from the company's new web tracking system (MongoDB)**. A few weeks ago the original web tracking system was replaced by a newer system. This new system still tracks customers' web activity on the company web site and still produces a web activity index for each customer. To store the results, this system relies on a new ***noSQL*** database: MongoDB. No migration of the old web activity indices has been attempted, because migrations are costly in terms of money, time, and resources. The idea is that eventually the new system will cover all customers and the old system will be completely abandoned. Till then, though, indices from the new system and indices from the old system will have to be merged together at execution time.

6. **Customer Products (PostgreSQL).** For this experiment, only customers who already bought one product are considered. This dataset contains the one product owned by each customer and it is stored in a PostgreSQL database.

The goal of this experiment is to retrieve the data from all of these data sources, blend them together, and train a model to predict the likelihood of a customer buying a second product.

The blending challenge of this experiment is indeed an extensive one. We want to collect data from all of the following databases: ***MySQL, MongoDB, Oracle, MariaDB, MS SQL Server, and PostgreSQL***. Six databases in total: five relational databases and one noSQL database.

Will they all blend?

Topic. Next Best Offer (NBO). Predict likelihood of customer to buy a second product.

Challenge. Blend together data from six commonly used SQL and noSQL databases.

Access Mode. Dedicated connector nodes or generic connector node with JBDC driver.

The Experiment

Let's start by connecting to all of these databases and retrieving the data we are interested in.

Relational Databases

Data retrieval from all relational SQL-powered databases follows a single pattern:

a. ***Define Credentials***.

- Credentials can be defined at the workflow level (right-click the workflow in the KNIME Explorer panel and select Workflow Credentials). Credentials provided this way are encrypted.
- Alternatively, credentials can be defined in the workflow using a Credentials Input node. The Credentials Input node protects the username and password with an encryption scheme.
- Credentials can also be provided explicitly as username and password in the configuration window of the connector node. A word of caution here. This solution offers no encryption unless a Master Key is defined in the Preferences page.

b. ***Connect to Database***.

With the available credentials we can now connect to the database. To do that, we will use a connector node. There are two types of connector nodes in KNIME Analytics Platform.

- ***Dedicated connector nodes***. Some databases, with redistributable JDBC driver files, have dedicated connector nodes hosted in the Node Repository panel. Of our 6 databases, ***MySQL, PostgreSQL, and SQL Server*** enjoy the privilege of dedicated connector nodes. Dedicated connector nodes encapsulate the JDBC driver file and other settings for that particular database, making the configuration window leaner and clearer.
- ***Generic connector node***. If a dedicated connector node is not available for a given database, we can resort to the generic Database Connector node. In this case, the JDBC driver file has to be uploaded to KNIME Analytics Platform via the *Preferences -> KNIME -> Database* page. Once the JDBC driver file has been uploaded, it will also appear in the drop-down menu in the configuration window of the Database Connector node. Provided the appropriate JDBC driver is selected and the database hostname and credentials have been set, the Database Connector node is ready to connect to the

selected database. Since a dedicated connector node was missing, we used the Database Connector node to connect to ***Oracle*** and ***MariaDB*** databases.

c. ***Select Table and Import Data***.

Once a connection to the database has been established, a Database Table Selector node builds the necessary SQL query to extract the required data from the database. A Database Connection Table Reader node then executes the SQL query, effectively importing the data into KNIME Analytics Platform.

It is comforting that this approach - *connect to database, select table, and extract data* – works for all relational databases. It is equally comforting that the Database Connector node can reach out to any database. This means indeed that with this schema and with the right JDBC driver file I can connect to all existing databases, including vintage versions or those of rare vendors.

NoSQL Databases

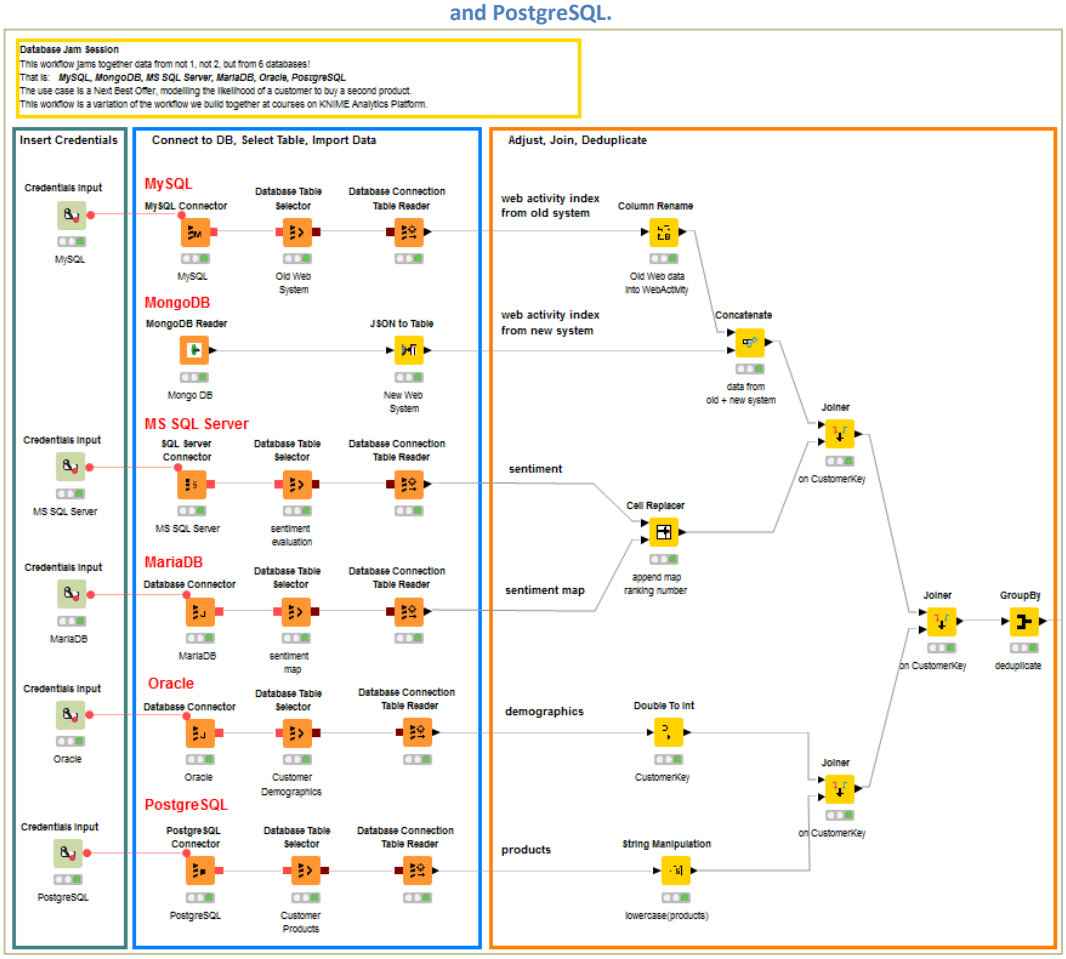
Connecting to a NoSQL database, such as MongoDB, follows a different node sequence pattern.

In KNIME Labs, a MongoDB sub-category hosts a few nodes that allow you to perform basic database operations on a MongoDB database. In particular, the MongoDB Reader node connects to a MongoDB database and extracts data according to the query defined in its configuration window.

Credentials here are required within the configuration window and it is not possible to provide them via the Credentials Input node or the Workflow Credentials option.

Data retrieved from a MongoDB database are encapsulated in a JSON structure. No problem. This is nothing that the JSON to Table node cannot handle. At the output of the JSON to Table node, the data retrieved from the MongoDB database are then made available for the next KNIME nodes.

Figure 14.1. This is the part of the workflow that blends data from six different databases: MySQL, MongoDB, SQL Server, Oracle, MariaDB, and PostgreSQL.



Train a Predictive Model

Most of our six datasets contain information about all of the customers. Only the web activity datasets alone do not cover all customers. However, together they do. The old web activity dataset is concatenated with the new web activity dataset. After that, all data coming from all of the different data sources are adjusted, renamed, converted, and joined so that one row represents one customer, where the customer is identified by its unique customer key.

Note. Notice the usage of a GroupBy node to perform a deduplication operation. Indeed, grouping data rows on all features allows for removal of identical rows.

The resulting dataset is then partitioned and used to train a machine learning model. As machine learning model, we chose a random forest with 100 trees and we trained it to predict the value in "Target" column. "Target" is a binary feature representing whether a customer bought a second product. So training the model to predict the value in "Target" means that we are training the model to produce the likelihood of a customer to buy a second product, given all that we already know about her/him.

The model is then applied to the test set and its performance evaluated with a Scorer node. The model accuracy was calculated to be around 77%.

Measuring the Influence of Input Features

A very frequent task in data analytics projects is to determine the influence of the input features on the trained model. There are many solutions to that, which also depend on the kind of predictive model that has been adopted.

A classic solution that works with all predictive algorithms is the backward feature elimination procedure (or its analogous forward feature construction).

Backward Feature Elimination starts with all N input features and progressively removes one to see how this affects the model performance. The input feature whose removal lowers the model's performance the least is left out. This step is repeated until the model's performance is worsened considerably. The subset of input features producing a high accuracy (or a low error) represents the subset of most influential input features. Of course, the definition of high accuracy (or low error) is arbitrary. It could mean the highest accuracy or a high enough accuracy for our purposes.

The metanode, named "Backward Feature Elimination" and available in the Node Repository under *KNIME Labs/Wide Data/Feature Selection*, implements exactly this procedure. The final node in the loop, named "Feature Selection Filter", produces a summary of the model performance, for all steps where the input feature with lowest influence had been removed.

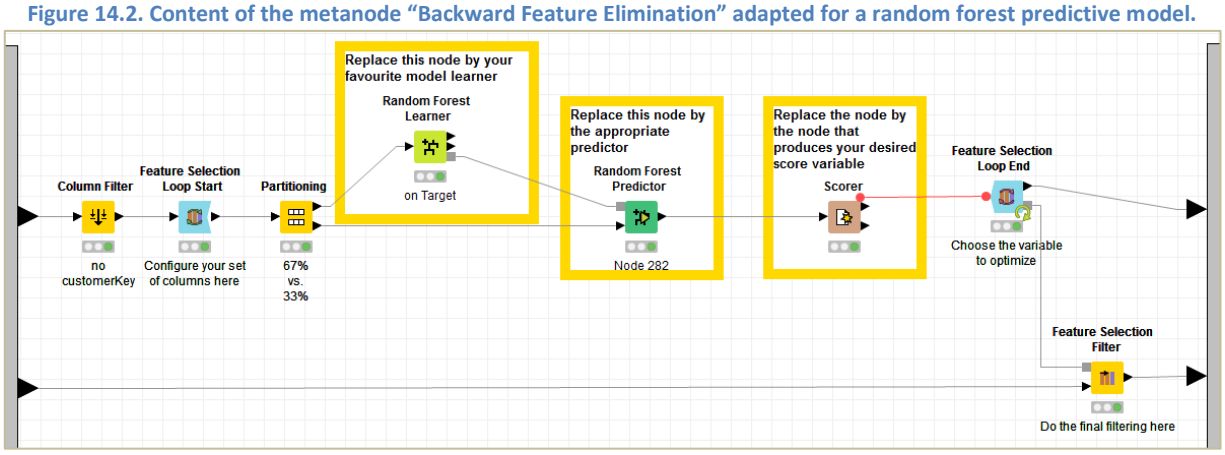
Remember that the Backward Feature Elimination procedure becomes slower with the higher number of input features. It works well with a limited number of input features, but avoid using it to investigate hundreds of them.

In addition, a random forest offers a higher degree of interpretability with respect to other machine learning models. One of the output ports of the Random Forest Learner node provides the number of times an input feature has been the candidate for a split and the number of times it has actually been chosen for the split, for levels 0, 1, and 2 across all trees in the forest. For each input feature, we subsequently defined a heuristic measure of influence, borrowed from the KNIME whitepaper [“Seven Techniques for Data Dimensionality Reduction”](), as:

```
influence index = Sum(# splits) / sum(# candidates)
```

The input features with highest influence indices are the most influential ones on the model performance.

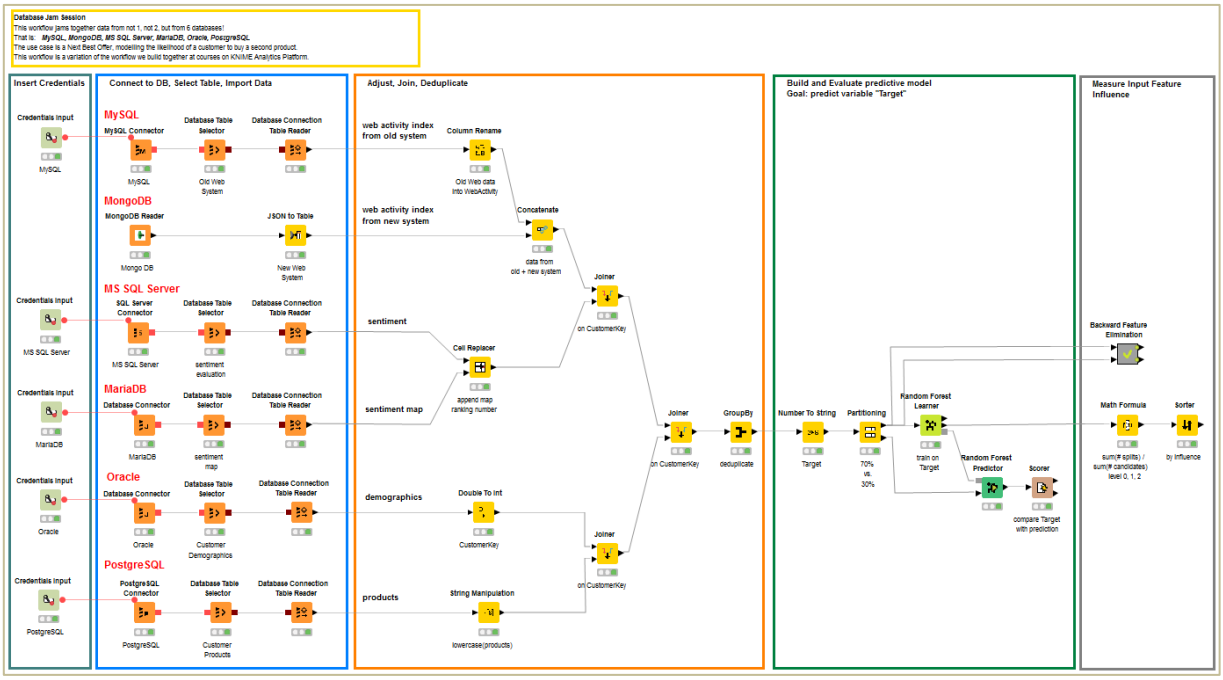
Figure 14.2. Content of the metanode “Backward Feature Elimination” adapted for a random forest predictive model.



The final workflow is shown in figure 14.3 and it is downloadable from the KNIME EXAMPLES server under: *01_Data_Access/02_Databases/08_Database_Jam_Session.*

In figure 14.3 you can see the five parts of our workflow: Credentials Definition, Database Connections and Data Retrieval, Data Blending to reach one single data table, Predictive Model Training, and Influence Measure of Input Features.

Figure 14.3. This workflow blends data from 6 different databases: MySQL, MongoDB, SQL Server, Oracle, MariaDB, and PostgreSQL. The blended dataset is used to train a model to predict customer's likelihood to buy a second product. The last nodes measure input features' influence on the final predictions.



The Results

Yes, data from all of these databases do blend!

In this experiment, we blended data from six, SQL based and noSQL, databases - *Oracle, MongoDB, MySQL, MariaDB, SQL Server, and PostgreSQL* – to reach one single data table summarizing all available information about our customers.

In this same experiment, we also trained a random forest model to predict the likelihood of a customer buying a second product.

Finally, we measured each input feature's influence on the final predictions, using a Backward Feature Elimination procedure and a heuristic influence measure based on the numbers of splits and candidates in the random forest. Results from both procedures are shown in figures 14.4 and 14.5. Both figures show the prominent role of Age and Estimated Yearly Income and the negligible role of Gender, when predicting whether customer will buy a second product.

Figure 14.4. Bar rendering of the influence indices calculated for all input features.

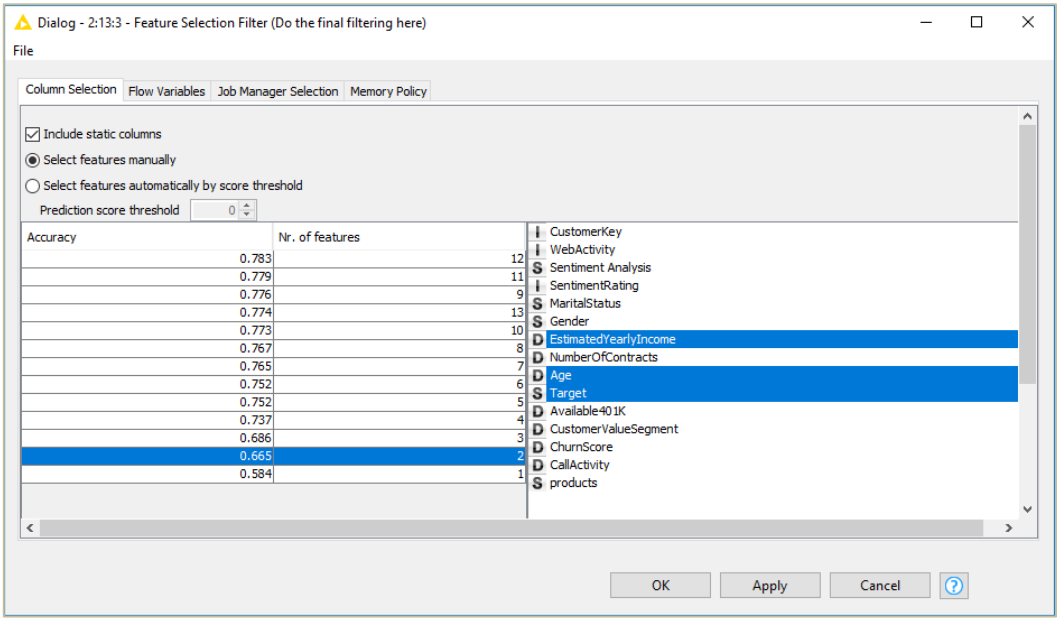
Sorted Table - 2:282 - Sorter (by influence)

File

Table "default" - Rows: 13 | Spec - Columns: 7 | Properties | Flow Variables

Row ID	#splits ...	#splits ...	#splits ...	#candi...	#candi...	#candi...	influence index
Age	20	24	62	25	37	95	
Sentiment An...	13	27	39	20	42	94	
NumberOfCo...	19	26	29	20	59	79	
EstimatedYea...	1	12	59	18	38	101	
products	6	25	35	19	45	90	
WebActivity	7	22	40	28	58	95	
SentimentRating	13	20	26	22	44	107	
CustomerValu...	7	11	28	22	41	82	
ChurnScore	14	15	26	28	48	110	
CallActivity	0	10	24	26	42	92	
MaritalStatus	0	4	17	19	36	80	
Available401K	0	3	9	27	55	96	
Gender	0	1	0	26	55	79	

Figure 14.5. Accuracy for subsets of input features from the configuration window of the Feature Selection Filter node.



This whole predictive and influence analysis was made possible purely because of the data blending operation involving the many different database sources. The main result is therefore another yes! Data can be retrieved from different databases and they all blend!

The data and use case for this post are from the basic and advanced course on KNIME Analytics Platform. The course, naturally, covers much more and goes into far more detail than what we have had the chance to show here.

Note. Just in case, you got intrigued and you want to know more about the courses that KNIME offers, you can refer to the course web page on the KNIME web site. Here you can find the courses schedule and a description of their content. In particular, the slides for the basic and advanced course can now be downloaded for free from https://www.knime.org/course-materials-download-registration-page.

15. Teradata Aster meets KNIME Table. What is that Chest Pain?

Author: Kate Phillips, Teradata
Workflow in: *EXAMPLES/01_Data_Access/02_Databases/09_Teradata_Aster_meets_KNIME_Table*
Posted on: April 24, 2017

The Challenge

Today's challenge is related to the healthcare industry. You know that little pain in the chest you sometimes feel and you do not know whether to run to the hospital or just wait until it goes away? Would it be possible to recognize as early as possible just how serious an indication of heart disease that little pain is?

The goal of this experiment is to build a model to predict whether or not a particular patient with that chest pain has indeed heart disease.

To investigate this topic, we will use open-source data obtained from the University of California Irvine Machine Learning Repository, which can be downloaded from http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/. Of all datasets contained in this repository, we will use the processed Switzerland, Cleveland, and VA data sets and the reprocessed Hungarian data set.

These data were collected from 920 cardiac patients: 725 men and 193 women aged between 28 and 77 years old; 294 from the Hungarian Institute of Cardiology, 123 from the University Hospitals in Zurich and Basel, Switzerland, 200 from the V.A. Medical Center in Long Beach, California, and 303 from the Cleveland Clinic in Ohio.

Each patient is represented through a number of demographic and anamnestic values, angina descriptive fields, and electrocardiographic measures (http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names).

In the dataset each patient condition is classified into 5 levels according to the severity of his/her heart disease. We simplified this classification system by transforming it into a binary class system: 1 means heart disease was diagnosed, 0 means no heart disease was found.

This is not the first time that we are running this experiment. In a not even that remote past, we built a Naïve Bayes KNIME model on the same data to solve the same problem. Today we want to build a logistic regression model and see if we get any improvements on the Naïve Bayes model performance.

Original patient data are stored in a Teradata database. The predictions from the old Naïve Bayes model are stored in a KNIME Table.

Teradata Aster is a proprietary database system that may be in use at your company/organization. It is designed to enable multi-genre advanced data transformation on massive amounts of data. If your company/organization is a Teradata Aster customer, you can obtain the JDBC driver that interfaces with KNIME by contacting your company's/organization's Teradata Aster account executive.

Table format is a KNIME proprietary format to store data efficiently, in terms of size and retrieval speed, and completely -i.e. also including their structure metadata. This leads to smaller local files, faster reading, and minimal configuration settings. In fact, the Table Reader node, which reads such Table files, only needs the file path and

retrieves all other necessary information from the metadata saved in the file itself. Files saved in KNIME Table format carries an extension ".table".

Teradata Aster on one side, KNIME Table formatted file on the other side. The question, as usual, is: Will they blend? Let's find out.

Topic: Predicting heart disease. Is this chest pain innocuous or serious?

Challenge: Blend data from Teradata Aster system with data from a KNIME .table file. Build a predictive model to establish presence or absence of heart disease.

Access Mode: Database Connector node with Teradata JDBC driver to retrieve data from Teradata database. Table Reader node to read KNIME Table formatted files.

The Experiment

Accessing the Teradata Aster database

1. First of all, we needed the appropriate JDBC driver to interface Teradata Aster with KNIME. If your company/organization is a Teradata Aster customer, the noarch-aster-jdbc-driver.jar file can be obtained by contacting your Teradata Aster account executive.
2. Once we downloaded the noarch-aster-jdbc-driver.jar file, we imported it into the list of available JDBC drivers in KNIME Analytics Platform.
    a. Open KNIME Analytics Platform and select File -> Preferences -> KNIME -> Databases -> Add File.
    b. Navigate to the location where you saved the noarch-aster-jdbc-driver.jar file.
    c. Select the .jar file, then click Open -> OK.
3. In a KNIME workflow, we configured a Database Connector node with the Teradata Aster database URL (<protocol> = jdbc:ncluster), the just added JDBC driver (com.asterdata.ncluster.jdbc.core.NClusterJDBCDriver) and the credentials to the same Teradata Aster database.
4. The Database Connector node was then connected directly to a Database Reader node. Since we are quite expert SQL coders, we implemented the data pre-processing into the database Reader node in the form of SQL code. The SQL code selects the [schema].heartdx_prelim table, creates an ID variable called "rownum" (without quotes) and the new binary response variable, named dxpresent, (disease =1, no disease=0), and recodes missing values (represented by -9s and, for some variables, 0s) as true NULL values. The SQL code is shown below:

```
DROP TABLE IF EXISTS [schema].knime_pract;

CREATE TABLE [schema].knime_pract
DISTRIBUTE BY HASH(age)
COMPRESS LOW AS (
      SELECT
            (ROW_NUMBER() OVER (ORDER BY age)) AS rownum,
            age,
              gender,
            chestpaintype,
            CASE
                  WHEN restbps = 0 THEN null
                  ELSE restbps
              END AS restbps,
              CASE
                  WHEN chol = 0 THEN null
```

```
                    ELSE chol
               END AS chol,
               fastbloodsug,
               restecg,
               maxheartrate,
               exindang,
               oldpeak,
               slope,
               numvessels,
               defect,
               dxlevel,
               CASE
                    WHEN dxlevel IN ('1', '2', '3', '4') THEN '1'
                    ELSE '0'
               END AS dxpresent
          FROM [schema].heartdx_prelim
    );

    SELECT * FROM [schema].knime_pract;
```

If you are not an expert SQL coder, you can always use the KNIME in-database processing nodes available in the Node Repository in the Database/Manipulation sub-category.

Building the Predictive Model to recognize possible heart disease

1. At this point, we have imported the data from the Teradata Aster database into our KNIME workflow. More pre-processing, however, was still needed:
    a. Filtering out empty or almost empty columns; that is, columns with too many missing values (please see this article for tips on dimensionality reduction: https://www.knime.org/blog/seven-techniques-for-data-dimensionality-reduction)
    b. On the remaining columns, performing missing value imputation in a Missing Value node by replacing missing numeric values (both long and double) with the median and missing string values with the most frequently occurring value
    c. Partitioning the dataset into training and test set using a Partitioning node (80% vs. 20%)

2. After this pre-processing has been executed, we can build the predictive model. This time we chose a logistic regression model. A Logistic Regression Learner node and a Regression Predictor node were been introduced into the workflow.
    a. Column named dxpresent was used as the target variable in the Logistic Regression Learner node.
    b. Box “Append columns…” was checked in the Regression Predictor node. This last option is necessary to produce the prediction values which will later be fed into an ROC Curve node to compare the 2 models performances.

Reading data from the KNIME .table file

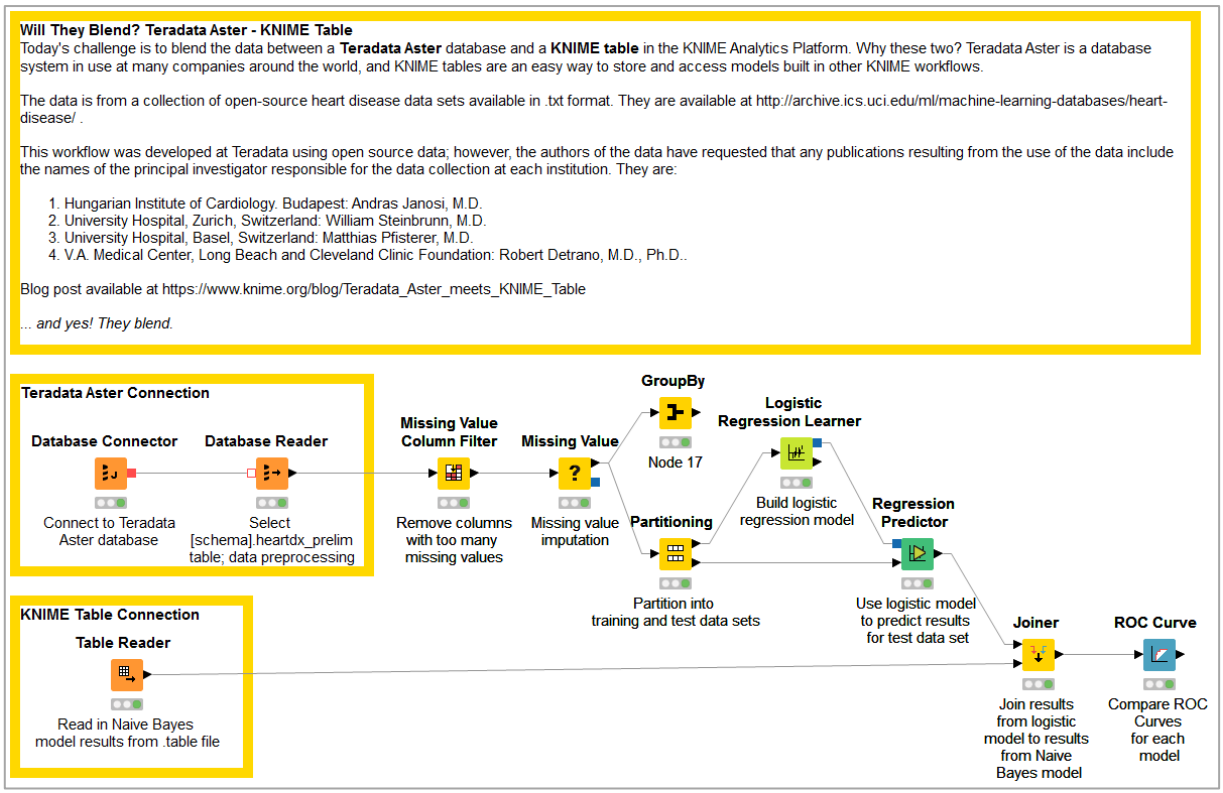
1. Here we just needed to write the Table file path into a Table Reader node. Et voilà we got the predictions previously produced by a Naïve Bayes model.

Note. *The file path is indeed all you need!* All other necessary information about the data structure is stored in the .table file itself.

Blending Predictions

1. After training the logistic regression model, we used a Joiner node to connect its predictions to the older predictions from the Naïve Bayes model.
2. We then connected an ROC Curve node, to display the false positives and true positives, through P(dxpresent=1), for both models.

Figure 15.1. This workflow retrieves data from a Teradata Aster database, builds a predictive model (Logistic Regression) to recognize the presence of a heart disease, blends this model's predictions with the predictions of an older model (Naïve Bayes) stored in a KNIME Table file, and then compares the 2 models performances through an ROC curve.

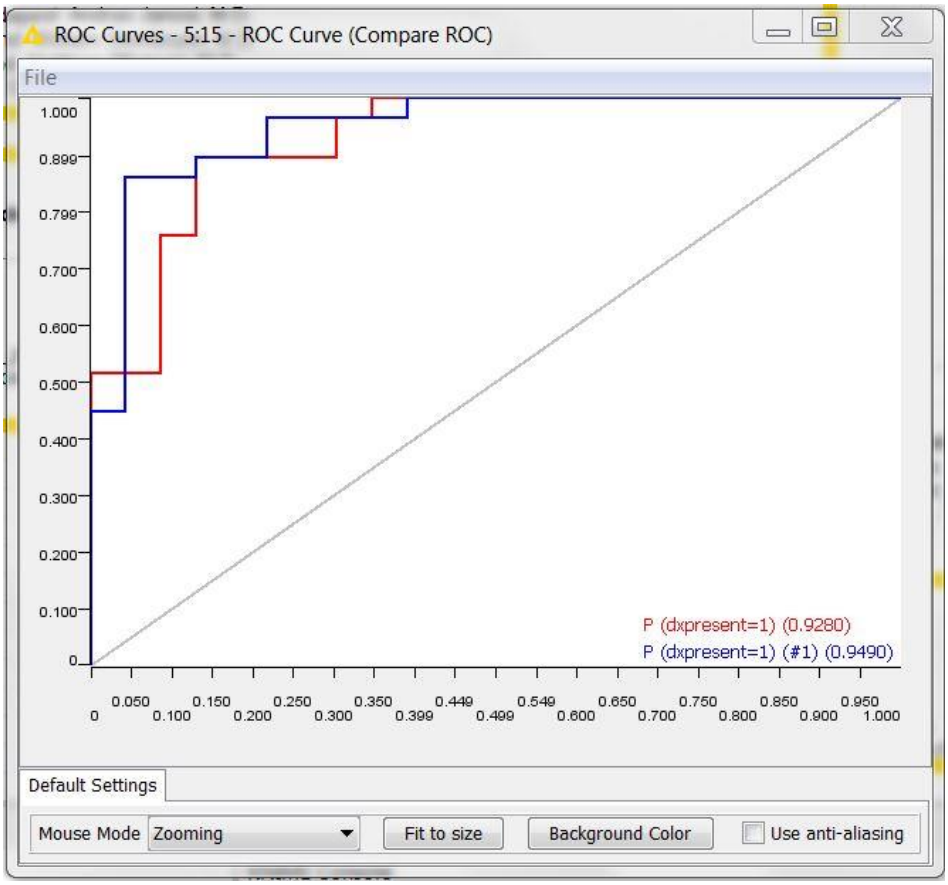


The final workflow is shown in figure 15.1 and it is available for download on the EXAMPLES server under 01_Data_Access/02_Databases/09_Teradata_Aster_meets_KNIME_Table.

The Results

The ROC curves resulting from the workflow are shown in figure 15.2. The red curve refers to the logistic regression, the blue curve to the old Naïve Bayes model. We can see that the Naïve Bayes model, though being older, is not obsolete. In fact, it shows an area under the curve (0.95) comparable to the area under the curve of the newer logistic regression (0.93).

Figure 15.2. ROC curves of the Logistic Regression (in red) and of the old Naive Bayes model (in blue)

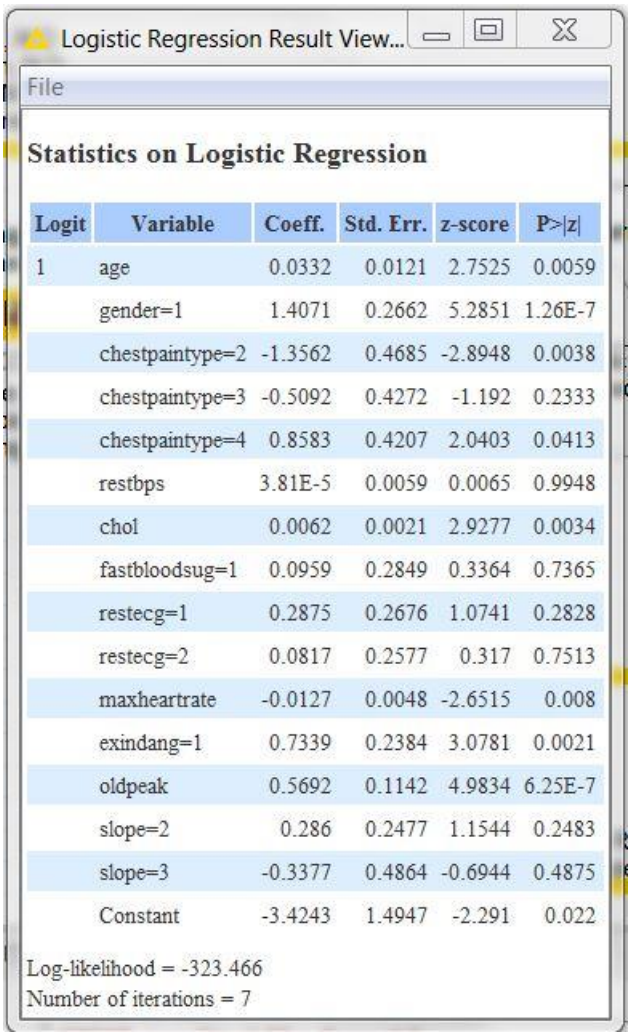


To conclude, let's spend two words on the logistic regression model interpretation. After opening the view of the Logistic Regression node, we get the table in figure 15.3. There, from the coefficient values you can see that gender and chest pain type #2 (atypical angina) are the main drivers for the prediction, although patients with chest pain type #2 are less likely to have heart disease than those with other types of chest pain.

Are men more affected than women by heart disease? Does this describe a general a priori probability or is it just the a priori probability of the data set? It would be interesting here to see how many men with chest pain =2 have heart disease and how many do not. Same for women. We can investigate this with a GroupBy node following the Missing Value node. We configure the GroupBy node to group data rows by gender, chest pain type, and dxpresent; our desired aggregation is the count of the number of rows (on rownum column).

After execution, we find that 60 out of 193 women have chest pain type #2; of the women with this chest pain type, 4 have heart disease while 56 do not. In other words, our data set shows that women with chest pain type #2 have only a 4/60 = 6.7% chance of having heart disease. For the men, we find that 113 out of 725 men have chest pain type #2; of the men with this type of chest pain, 20 have heart disease while 93 do not. According to our data, men with chest pain type #2 have a 20/113 = 17.7% chance of having heart disease.

Figure 15.3. Coefficients of the Linear Regression model to predict presence/absence of heart disease



In this experiment we have successfully blended predictions from a logistic regression model trained on data stored in a Teradata database with older predictions from a Naïve Bayes model stored in a KNIME Table formatted file.

Again, the most important conclusion of this experiment is: Yes, they blend!

The authors of the databases have requested that any publications resulting from the use of the data include the names of the principal investigator responsible for the collection of the data at each institution. They would be:
1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

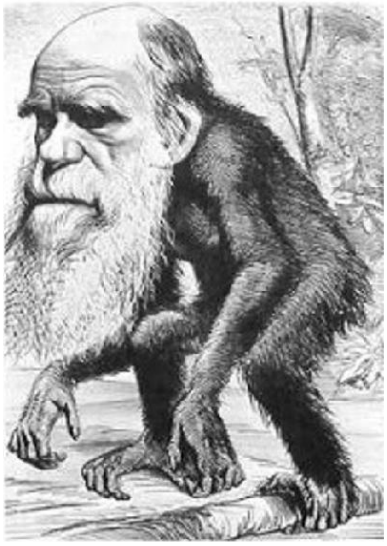
16. OCR on Xerox Copies meets Semantic Web. Have Evolutionary Theories changed?

Author: Dario Cannone, Data Scientist, Miriade (www.miriade.it)

Workflow in:

EXAMPLES/99_Community/01_Image_Processing/02_Integrations/03_Tess4J/02_OCR_meets_SemanticWeb

Posted on: July 3, 2017



The Challenge

Scientific theories are not static over time. As more research studies are completed, new concepts are introduced, new terms are created and new techniques are invented. This is of course also true for evolutionary theories. That is, evolutionary theories themselves have evolved over time!

In today's challenge we are going to show how the theory of evolution has evolved from the first Darwin's formulation to the most recent discoveries.

The foundation stone of evolutionary biology is considered to be the book "On the Origin of Species" (1859) by Charles Darwin. This book contains the first revolutionary formulation of the theory of evolutionary biology. Even though the book at the time has produced a revolution in the approach to species evolution, many of the concepts illustrated there might seem now incomplete or even obsolete. Notice that it was published in 1859, when nothing was known about DNA and very little about genetics.

In the early 20th century, indeed, the Modern Synthesis theory reconciled some aspects of Darwin's theory with more recent research findings on evolution.

The goal of this blog post is to represent the original theory of evolution as well as the Modern Synthesis theory by means of their main keywords. Changes in the used keywords will reflect changes in the presented theory.

Scanned Xerox copies of Darwin's book abound on the web, like for example at http://darwin-online.org.uk/converted/pdf/1861_OriginNY_F382.pdf. How can we make the contents of such copies available to KNIME? This is where Optical Character Recognition (OCR) comes into play.

On the other side, to find a summary of the current evolutionary concepts we could just query Wikipedia, or better DBPedia, using semantic web SPARQL queries.

Xerox copies on one side, read via OCR, and semantic web queries on the other side. Will they blend?

Topic: Changes in the theory of Evolution

Challenge: Blend a Xerox copy from a book with semantic web queries.

Access Mode: Image reading, OCR library, SPARQL queries.

The Experiment

Reading the Content of a PNG Xerox Copy via Optical Character Recognition (OCR)

Darwin's book is only available in printed form. The first part of our workflow will try to extract the content of the book from its scanned copy. This is only possible using an Optical Character Recognition software.

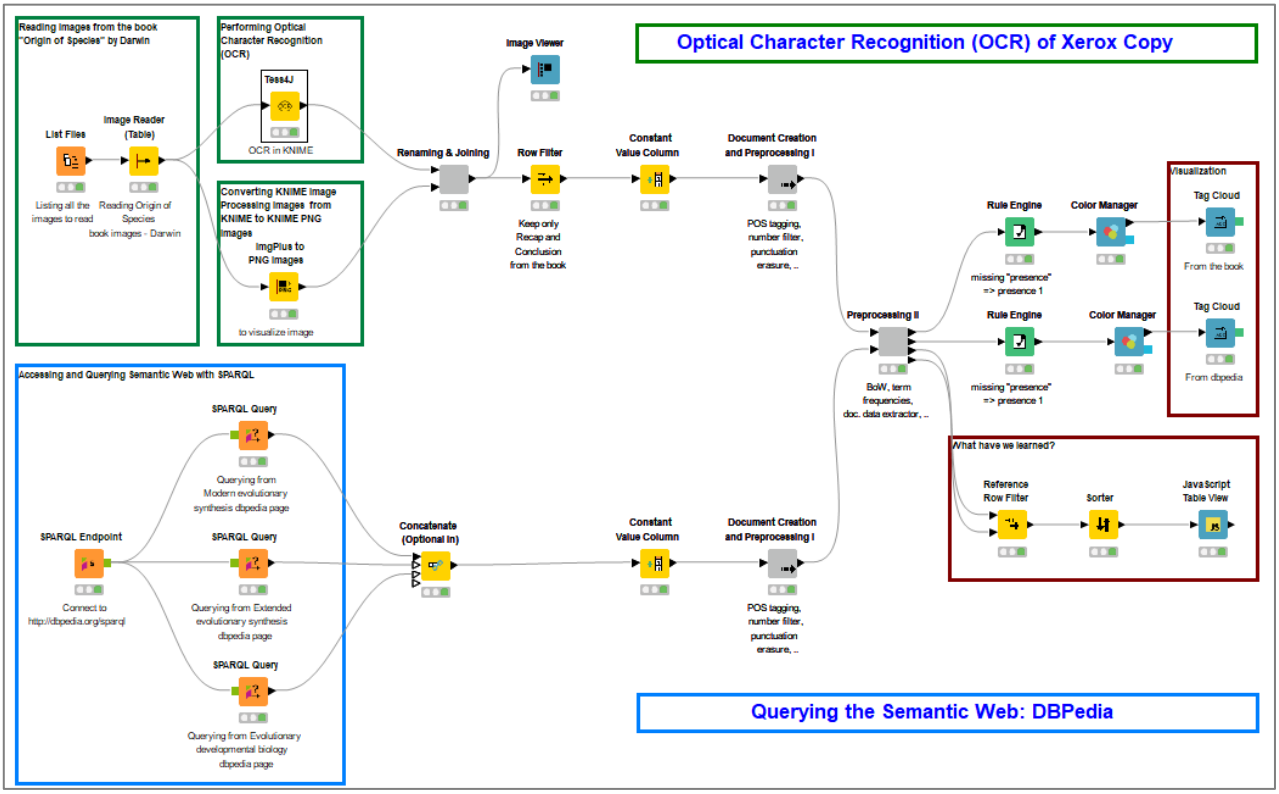
We are in luck! KNIME Analytics Platform integrates the Tesseract OCR software as the KNIME Image Processing - Tess4J Integration. This package is available under KNIME Community Contributions - Image Processing and Analysis extension (see how to install KNIME Extensions). Let's continue step by step, after the package installation.

The Image Reader node reads the locally stored image files of the pages of the book "On the Origin of Species".

- The read images are sent to the Tess4j node, which runs the Tesseract OCR library and outputs the recognized texts as Strings, one text String for each processed PNG page file.
- Each page text is then joined with the corresponding page image, converted from ImgPlusValue to PNG format in a KNIME PNGImageCell data cell. The goal of this step is to allow later on for visual inspection of the processed pages via the Image Viewer node.
- Notice that only the "Recap" and the "Conclusions" sections of Darwin's book are processed. That should indeed be enough to extract a reasonable number of representative keywords.

This part of the workflow is shown in the upper branch of the final workflow in figure 16.1, the part labelled as "Optical Character Recognition of Xerox Copy".

Figure 16.1. Final workflow where OCR (upper branch) meets Semantic Web Queries (lower branch) to show the change in keywords of original and later formulations of the theory of evolution. This workflow is available on the KNIME EXAMPLES server under *99_Community/01_Image_Processing/02_Integrations/03_Tess4J/02_OCR_meets_SemanticWeb*



Querying the Semantic Web: DBPedia

On the other side, we want to query DBPedia for the descriptions of the modern evolutionary theories. This part can be seen in the lower branch of the workflow in figure 16.1, the one named "Querying the Semantic Web: DBPedia".

- First, we establish a connection to DBpedia SPARQL endpoint: http://dbpedia.org/sparql.
- Then we make three queries on the 3 pages "Modern evolutionary synthesis", "Extended evolutionary synthesis" and "Evolutionary developmental biology" respectively.
- The results from the 3 queries are collected with a Concatenate node.

Blending Keywords from the Xerox Copy and from the Semantic Web Queries

We have now texts from the book and texts from DBPedia. We want to distill all of them to just a few representative keywords and blend them together.

- Both the two branches of the workflows pass through “Document Creation and Preprocessing I” wrapped metanode. Here standard text processing functions are applied, such as: case conversion, punctuation removal, number filtering and POS tagging.
- In the following wrapped metanode, named “Preprocessing II”, the terms are extracted and their relative frequencies are computed. The two lists of terms are then joined together. The column “presence” marks the terms common to both datasets with a 2 and the terms found in only one dataset with a 1.
- The two tag clouds are created, one from the terms in Darwin’s book and the other from the terms in the DBpedia search results. Words common to both datasets are colored in red.
- Finally, we can isolate those innovative terms, used in the description of the new evolutionary theories but not in Darwin’s original theory. This is done with a “Reference Row Filter” node and displayed with a “Javascript Table View”.

The final workflow is available on the EXAMPLES server under:
99_Community/01_Image_Processing/02_Integrations/03_Tess4J/02_OCR_meets_SemanticWeb.

The Results

Figure 16.2 shows the tag cloud with the terms from Darwin’s book “On the Origin of the Species”, while figure 16.3 shows the tag cloud from the terms found in the results from DBPedia queries.

Natural Selection is a central concept in Darwin’s evolutionary theory and this is confirmed by the central position of the two words, “natural” and “selection”, in the tag cloud in figure 16.2. The two words are in red. This means that the same terms are also found in modern evolutionary theories, even though in a less central position inside the tag cloud (Fig. 16.3).

Interestingly enough the word “evolution” is not found in Darwin’s book. Although this term was soon associated with Darwin’s theories and became popular, Darwin himself preferred to use the concept of “descent with modification” and even more “natural selection”, as we have remarked earlier.

Words like “species” and “varieties” also play a central role into Darwin’s theory. Indeed, the whole theory spawned from the observation of the species variety on Earth. On the opposite words like “modern”, “synthesis”, and “evolution” are the cornerstone of the modern evolutionary theories.

Figure 16.2. Word Cloud generated from the “Recap” and “Conclusions” sections in the book “On the Origin of the Species” by Charles Darwin

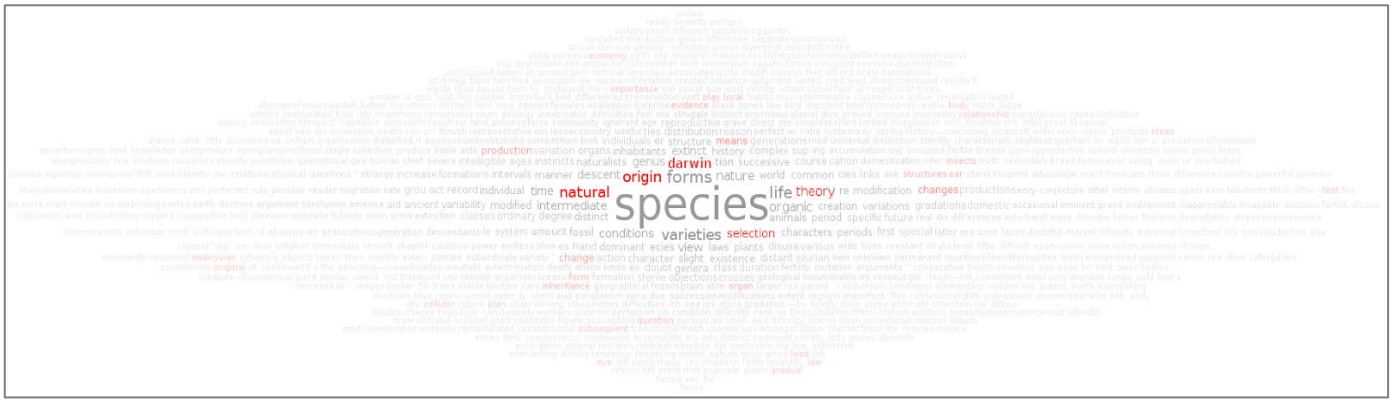
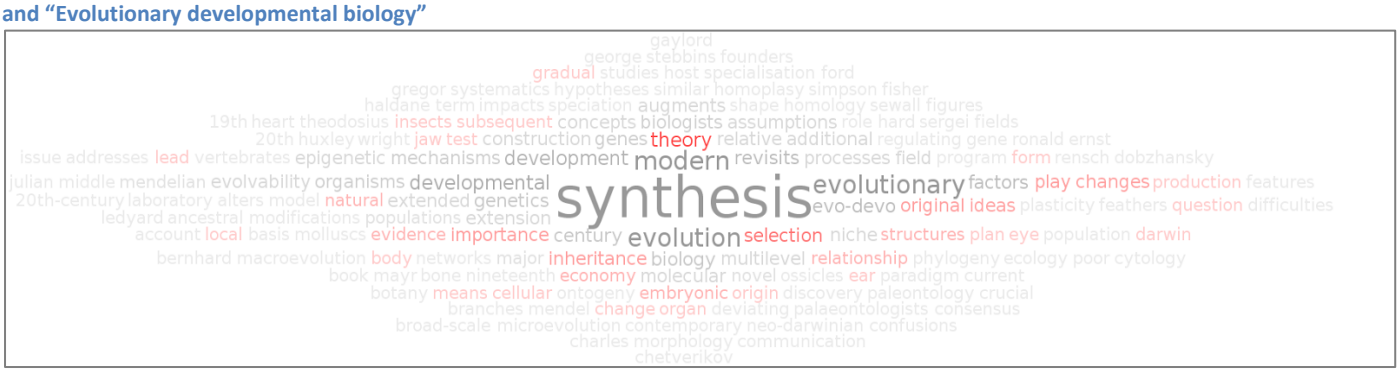


Figure 16.3. Word Cloud generated from results of DBPedia queries: "Modern evolutionary synthesis", "Extended evolutionary synthesis" , and "Evolutionary developmental biology"



What has been learned from the publication time of the book "On the Origin of Species" to the current time? One thing that Darwin for sure could not know is genetics!

If we look for the word "gene" in the table view of the Javascript Table View node, we surely find "genetics", "gene", and a number of other related words! Remember that this table view displays the terms found in the description of modern evolutionary theories but not in Darwin's original book (Fig. 16.4).

Figure 16.4. List of "genetics" related words present in the modern evolutionary theories (as derived from the queries to DBPedia) and not present yet in Darwin's original book (as derived from the OCR processing of the scanned PNG images of the book pages). Such *words are all listed in the table view of the Javascript Table View node in the final workflow.*

JavaScript Table View

Show 10 entries Search: gene

Term as String	TF rel
genetics	0.04201680672268908
genes	0.036036036036036036
epigenetic	0.02702702702702703
gene	0.009009009009009009

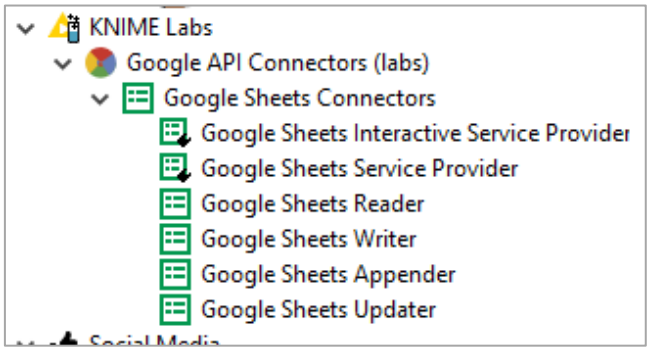
Showing 1 to 4 of 4 entries (filtered from 128 total entries) Previous 1 Next

From the results of this small experiment we understand that the evolutionary theory has itself evolved from Darwin seminal work to the modern research studies.

Also, in this experiment, we successfully blend data from a scanned Xerox copy and data from DBPedia. We have used the Tesseract OCR integration to extract the book content and SPARQL queries to extract DBPedia descriptions … and yes, they blend!

17. A Recipe for Delicious Data: Mashing Google and Excel Sheets

Author: Anna Martin, KNIME revisited by: Rene Damyon and Oleg Yasnev KNIME
Workflow in: *EXAMPLES/01_Data_Access/01_Common_File_Types/08_Google_Sheet-meets-Excel*
Posted on: July 31, 2017 revisited on: Jan 08 2018



The Challenge

A local restaurant has been keeping track of its business on Excel in 2016 and moved to Google Sheets in 2017. The challenge was then to include data from both sources to compare business trends in 2016 and in 2017, both as monthly total and Year To Date (YTD) revenues.

The technical challenge of this experiment was then of the "Will they blend?" type: mashing the data from the Excel and Google spreadsheets into something delicious… and digestible. The data blending was indeed possible and easy for public Google Sheets. However, it became more cumbersome for private Google Sheets, by requiring a few external steps for user authentication.

From the experience of such blog post, a few Google Sheets dedicated nodes have been built and released with the new KNIME Analytics 3.5. A number of new nodes indeed are now available to connect, read, write, update, and append cells, rows, and columns into a private or public Google Sheet.

The technical challenge then has become easier: accessing Google Sheets with these new dedicated nodes and mashing the data with data from an Excel Sheet. Will they blend?

Topic. Monthly and YTD revenue figures for a small local business.

Challenge. Retrieve data from Google Sheets using the new Google Sheets nodes available in KNIME Analytics Platform 3.5.

Access Mode. Excel Reader node and Google Sheets Reader node for private and public documents.

Before diving into the technicalities, let's spend a few words on the data. Both the Excel file and the Google spreadsheet contain the restaurant bill transactions with:

- date and time (DATE_TIME),
- table number (TABLE)
- bill amount (SUM)

The Experiment

The Excel Spreadsheet can be read easily with an Excel Reader node (see blog post: "Will they blend? Hadoop Hive meets Excel").

Accessing a Google Sheet Document

From a web browser:

- Create a Google account at www.google.com (if you do not have one already). This account will be used for user authentication to access the Google Sheets document.
- Open the Google Sheets document in a browser while logged into that Google account. This will allow the Google Sheets Reader node to list it alongside other documents when configuring the node.

Now in the workflow:

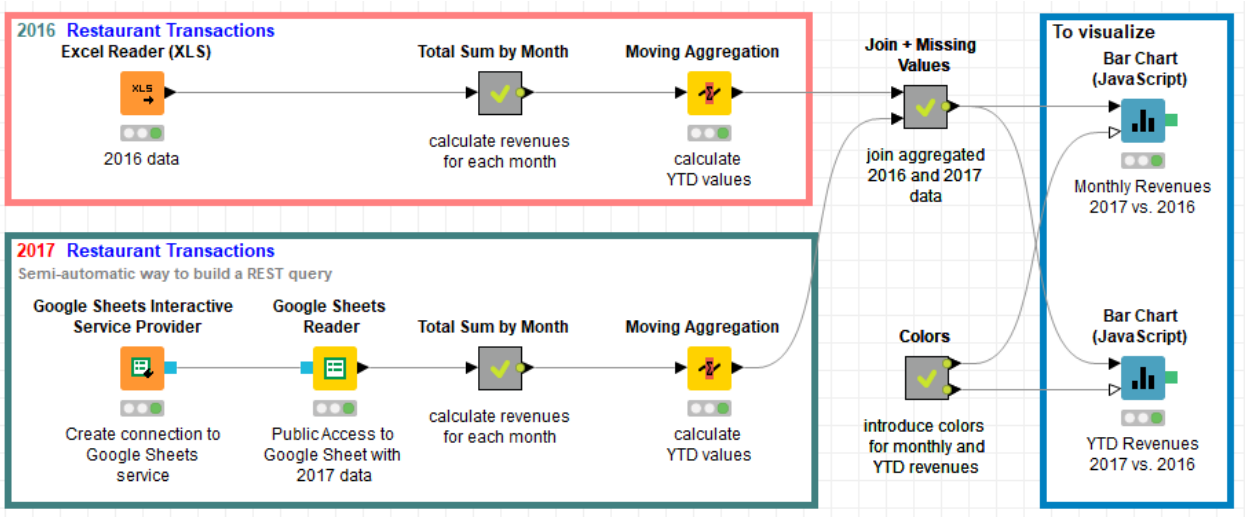
- First, we need to authenticate a Google account using the Google Sheets Interactive Service Provider node.
- Then we need to select and read the document with the Google Sheets Reader node. Here we can also, specify the range of data cells to be read. At the output port the content of the specified data cells is provided.

We are using just two Google Sheets nodes: Google Sheets Interactive Service Provider - to connect to Google's services and store our authorization credentials - and Google Sheets Reader - to select the document and return the selected data -.

Note. We use the same step sequence as well as the same nodes to access a public or a private Google Sheet document. There is no need any more to differentiate the access procedure!

The final workflow, to access data from public and private Google Sheets, is shown in figure 17.1 and can be downloaded from the KNIME EXAMPLES server from *01_Data_Access/01_Common_Type_Files/08_Google_Sheet-meets-Excel*

Figure 17.1. This workflow blends together data from an Excel spreadsheet and data from a public or private Google Sheet document. This workflow is available on the KNIME EXAMPLES server under 01_Data_Access/01_Common_Type_Files /08_Google_Sheet-meets-Excel.



Authentication of Google User

The node to authenticate the Google user is the *Google Sheets Interactive Service Provider* node. This node connects to Google services and authenticates the user via the Google authentication page.

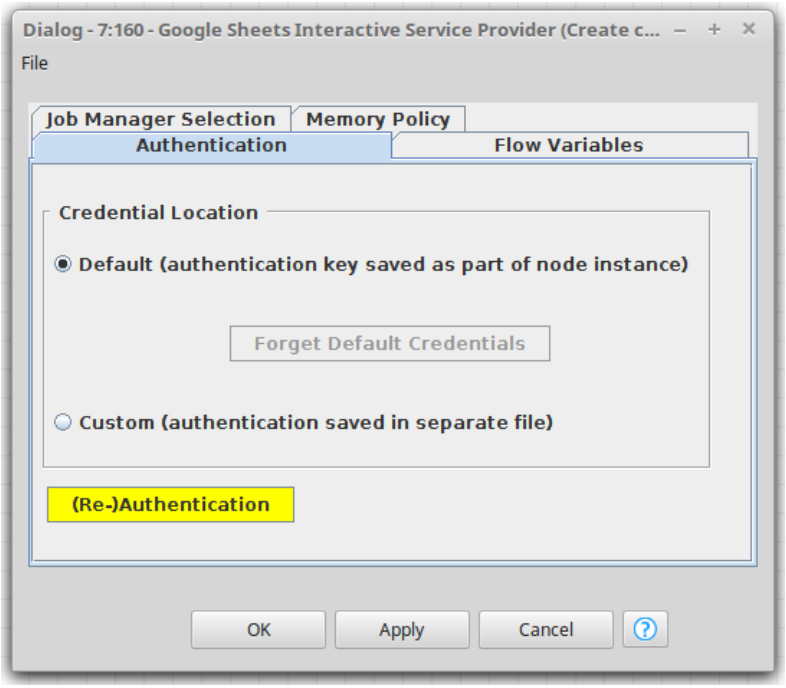
If you open the node configuration window, just click the button named *(Re-)Authentication* (Fig. 17.2). This takes you to the Google authentication page on your web browser. If you insert here your credentials, the node can then try to establish a connection with Google services.

If connection was successful, the (Re-)Authentication button will turn some shade of green: full green on Mac, green contour on Windows.

These authentication credentials can be saved as part of the workflow (option *"Default"*) or they can be persisted to a separate file to share with other workflows (option "*Custom*").

Note. When exporting your workflow or when in need to make the workflow forget the authentication credentials, just click the button named Forget Default Credentials.

Figure 17.2. The configuration dialog for the Google Sheets Interactive Service Provider node



Selecting and Reading the Google Sheet Document

Connecting a Google Sheets Reader node with our Google Sheets Interactive Service Provider node allows to select the document and, if needed, also a particular sheet or a range or data cells.

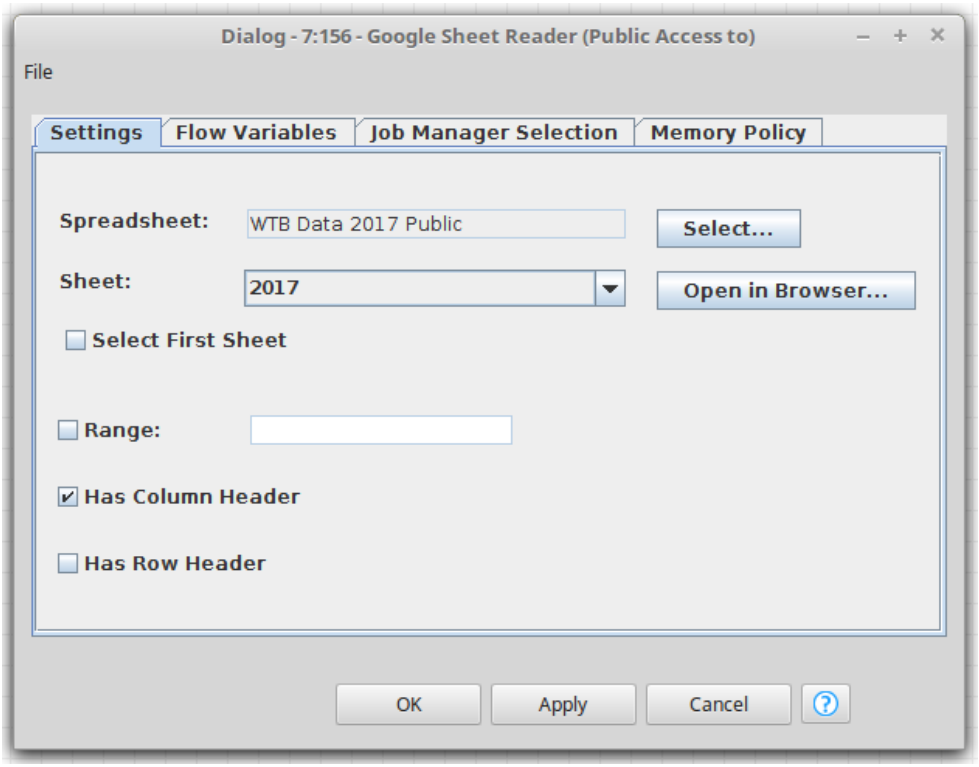
Configuration is as simple as clicking the Select button to display a list of available documents and choosing the one we want. The "Open in Browser" option in the configuration window of the Google Sheets Reader node opens the document on a web browser for a quick inspection.

Remember! If a document doesn't appear in this list, make sure that you have permissions to access it and that you've opened it once within a browser to associate it with your Google account.

For this experiment, we accessed and read the following Google Sheet document Google Sheets sample data: Google Sheets Public Data 2017.

We know that our data has header information in the first row of the document. Selecting "Has Column Header" will set these values as the names of the columns in the node output.

Figure 17.3. The configuration dialog for the Google Sheets Reader node



The rest of the workflow calculates the revenues by month as total sum and as Year To Date (YTD) cumulative sum. While the monthly total sum is calculated with a GroupBy node, the YTD cumulative sum is calculated with a Moving Aggregation node.

The Results

The two bar charts below show the restaurant revenues in euros as a total monthly sum and as a Year To Date (YTD) monthly cumulative sum respectively, for both groups of transactions in 2016 (light orange, from the Excel file) and in 2017 (darker orange, from the Google Sheet document).

We are happy to see that the small restaurant we used as an example has increased its business sales in 2017 with respect to the same months in 2016.

We are also happy to see that new Google Sheets nodes available now in KNIME Analytics Platform 3.5!

In this experiment we retrieved and blended data from an Excel spreadsheet and a Google Sheet document. Our Excel spreadsheets and Google Sheets documents blended easily, to produce a delicious dish for our restaurant business.

Figure 17.4. Total monthly revenues for our restaurant in year 2016 (light orange on the left) and in year 2017 (darker orange on the right). Business in 2017 seems to be better than in 2016.

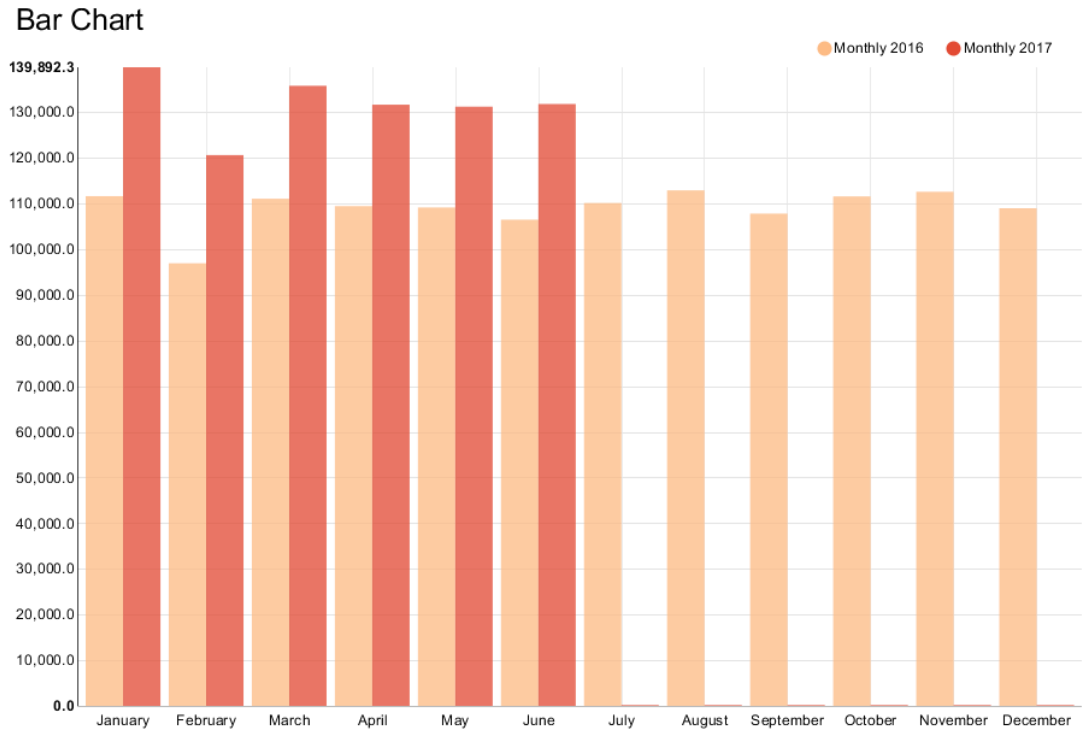
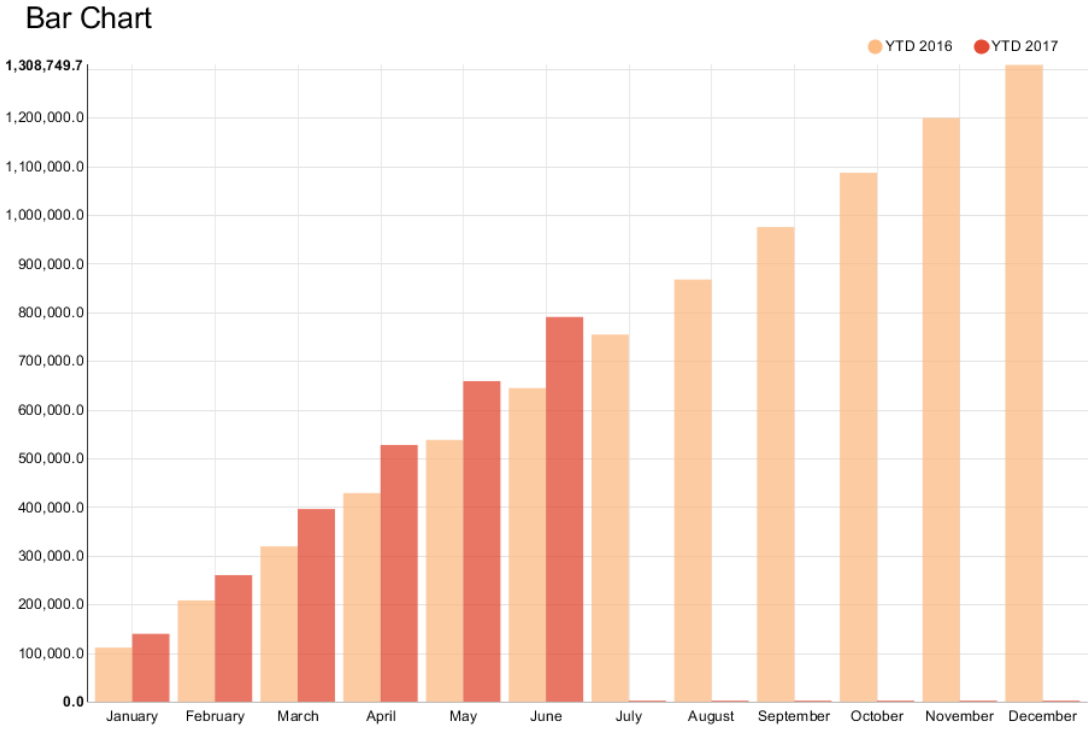


Figure 17.5. Cumulative YTD revenues for our restaurant in year 2016 (light orange on the left) and in year 2017 (darker orange on the right). Also here, business in 2017 seems to be better than in 2016.



So, if you are asked by your friend running a local business whether you can blend data from Excel spreadsheets and Google Sheet documents, the answer is: Yes, they blend … and please use the new Google Sheets nodes!

18. SugarCRM meets Salesforce. Crossing Accounts and Opportunities

Author: Roland Burger, KNIME AG
Workflow in: *EXAMPLES/01_Data_Access/05_REST_Web_Services/06_SugarCRM_meets_Salesforce*
Posted on: September 25, 2017

The Challenge

Businesses use Customer Relationship Management (CRM) systems to keep track of all their customer related activities – creating leads and opportunities, managing contacts and accounts, sending quotes and invoices, etc. As long as it is somehow related to the stream of revenue, it is (or at least should be) stored in a CRM system.

Since there is more than one CRM solution on the market, there is a distinct chance that your organization uses multiple CRM platforms. While there might be sound reasons for this, it also poses a significant challenge: How do you combine data from several platforms? How do you generate a single, consolidated report that shows you how well the sales activities of your whole company are going?

One option is to export some tables, fire up your spreadsheet software of choice, and paste the stuff together. Then do the same thing next week. And the week after. And the week after that one (you get the point). Doesn't sound too enticing? Fear not! This is KNIME, and one of our specialties is to save you the frustration of doing things manually. Fortunately, both SugarCRM and Salesforce allow their users to access their services via REST API, and that is exactly what we are going to do in this blog post.

There are a couple of prerequisites here. First of all, you obviously need accounts for SugarCRM and Salesforce. If you don't have them but still want to try this yourself, you'll be happy to see that both companies offer free trial licenses:

https://info.sugarcrm.com/trial-crm-software.html?utm_source=crmsoftware&utm_medium=referral&utm_campaign=crmsoftware-review

https://developer.salesforce.com/signup

You can learn more about how to use the REST APIs of SugarCRM and Salesforce here:

http://support.sugarcrm.com/Documentation/Sugar_Developer/Sugar_Developer_Guide_7.9/Integration/Web_Services/v10/

https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_what_is_rest_api.htm

Topic. Get a consolidated view of all customer data from two separate platforms

Challenge. Query data from SugarCRM and Salesforce via their APIs

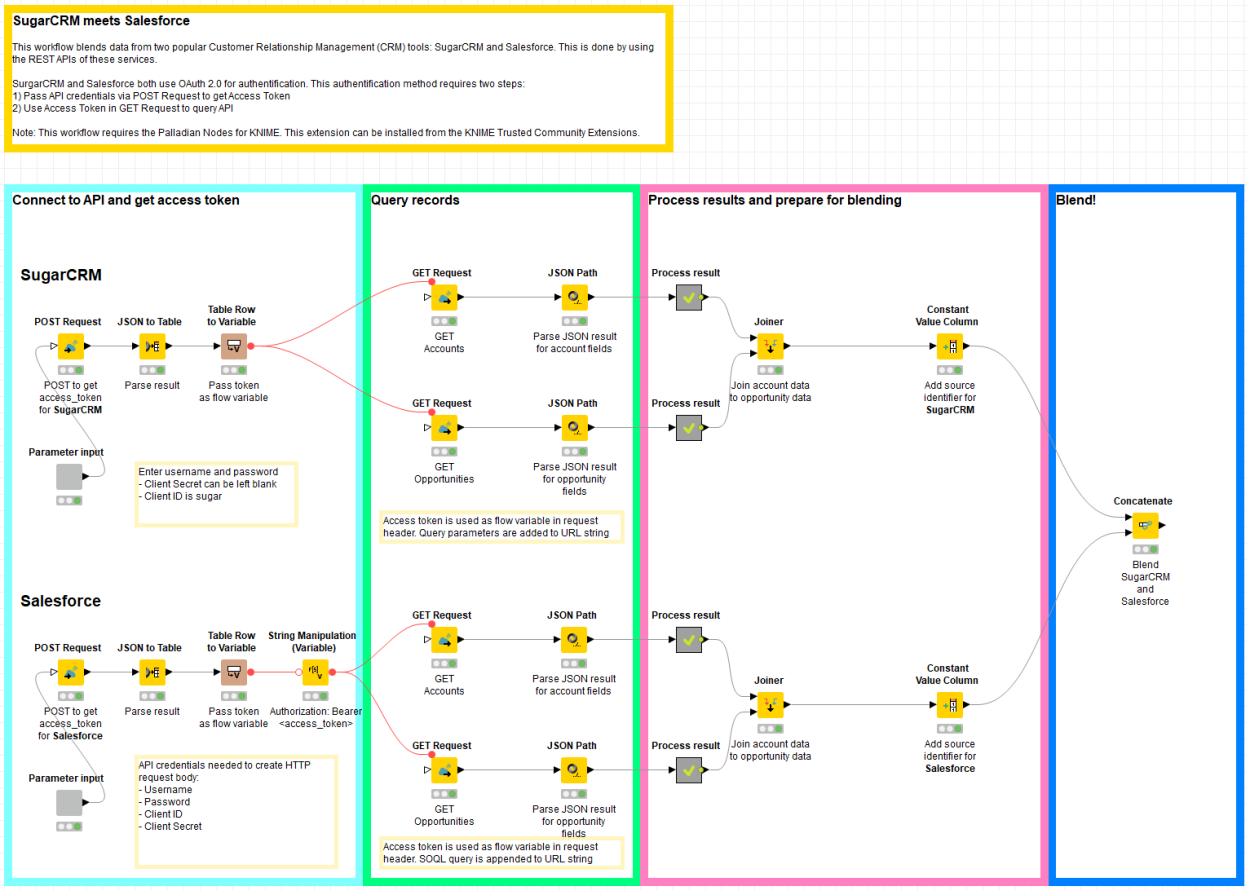
Access Mode. KNIME REST Web Services

The Experiment

The Challenge here is to access data from a Salesforce CRM and data from a SugarCRM and blend them together. Luckily I have access to installations of both CRM systems.

The workflow depicted in figure 1 accesses a SugarCRM system (upper branch) and a Salesforce CRM system (lower branch). Let's see which steps are required more in detail.

Figure 18.1. This workflow accesses and blends data from a SugarCRM installation and from a Salesforce installation. This workflow is available on the KNIME EXAMPLES Server under 01_Data_Access/05_REST_Web_Services/06_SugarCRM_meets_Salesforce



Accessing data from SugarCRM

SugarCRM can be accessed via REST API. The data can be retrieved via an authorized GET Request, that is a GET Request using an authorization token. Authorization token is obtained via a POST Request.

1. The first step then consists of retrieving the authorization token via a POST Request. For the POST Request we need: the URL of the authorization endpoint and the parameters to fit in the request body. The request body has to be delivered in JSON format.

   The first wrapped metanode in the upper branch of the workflow in figure 18.1, named “Parameter Input”, creates the appropriate request body in JSON format (see figure 18.2).

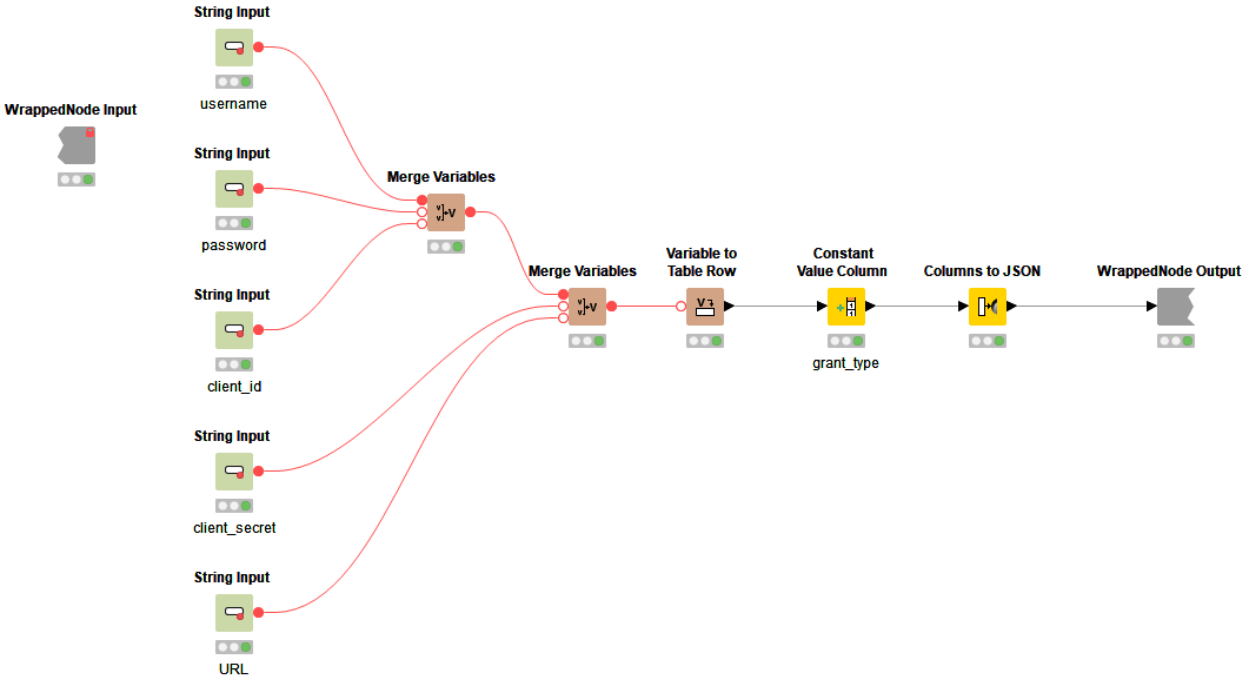
   a. The required parameters are entered via String Input nodes: Username, Password, Client ID, Client Secret (it can be empty). Additionally, the URL of authorization endpoint shaped as described in https://<site_url>/rest/v10/oauth2/token, is also entered.

   b. After merging all flow variables and transferring all of them to a table, the column for grant_type is added through the Constant Value Column node. Option grant_type can assume many values, but for the purpose of this blog post, it will be “password”.

   c. The Columns to JSON node packages all parameter columns into a JSON object, which will be used as request body.

   d. After executing the “Parameter Input” wrapped metanode, we get a table with two columns: a JSON request body containing the credentials and the URL endpoint.

Figure 18.2. Content of the Metanode named “Parameter Input”. This node collects the parameters and the URL endpoint required to get an authorization code to access the SugarCRM tool. Request parameters are packaged together in a JSON structure.



2. Next, the JSON structure request body and the endpoint URL are fed into a POST Request node to get the access token. In the configuration window, we feed the URL column into the appropriate setting in the “Connection Settings” tab. Then, in the “Request Body” tab we feed the JSON column in “Use column’s content as body” setting. Let’s run the node. If everything works out, we should get a HTTP status code 200 as well as a JSON column named “body” as a response.

3. The column named “body” is transformed back into a KNIME data table through the node JSON to Table. One of the resulting columns is named access_token and contains the authorization token we need to access SugarCRM data. This token is finally transformed into a flow variable, named “access_token”, and then injected into two GET Request nodes.

4. The first GET Request node queries account data; the second GET Request node queries opportunity data. We configure the two GET Request nodes appropriately. For a list of available endpoints, see: http://support.sugarcrm.com/Documentation/Sugar_Developer/Sugar_Developer_Guide_7.9/Integration/Web_Services/v10/Endpoints/

   URL is set as https://<site_url>/rest/v10/Accounts or https://<site_url>/rest/v10/Opportunities followed by “&” and some optional parameters.

   Here, we only ask for five records (max_num=5) and only for records created by the current user (my_items=TRUE).

   The final request URL would then be something like:

   https:// <site_url>/rest/v10/Accounts&max_num=5&my_items=TRUE

   Hint. To get all available records, use max_num=-1.

In the Request Headers tab, click “Add header parameter”. For Header key, use “OAuth-Token”. For Header value, use “test” or some other placeholder. Next, go to tab Flow Variables, identify setting “Request header key selector”, and set “access_token” as string variable for item “0”.
Let’s now execute the node and check the results. Again, we want a 200 status and a JSON output column.

5. The JSON we got as a response from the server has all the data we need. The JSON Path node will parse the JSON structure and extract the data we are interested in. For this example, I extract only Account ID, Account name, industry, country, and ID of the user assigned to this account.

   Hint. Make sure to use a collection query rather than a single query since we want the data for all records.

6. Next, the Ungroup node transforms the collection items into single data rows.

7. Finally, the Constant Value Column node adds a source identifier, which is “SugarCRM”.

Accessing data from Salesforce

In principle, the procedure to access data from Salesforce is very similar to the procedure used for SugarCRM.

1. Also here we start with a “Parameter Input” wrapped metanode, similar to the one we used to access data from SugarCRM. Also in this metanode we enter the credentials to access the system. If you don’t know how to create Salesforce API credentials, have a look here: http://www.calvinfroedge.com/salesforce-how-to-generate-api-credentials/. Note that for Password, you need to use the Security Token.

2. In Salesforce, all parameters need to be URL-encoded before reaching the body of the POST Request. The FormEncodedHttpEntityCreator node performs URL-encoding. This node is part of the KNIME Palladian extension. This node returns a binary column, and therefore we use a Binary Objects to Strings node to convert the parameter object back into String format.

3. The parameter options in the newly generated String represent the body for the POST Request. Again in the POST Request node, we set the endpoint URL column and the parameter String column to use as Request Body. Here we also need to add a Request Header with Header Key “Content-Type” and Header value “application/x-www-form-encoded”.
   Executing the POST Request node should also return a 200 status and a JSON column containing the access token.

4. We then convert the JSON structured response to a KNIME table, we extract the access token and we pass it to the GET Request node in the shape of a flow variable.
   For use as a request header, the access token needs to have the format “Bearer <access_token>”. This is done in the String Manipulation (Variable) node.

5. Next we use a GET Request node to access the data within Salesforce. For a list of the objects and fields you can query, log in to Salesforce, go to Setup -> Objects and then Fields -> Object Manager.

   a. The query is entered in the URL field of the “Connection Settings” tab. It takes the following format: https://<your_instance>.salesforce.com/services/data/v39.0/query?q=<query_string>
      The query string is an SOQL query. At first glance, SOQL looks very much like SQL, so people familiar with SQL syntax should not have any problems here (for more information, see here:

https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql.htm). Here is an example query (note the spaces encoded as "+"):
https://eu11.salesforce.com/services/data/v39.0/query?q=SELECT+Name,+AccountNumber,+OwnerId,+BillingAddress,+Industry+FROM+Account

b. In the Request Headers tab, we add the authorization token. We click "Add header parameter". As Header key, we use "Authorization". As Header value, we use "test" or some other placeholder. Next, go to the Flow Variables tab, identify the Request header key selector field and set the flow variable containing the token for item "0".

c. Let's run it and hope for a 200 status! If execution is successful, we get a JSON structured response containing the requested data.

d. Also in this case, we rely on two GET Request nodes. One node extracts information about the Accounts and one node about the Opportunities in the system.

6. Next, we use a JSON Path node to extract the relevant data.

7. This is followed by some more data processing, after which we can join the account data with the opportunity data. To wrap things up, we add a column that identifies Salesforce as the source of this data.

Blending the data from Salesforce with the data from SugarCRM

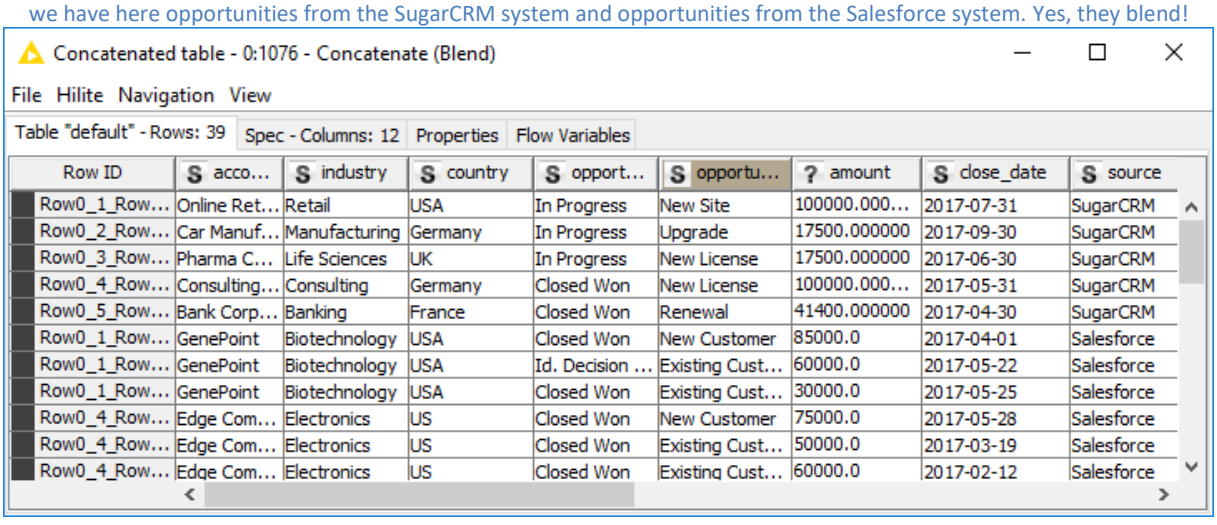
The only thing left now is to concatenate the SugarCRM data and the Salesforce data. Will they blend? Indeed, we use a Concatenate node to put together the two sets of data.

The final workflow is reported in figure 18.1. It successfully blends data from SugarCRM and Salesforce and it is downloadable from the KNIME EXAMPLES server under:
01_Data_Access/05_REST_Web_Services/06_SugarCRM_meets_Salesforce

The Results

Yes, they blend! Thanks to the REST API integration in KNIME Analytics Platform, we were able to extract data from a SugarCRM system and from a Salesforce system and to blend them in a single table (Fig. 18.3).

Figure 18.3. This data table represents the results from the blending workflow depicted in figure 18.1. As you can see from the source column, we have here opportunities from the SugarCRM system and opportunities from the Salesforce system. Yes, they blend!



Row ID	acco...	industry	country	opport...	opportu...	amount	close_date	source
Row0_1_Row...	Online Ret...	Retail	USA	In Progress	New Site	100000.000...	2017-07-31	SugarCRM
Row0_2_Row...	Car Manuf...	Manufacturing	Germany	In Progress	Upgrade	17500.000000	2017-09-30	SugarCRM
Row0_3_Row...	Pharma C...	Life Sciences	UK	In Progress	New License	17500.000000	2017-06-30	SugarCRM
Row0_4_Row...	Consulting...	Consulting	Germany	Closed Won	New License	100000.000...	2017-05-31	SugarCRM
Row0_5_Row...	Bank Corp...	Banking	France	Closed Won	Renewal	41400.000000	2017-04-30	SugarCRM
Row0_1_Row...	GenePoint	Biotechnology	USA	Closed Won	New Customer	85000.0	2017-04-01	Salesforce
Row0_1_Row...	GenePoint	Biotechnology	USA	Id. Decision ...	Existing Cust...	60000.0	2017-05-22	Salesforce
Row0_1_Row...	GenePoint	Biotechnology	USA	Closed Won	Existing Cust...	30000.0	2017-05-25	Salesforce
Row0_4_Row...	Edge Com...	Electronics	US	Closed Won	New Customer	75000.0	2017-05-28	Salesforce
Row0_4_Row...	Edge Com...	Electronics	US	Closed Won	Existing Cust...	50000.0	2017-03-19	Salesforce
Row0_4_Row...	Edge Com...	Electronics	US	Closed Won	Existing Cust...	60000.0	2017-02-12	Salesforce

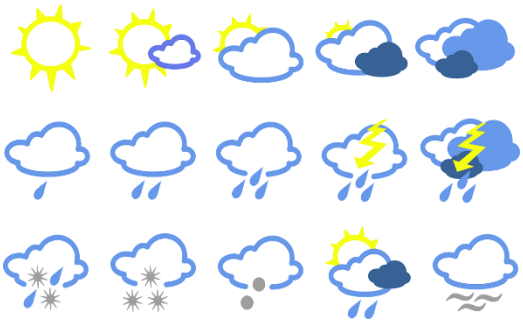
We can now create a nice report based on the extracted data, send it around, and make everyone happy - all those promising opportunities together at last.

Need to do the same thing again next week? Just re-execute the workflow! (Or let KNIME Server automate that for you). No more manual data exports, no more copy and paste, but a lot more time for other things.

In this blog post, we used the SugarCRM and Salesforce REST APIs to retrieve data from both systems. While this is a great thing to do in itself, there's still a lot more to it! The GET Request and POST Request nodes are very powerful tools within the KNIEM Analytics Platform. You can access a number of other REST services, authenticated and not, available on the Internet, for reading, searching, extracting, and even modifying and writing records!

19. Finnish meets Italian and Portuguese through Google Translate API. Preventing Weather from Getting Lost in Translation.

Author: Heather Fyson and Phil Winters, KNIME AG
Workflow in: *01_Data_Access/05_REST_Web_Services/07_Translate_Using_Google_API*
Posted on: October 9, 2017



The Challenge

Talking about the weather isn't just a British pastime. Everyone does it. Especially when they're about to go on holiday.

Winter & Sun Travel organizes a diverse range of holidays for people, sending them to different destinations all over the world.

To make sure the weather doesn't get lost in translation, they use KNIME Analytics Platform and Google Translate! Winter & Sun, a UK-based company, has put together a list of weather terms in English which they use as a base to translate into both the customer's native language and the language spoken at their holiday destination. The weather terms - translated specifically to suit each customer's language pair - are then put together in a small report the customer can take on holiday and refer to whenever they need to.

Three sets of customers have just booked holidays with Winter & Sun Travel:

The Hämäläinen Family from **Finland** are desperate to find sun and light and are off to Procida, **Italy** for a beach holiday.

Amanda and Mário from the Amazonas (one of the world's least windy regions) in **Brazil** want to go to Sylt, **Germany** for some kitesurfing.

The **Browns** from Yuma in **Arizona, US** are escaping the hot desert for some skiing in **France**.

Topic. Customizing Documents for the end-user needs

Challenge. Blending different languages into a final customized document

Access Mode. Google Translate API

The Experiment

1. Get a Google API key enabled for the Google Translate service from the Google API Console. Here is some help on how to obtain a Google API key. To understand how to use the Google Translate URL, which options are available, and how to append the API key, check the Google Translate API instructions.

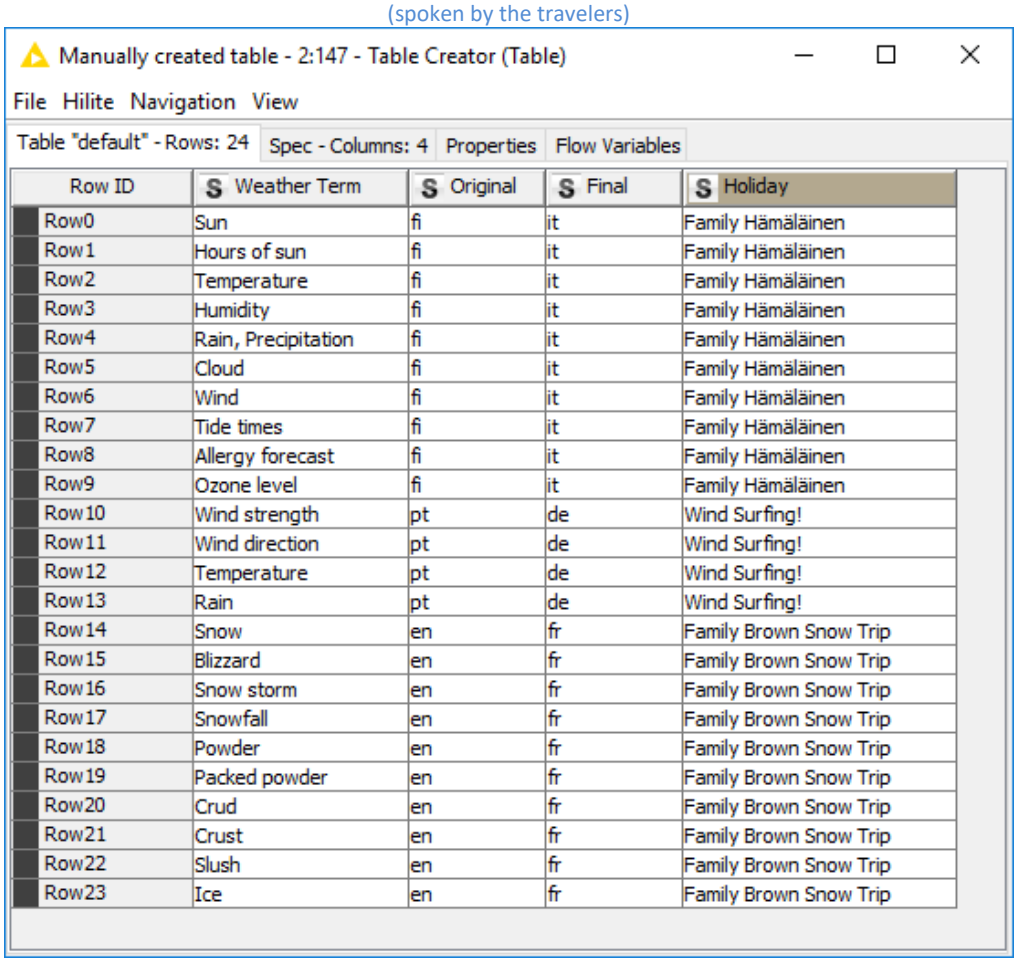
   Beware. Google Translate API is a paid service! Check its pricing information before creating an API key.

2. Winter & Sun put together a table of terms in English that would suit these three different types of holiday.

In the same table, they also entered language codes for the required translations, each time going from English into an “Original” and a “Final” language. A complete list of Google API Translation codes is available here: https://cloud.google.com/translate/docs/languages

In the KNIME workflow, a Table Creator node is introduced to host this translation map.

Figure 19.1. Translation Map including term in English (spoken by the agency), original language (in the touristic location), and final language (spoken by the travelers)



Row ID	Weather Term	Original	Final	Holiday
Row0	Sun	fi	it	Family Hämäläinen
Row1	Hours of sun	fi	it	Family Hämäläinen
Row2	Temperature	fi	it	Family Hämäläinen
Row3	Humidity	fi	it	Family Hämäläinen
Row4	Rain, Precipitation	fi	it	Family Hämäläinen
Row5	Cloud	fi	it	Family Hämäläinen
Row6	Wind	fi	it	Family Hämäläinen
Row7	Tide times	fi	it	Family Hämäläinen
Row8	Allergy forecast	fi	it	Family Hämäläinen
Row9	Ozone level	fi	it	Family Hämäläinen
Row10	Wind strength	pt	de	Wind Surfing!
Row11	Wind direction	pt	de	Wind Surfing!
Row12	Temperature	pt	de	Wind Surfing!
Row13	Rain	pt	de	Wind Surfing!
Row14	Snow	en	fr	Family Brown Snow Trip
Row15	Blizzard	en	fr	Family Brown Snow Trip
Row16	Snow storm	en	fr	Family Brown Snow Trip
Row17	Snowfall	en	fr	Family Brown Snow Trip
Row18	Powder	en	fr	Family Brown Snow Trip
Row19	Packed powder	en	fr	Family Brown Snow Trip
Row20	Crud	en	fr	Family Brown Snow Trip
Row21	Crust	en	fr	Family Brown Snow Trip
Row22	Slush	en	fr	Family Brown Snow Trip
Row23	Ice	en	fr	Family Brown Snow Trip

3. Since Google API doesn’t know what to do with an “English to English” translation, if English is required as final language, those rows are split out and not translated via a Row Splitter node.

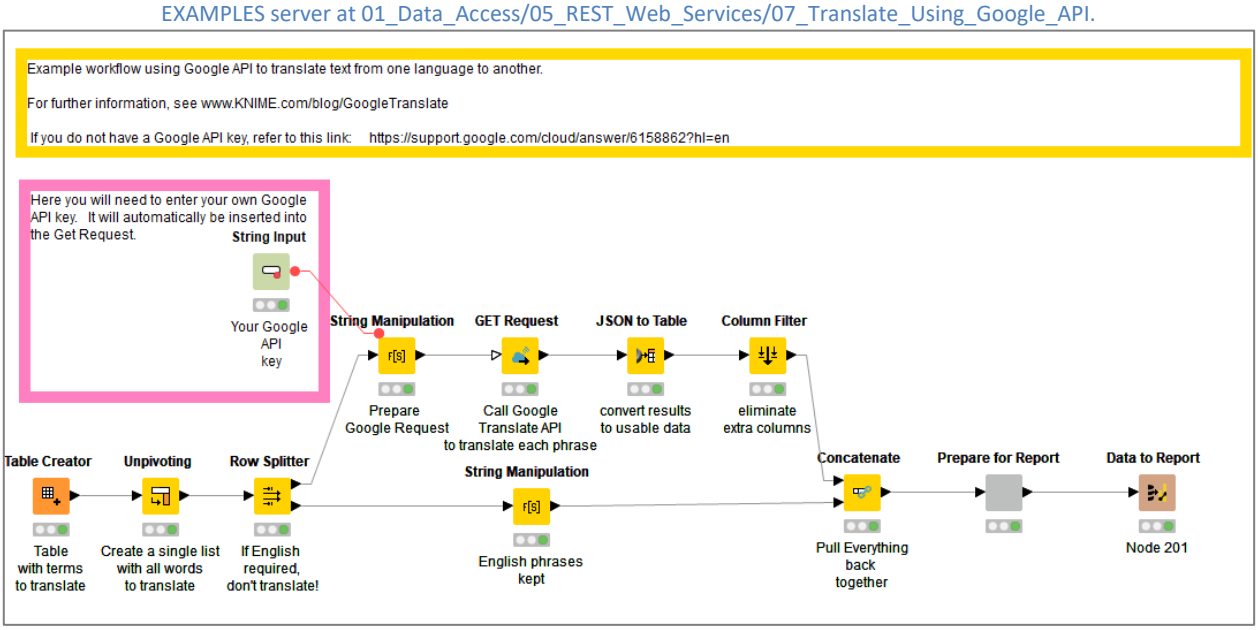
4. In the following String Manipulation node, Google Translate Requests are built that combine their Google API Key along with the required options.

5. The GET Request node calls Google Translate service, translates the terms – into Finnish and Italian for Hämäläinens, into Brazilian and German for Amanda and Mario and into French for the Browns from Arizona. The request is actually submitted once for each row in the input table.
6. The result from each request to the Google Translate API comes back as JSON statement appended to each row. The JSON structure is then broken down into its parts with the JSON to Table node and only those columns that are of interest are kept.

7. The English terms, that skipped translation, are added to the end of the table so that we have a complete set of translated terms.

8. Finally, some cleanup is performed to be able to transfer everything to BIRT, creating the final report for each family.

Figure 19.2. The Winter & Sun translation workflow accessing the Google Translate API service. This workflow is available on the KNIME EXAMPLES server at 01_Data_Access/05_REST_Web_Services/07_Translate_Using_Google_API.



The Results

Using the translated results, Winter & Sun Travel are able to produce weather cards for their customers, quickly and automatically, personalized into the respective languages. All they need is KNIME Analytics Platform and an account for Google Translate.

If you want to try this out, you can access this workflow through the KNIME EXAMPLES server at:

01_Data_Access/05_REST_Web_Services/07_Translate_Using_Google_API.

Just enter your own Google API key enabled for the Google Translate service.

And Bob's your uncle / fertig ist der Lack / Le tour est joué / se on siinä / aí está!

Figure 19.3. Weather Card by Winter & Sun Travel agency for the Brown family, created with the workflow hosted on the KNIME EXAMPLES server at 01_Data_Access/05_REST_Web_Services/07_Translate_Using_Google_API.

Family Brown Snow Trip

Original Language: **English**	Final Language: **French**
Snow	Neige
Blizzard	Blizzard
Snow storm	Tempête de neige
Snowfall	Chute de neige
Powder	Poudre
Packed powder	Poudre enroulée
Crud	Crud
Crust	Croûte
Slush	Neige fondante
Ice	La glace

Have a Wonderful Holiday! Your Winter and Sun Team

20. Google Big Query meets SQLite. The Business of Baseball Games

Author: Dorottya Kiss, EPAM
Workflow in: EXAMPLES/*01_Data_Access/02_Databases/10_GoogleBigQuery_meets_SQLite*
Posted on: November 13, 2017

The Challenge

They say if you want to know American society, first you have to learn baseball. As reported in a New York Times article, America had baseball even in times of war and depression, and it still reflects American society. Whether it is playing, watching, or betting on the games, baseball is in some way always connected to the lives of Americans.

According to Accuweather, different weather conditions play a significant role in determining the outcome of a baseball game. Air temperature influences the trajectory of the baseball; air density has an impact on the distance covered by the ball; temperature influences the pitcher's grip; cloud coverage affects the visibility of the ball; and wind conditions - and weather in general - have various degrees of influence on the physical wellbeing of the players.

Another interesting article on Crowdhitter describes the fans' attendance of the games and how this affects the home team's success. Fan attendance at baseball games is indeed a key factor, in terms of both emotional and monetary support. So, what are the key factors determining attendance? On a pleasant day are they more likely to show up in the evening or during the day, or does it all just depend on the opposing team?

Some time ago we downloaded the data about attendance at baseball games for the 2016 season from Google's Big Query Public data set and stored them on our own Google Big Query database. For the purpose of this blending experiment we also downloaded data about the weather during games from Weather Underground and stored these data on a SQLite database.

The goal of this blending experiment is to merge attendance data at baseball games from Google Big Query with weather data from SQLite. Since we have only data about one baseball season, it will be hard to train a model for reliable predictions of attendance. However, we have enough data for a multivariate visualization of the various factors influencing attendance.

Topic: Multivariate visual investigation of weather influence on attendance of baseball games.

Challenge: Blend attendance data from Google Big Query and weather data from SQLite.

Access Mode: Database Connector node with Simba 4.2 JDBC driver compatible with access to Google Big Query and dedicated SQLite Connector node.

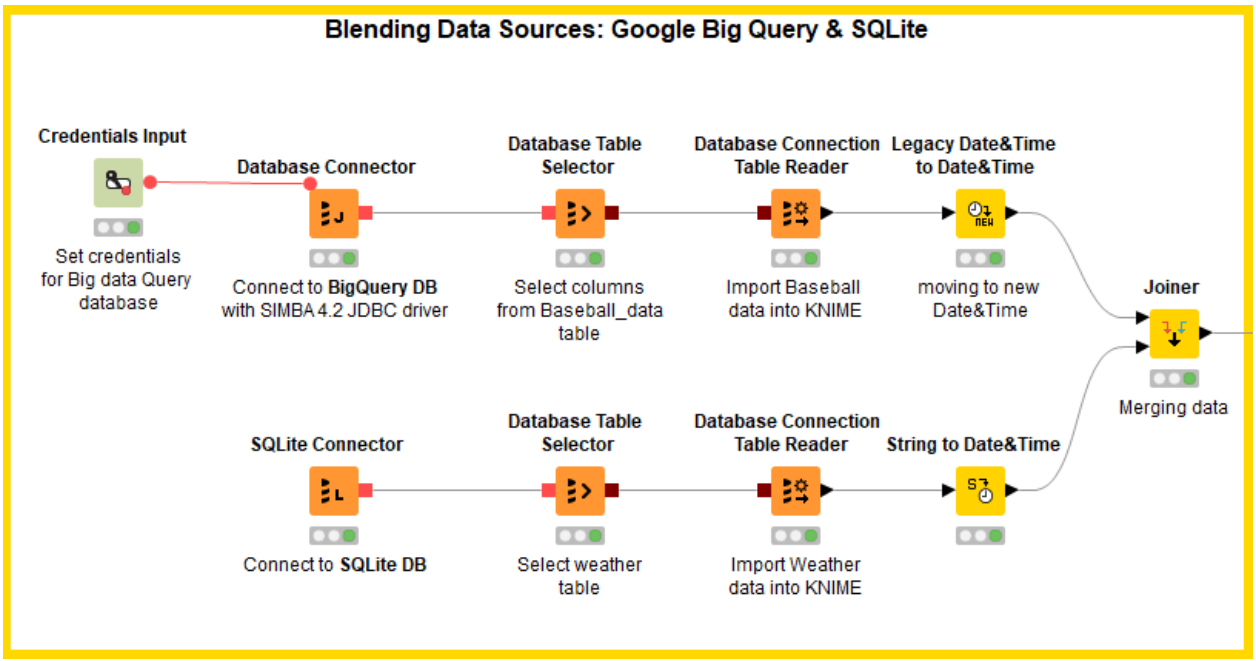
The Experiment

The first part of the workflow connects with the two data storage platforms: Google Big Query and SQLite.

KNIME Analytics Platform provides dedicated and generic nodes for database access. Using a generic node instead of a dedicated node does not change the database access performance. It just requires a few additional configuration settings that are not required in a dedicated node.

The Database/Connectors category in the Node Repository offers a dedicated connector for SQLite, but not a dedicated connector for Big Query, at least not yet. We will then use the dedicated SQLite Connector node to access SQLite and a generic Database Connector with the appropriate JDBC driver to connect to Google Big Query.

Figure 20.1. First part of the workflow merging Google Big Query and SQLite data

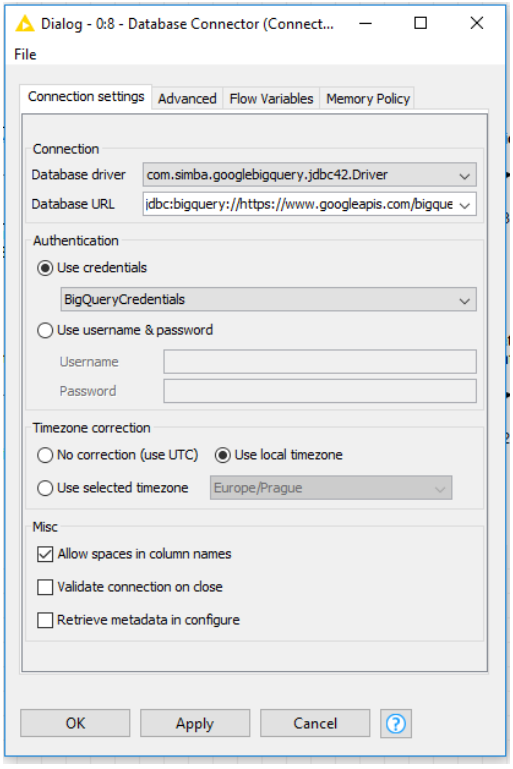


1. *Connect to Google Big Query*

To connect to Google Big Query, we downloaded the free Simba 4.2 compatible JDBC driver from the Google Cloud Platform (https://cloud.google.com/bigquery/partners/simba-drivers/).

The Simba 4.2 driver was then uploaded into KNIME Analytics Platform under: *File -> Preferences ->KNIME -> Databases* through the "Add directory" button.

Figure 20.2. Configuration window of the Database Connector node used to connect to Google Big Query



Then in a new workflow, we created a generic Database Connector node. Since the Simba compatible driver has already been uploaded, it now appears in the list of available database drivers for the setting “Database driver” in the node configuration window.

Besides the database JDBC driver, the Database Connector node also requires the database URL, the credentials, and a number of other database related settings (Fig. 20.2).

Note. Each database URL requires the appropriate protocol. For Google Big Query this is **jdbc:bigquery://**

Credentials can be supplied directly as username and password or via a Credentials Input node. This last option is recommended, since it provides default credential encryption and therefore it is more secure.

After that, a Database Table Selector node points to the baseball game attendance table according to a quite simple SQL query.

```
SELECT startTime, attendance, venueCity FROM Baseball_data.games_wide_copy
```

Finally, a Database Connection Table Reader node imports the data into KNIME Analytics Platform, where we now have the attendance data for all baseball games in the US from 2016-04-03 to 2016-10-02.

The full node sequence can be seen in the upper branch in figure 20.1.

2. *Connect to SQLite*

Now we need to import the weather data which are stored in a SQLite database.

It might seem surprising, but the steps required to access an SQLite database are exactly the same as the steps required to access the Google Big Query database (see lower branch in Fig. 20.1). Unlike Google Big Query, SQLite enjoys the privilege of a dedicated connector node, in which the driver file is embedded.

In the same workflow, we created a SQLite Connector node, where we provided the path of the SQLite file.

After that, a Database Table Selector node points to the Weather table according to a quite simple SQL query:

```
SELECT * FROM Weather
```

Finally, a Database Connection Table Reader node imports the data into KNIME Analytics Platform, where we now have the weather data for all baseball games in the US from 2016-04-03 to 2016-10-02.

Note. If you are lost as to which table to select in the configuration window of the Table Selector node, the Fetch Metadata button in the top left corner can come to the rescue. Indeed, the Fetch Metadata button allows you to explore the database content.

3. *Blending Baseball and Weather data*

Both datasets contain date type fields. Dates have been stored as Strings in SQLite and as DateTime objects in Google Big Query.

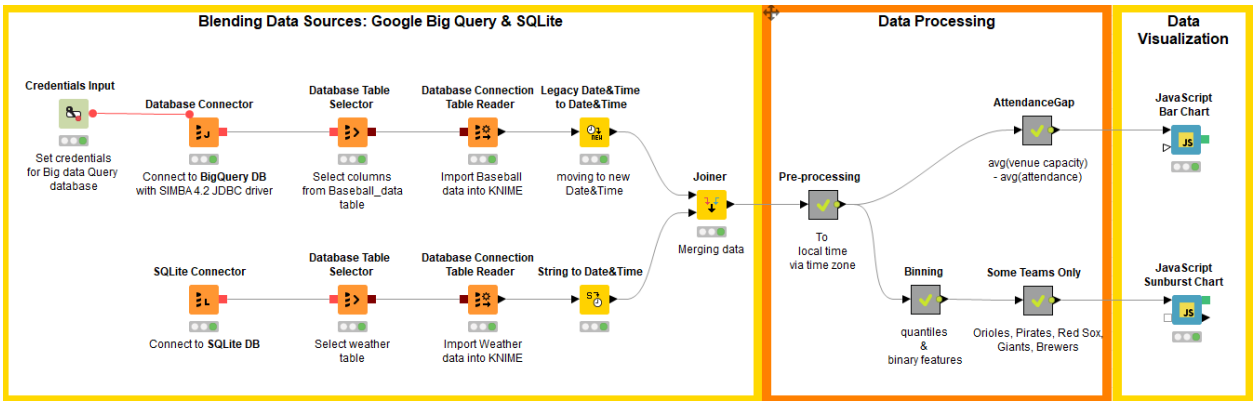
Note. With the latest version of KNIME Analytics Platform 3.4, new more flexible date and time column types have been introduced, representing just date, just time, or date&time, possibly with the associated time zone information.

However, for now, the DateTime fields in a database are imported into KNIME Analytics Platform using the old legacy DateTime column type. Therefore, after the data extraction, we need to convert such dates to the more recent Date&Time column type.

The “Legacy Date&Time To Date&Time” node converts the DateTime objects imported from Google Big Query into the new KNIME Date&Time type. The “String To Date&Time” node performs a similar conversion for the dates imported from SQLite.

Game attendance and weather data are joined together on the game date with a simple Joiner node using the inner join function. The full workflow is shown in figure 20.3.

Figure 20.3. – The final workflow merges attendance data from Google Big Query and weather data from SQLIte, pre-processes the resulting dataset, and visualizes attendance and the influence of the weather on attendance. Workflow is available on the EXAMPLES server in *01_Data_Access/02_Databases/10_GoogleBigQuery_meets_SQLite*



Inside the Pre-Processing metanode, the temperature difference between temp_high and temp_low is calculated. There the game time segment is also extracted, meaning afternoon, evening, or night. Such time segments are created using the hour information. Since all times in the database are expressed in UTC, the appropriate time zone has been assigned to each city, and the Date&Time objects have been shifted accordingly to express the local time using two Modify Time Zone nodes.

4. *Data Visualization*

First question: which team fills their stadiums most of the time? I am sure you have a guess, but let’s compare attendance numbers and stadium capacity.

In the metanode named AttendanceGap we calculate:

- the average attendance by HomeTeamName,
- the stadium fulfillment ratio (named AttendanceGap) as:

```
(stadium capacity - average attendance)/stadium capacity
```

Average attendance and stadium capacity are then sorted by decreasing AttendanceGap and represented side by side in a bar chart (Fig. 20.4) with a Javascript Bar Chart node.

Now to the second set of questions. Does weather influence game attendance? If yes, which weather conditions have the highest impact?

In order to visualize all these factors, we use a sunburst chart. Sunburst charts though do not represent numerical values, however, just ranges or nominal values. So, here we need to prepare the data accordingly.

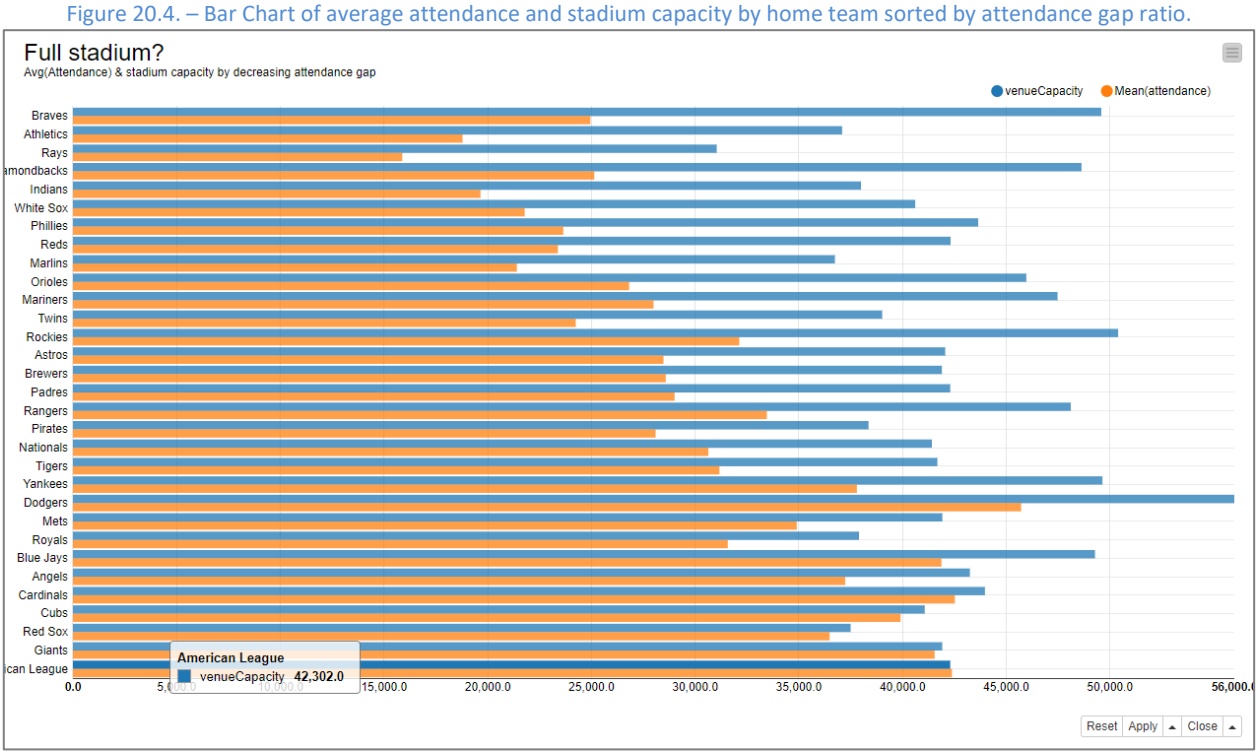
In the metanode named “Binning”, a Missing Value, an Auto-Binner, a Rule Engine, and a String Manipulation node transform our numerical data into quantile intervals and our binary data, like Rain =0/1, into String values, like “Rain”/”no Rain”. In particular, the Rule Engine node groups the games in ‘Low’, ‘Medium Low’, ‘Medium High’ and ‘High’ attendance categories and the Math Formula node calculates the game attendance ratio as attendance/stadium capacity.

In order to make the final chart easier to inspect, we concentrate on only 5 home teams: Orioles, Pirates, Giants, Red Sox, and Brewers. The Javascript Sunburst Chart node ends this branch of the workflow.

The Results

The bar chart shows the average attendance and the stadium capacity for each one of the home teams, sorted by attendance gap ratio.

Figure 20.4. – Bar Chart of average attendance and stadium capacity by home team sorted by attendance gap ratio.

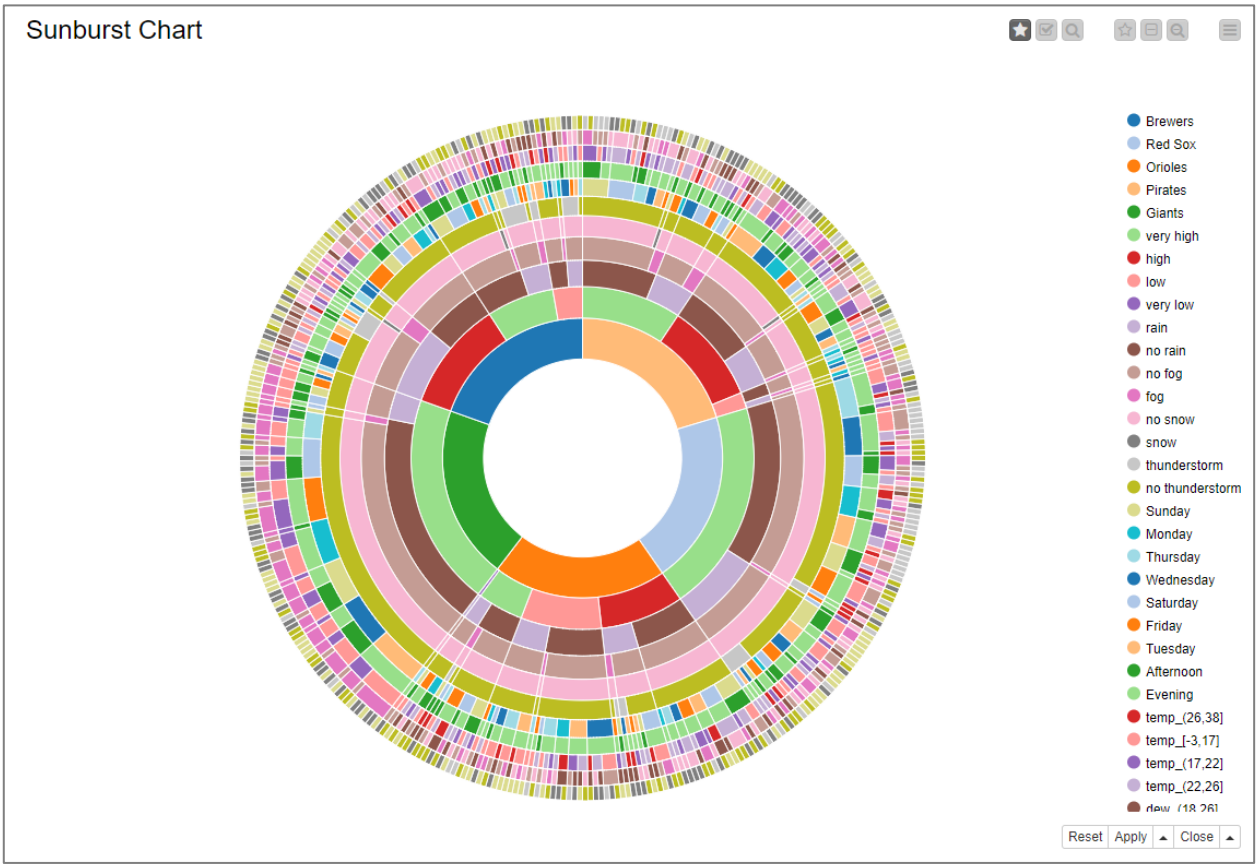


It looks as if only the American League, the Giants, the Red Sox, the Cubs, and the Cardinals can count on a large number of committed fans to fill their stadiums on a regular basis.

On the opposite the Braves, the Athletics, the Rays, and the Diamondbacks seemed to lack some support from their fans in the 2016 season.

Let's have a look now at the sunburst chart. The chart is organized in a number of concentric circles, each circle covering the value of a given input feature. In figure 20.5 we see the home teams in the most internal circle, then the attendance level, then rain, fog, snow, and thunderstorm presence or absence, the days of the week, the time segments, and finally the temperature, dew, and humidity intervals.

Figure 20.5. – Sunburst Chart for Orioles, Brewers, Red Sox, Pirates, and Giants on game attendance, weather conditions, weekdays, and day segments.



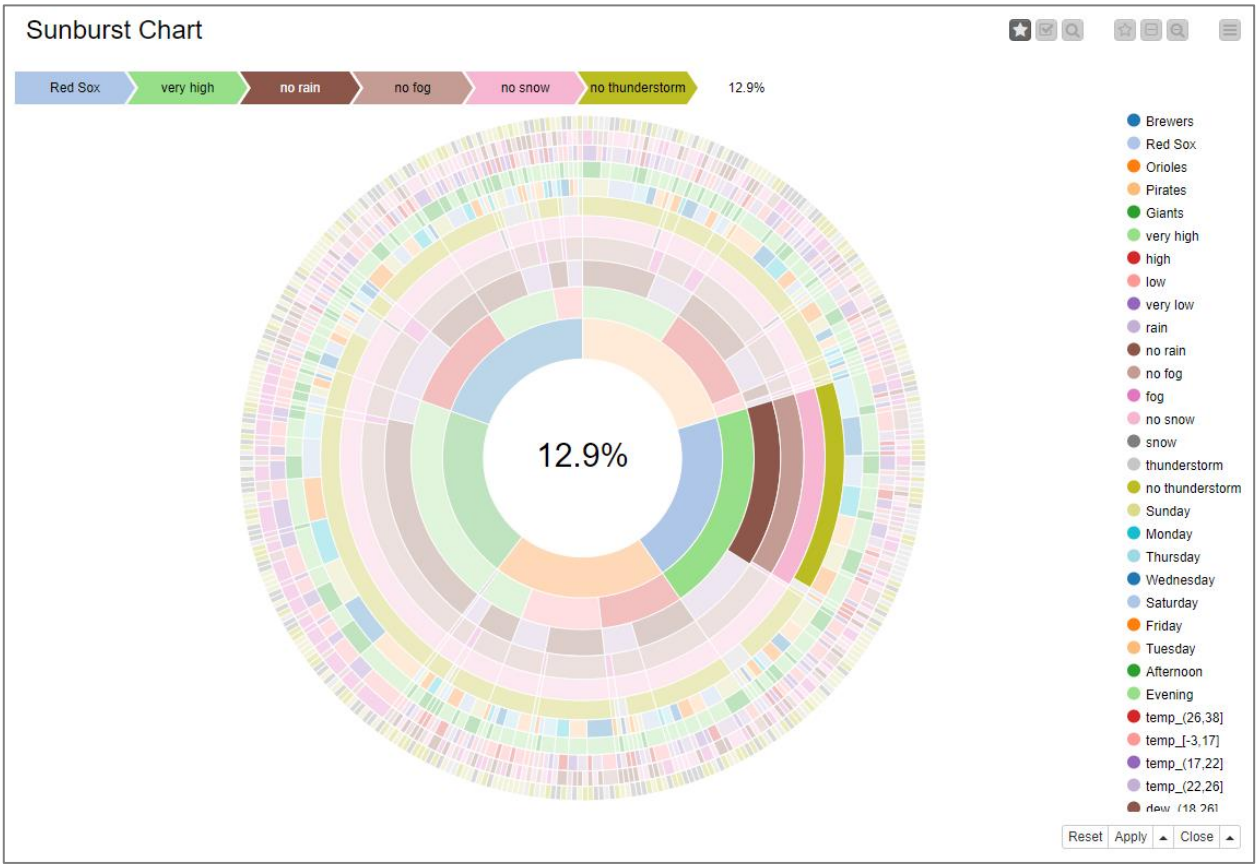
The sunburst chart allows for focus and selection modes. In focus mode, you can just mouse over and explore the size of the different subsets in the dataset.

For example, in perfect weather conditions (no rain, no snow, no thunderstorm, no fog), we have a very high attendance for the Red Sox. This segment accounts for almost 13% of the records in the dataset. If you inspect the chart further in focus mode, however, you discover that the Red Sox always have very high attendance, no matter what the weather conditions are.

If we continue our exploration, we see that the Orioles have had games with low attendance. However, here it does not look as if the weather conditions are predominantly different from the highly attended games. From a calendar point of view, though, Orioles games on Wednesday evenings suffer the most from lack of participation.

Brewers' games in adverse weather conditions, such as rain or thunderstorm, were not as well attended as other games in better weather conditions.

Figure 20.6. – Sunburst Chart. Isolating the subset of Red Sox games with very high attendance, no rain, no fog, no snow, and no thunderstorm. This subset accounts for 12.9% of all records.



If we take the time to explore it, the sunburst chart exposes a large amount of information organized across the selected input features, potentially showing unsuspected co-occurrences.

The workflow used for this blog post is available as usual on the KNIME EXAMPLES server under: EXAMPLES/*01_Data_Access/02_Databases/10_GoogleBigQuery_meets_SQLite*

21. SparkSQLmeets HiveQL. Women, Men, and Age in the State of Maine

Authors: Rosaria Silipo and Anna Martin, KNIME
Workflow in: EXAMPLES/*10_Big_Data/02_Spark_Executor/07_SparkSQL_meets_HiveQL*
Posted on: December 11, 2017

The Challenge

After seeing the foliage in Maine, I seriously gave a thought of moving up there in the beauty of nature and in the peace of a quieter life. I then started doing some research on Maine, its economy and its population.

As it happens, I do have the sampled demographics data for the state of Maine for the years 2009-2014, as part of the CENSUS dataset.

I have the whole CENSUS dataset stored on a Hive installation on a Cloudera cdh5.8 cluster running on the Amazon cloud. It could then be processed on Hive or on Spark using the KNIME Big Data Extension.

News!!! ***KNIME Big Data Extension*** has been ***open sourced*** with the last release of KNIME Analytics Platform 3.5. All Big Data nodes in the Node Repository now require no license to run. Check the "What's new in KNIME 3.5" page for more details on the new release.

KNIME Big Data Extension offers a variety of nodes to execute Spark or Hive scripts. Hive execution relies on the nodes for in-database processing. Spark execution has its dedicated nodes. However, it also provides an SQL integration to run SQL queries on the Spark execution engine.

We set our goal here to investigate the age distribution of Maine residents, men and women, using SQL queries. On Hive or on Spark? Why not both? We could use SparkSQL to extract men's age distribution and HiveQL to extract women's age distribution. We could then compare the two distributions and see if they show any difference.

But the main question, as usual, is: will SparkSQL queries and HiveQL queries blend?

Topic. Age distribution for men and women in the US state of Maine

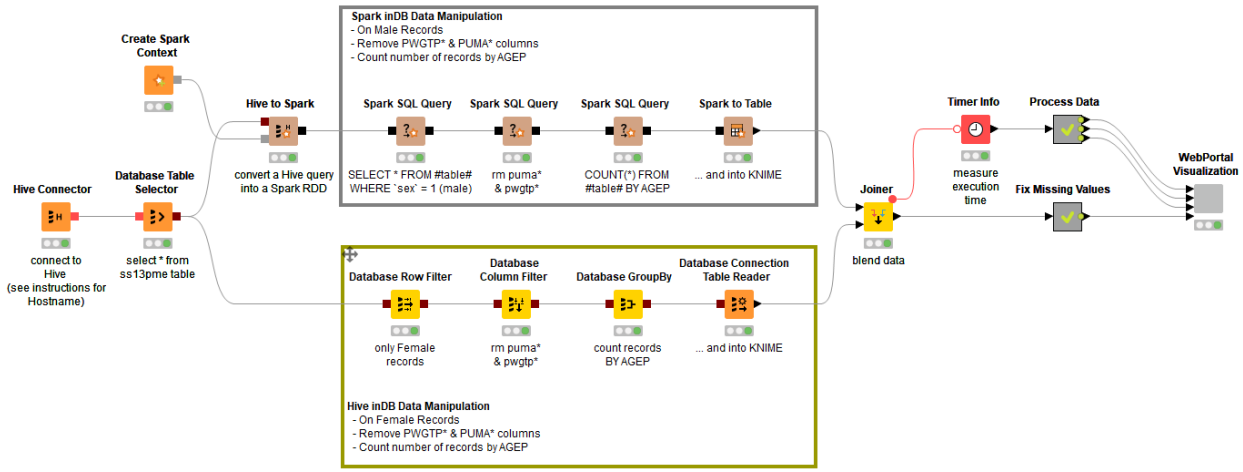
Challenge. Blend results from Hive SQL and Spark SQL queries.

Access Mode. Spark and Hive nodes for SQL processing

The Experiment

To explore age distributions of women and men living in Maine, we designed a workflow with two branches: the upper branch aggregates the age distribution of men using Spark SQL; the lower branch aggregates the age distribution of women using Hive SQL.

Figure 21.1. This workflow accesses a Hive database, extracts the CENSUS data for Maine residents, processes female records on Hive and male records on Spark, blends the results, and visualizes the two age distributions of men and women.



1. *Connecting to Hive*

Table ss13pme, stored in a Hive database, contains the CENSUS data of 1% of the Maine population over the 5 years following 2009, a total of circa 60k records. The workflow starts by connecting to the Hive database with a Hive Connector node followed by a Database Table Selector node.

At this point, the data flow follows two separate paths: one path will work on women's records and one path on men's records; one path will work with Spark SQL queries and one path with Hive SQL queries.

2. *Spark SQL queries to process men's records*

In the upper branch of the workflow, data records are processed using the Spark SQL integration. First of all, we need to create a Spark context and a Spark DataFrame for the data sitting on Hive. This is achieved by the Create Spark Context node and the Hive to Spark node respectively.

Then we build the SQL query with the help of a few Spark SQL nodes. A generic Spark SQL Query node extracts all men's records (sex = 1); then another generic SQL Query node removes less important columns; finally the last Spark SQL Query node counts the number of records by age.

Each Spark SQL node returns an additional Spark Dataframe containing the SQL instruction, but not the resulting data. When the Spark to Table node at the end of the sequence is executed, the Spark engine executes all DataFrames and the node returns the final result into a KNIME data table.

3. *Hive SQL queries to process women's records*

In the lower branch the data continue to flow on the Hadoop Hive platform. Again, here an SQL query extracts women's records (sex=2), removes unimportant columns, and counts records by age. The assembling of the full SQL query is obtained with a Database Row Filter node, a Database Column Filter node, and a Database GroupBy node.

At the end, the Database Connector Table Reader node executes the SQL query on the platform selected by the Database Connector node - i.e. the Hadoop Hive platform where the CENSUS data has been stored – and exports the results into a KNIME data table.

Note. The sequence of Database yellow nodes in the lower branch only builds the required SQL query string, but does not execute it. It is the final node of the sequence – Database Connector Table Reader– that executes the query and gets the job done.

Similarly, the sequence of Spark SQL nodes in the upper branch only builds the sequence of required DataFrames without executing them. It is the final node of the sequence – Spark to Table – that executes the DataFrames and gets the job done.

4. *Data Blending & Visualization*

The blending of the results of the execution of the two SQL queries is carried out by the Joiner node inside KNIME Analytics Platform, which joins men and women counts on age values.

Aggregated data have smaller size. Therefore, after aggregation, transfering the results into KNIME Analytics Platform and joining them it is not a problem. Of course, the join operation could also be run on Hive or on Spark.

In order to cover possible age holes in the original data, the demographics table is left-joined with an ad-hoc table including all ages between 0 and 100 in the Fix Missing Values metanode.

The final node of the workflow, named WebPortal Visualization, produces a line plot for both age distributions through a JavaScript Line Plot node (Fig. 21.2).

Note. Packing the JavaScript visualization node into a wrapped metanode automatically makes the plots also visible and controllable from the KNIME WebPortal.

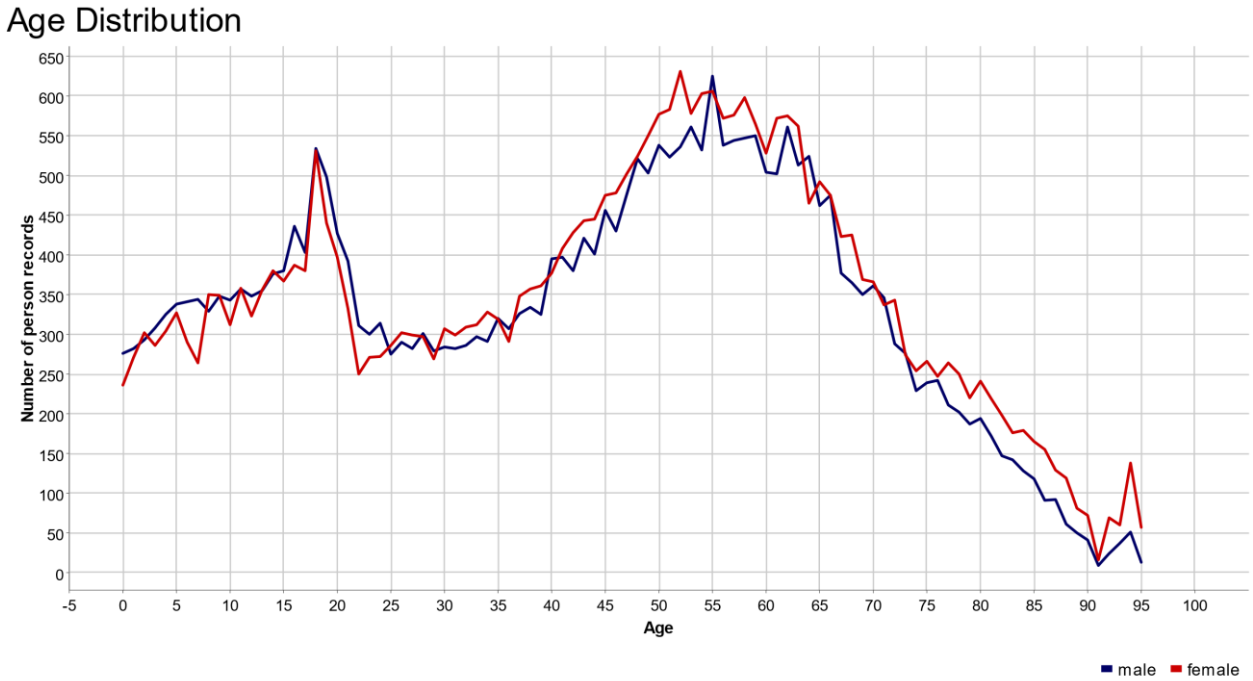
The whole workflow, calculating the age distribution for men through Spark SQL and the age distribution for women through Hive SQL, is displayed in figure 21.1 and is downloadable from the KNIME EXAMPLES server under *10_Big_Data/02_Spark_Executor/07_SparkSQL_meets_HiveQL*.

The Results

Maine is one of the smallest states inside the USA. Its economy has not been blooming for decades. It is not really a place where people move for a career advancement step, but rather for studying or retiring.

The WebPortal interactive page displaying the two age plots is shown in figure 21.2. On the y-axis we have the absolute number of people; on the x-axis the corresponding age values.

Figure 21.2. Age distribution for men (blue) and women (red) in the state of Maine. The distributions have been estimated on the CENSUS dataset for Maine, representing 1% of the Maine population between 2009 and 2014 for a total of circa 60k records.



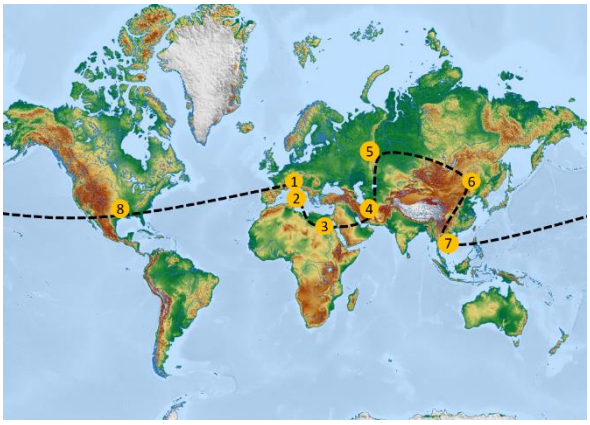
In the line plot above, women (in red) and men (in blue) are similarly distributed in age. No major differences are observed between the two distributions of men's and women's population in the State of Maine.

Two peaks are clearly identifiable: one covering the ages between 17 and 22 (students) and a larger plateau covering the ages between 50 and 70 (retired). People in working age, such as between 24 and 45 year old, are less prominent in both age distributions. This might be related to the lack of jobs in the area.

But the most important conclusion is: Yes, they blend! Spark SQL and HiveQL SQL scripts can be used together in a KNIME workflow and their results do blend!

22. Chinese meets English meets Thai meets German meets Italian meets Arabic meets Farsi meets Russian. Around the World in 8 Languages

Authors: Anna Martin, Hayley Smith, and Mallika Bose, KNIME
Workflow in: EXAMPLES/08_Other_Analytics_Types/01_Text_Processing/20_BlendLanguagesInTagCloud
Posted on: February 19, 2018



The Challenge

No doubt you are familiar with the adventure novel "Around the World in 80 Days" in which British gentleman Phileas Fogg makes a bet that he can circumnavigate the world in 80 days. Today we will be attempting a similar journey. However, ours is unlikely to be quite as adventurous as the one Phileas made. We won't be riding Elephants across the Indian mainland, nor rescuing our travel companion from the circus. And we certainly won't be getting attacked by Native American Sioux warriors!

Our adventure will begin from our offices on the Lake of Constance in Germany. From there we will travel down to Italy, stopping briefly to see the Coliseum. Then across the Mediterranean to see the Pyramids of Egypt and on through the Middle East to the ancient city of Persepolis. After a detour via Russia to see the Red Square in Moscow, our next stop will be the serene beaches of Thailand for a short break before we head off to walk the Great Wall of China (or at least part of it). On the way home, we will stop in and say hello to our colleagues in the Texas office.

Like all good travelers, we want to stay up-to-date with the news the entire time. Our goal is to read the local newspapers … in the local language of course! This means reading news in German, Italian, Arabic, Farsi, Chinese, Russian, Thai, and lastly, English. Impossible you say? Well, we'll see.

The real question is: will all those languages blend?

Topic. Blending news in different languages

Challenge. Will the Text Processing nodes support all the different encodings?

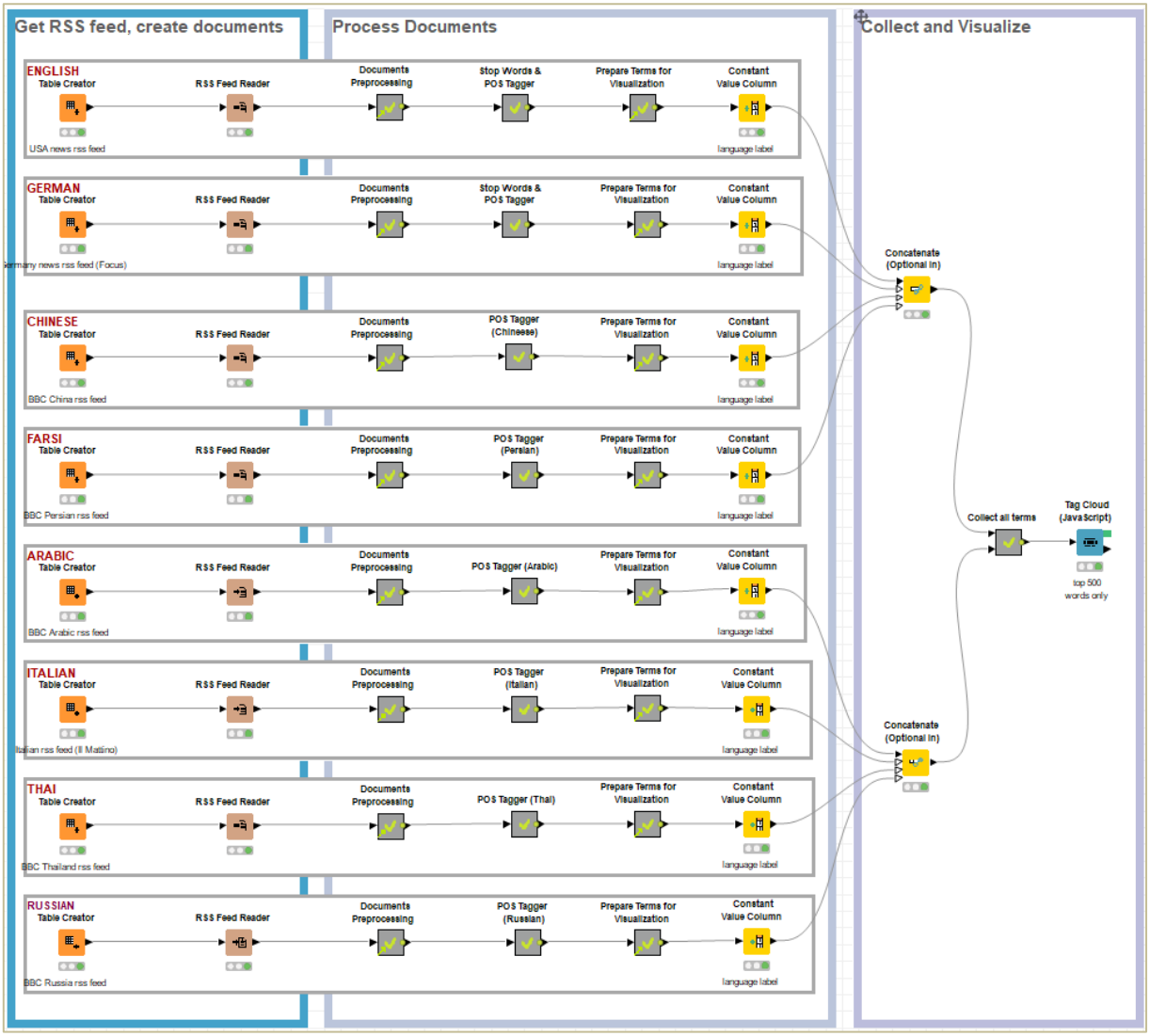
Access Mode. Text Processing nodes and RSS Feed Reader node

The Experiment

Retrieving News via RSS Feeds

To get the RSS news feeds, we used the RSS Feed Reader node. The RSS Feed Reader node connects to the RSS Feed URL, downloads the feed articles, parses them with the help of a tokenizer, and saves them in a document type column at the output port.

Figure 22.1. Final workflow connecting to several RSS feed URLs for news in eight different languages, pre-processing the articles, and creating a word cloud from the most frequent words. The workflow is available on the KNIME EXAMPLES server under 08_Other_Analytics_Types/01_Text_Processing/20_BlendLanguagesInTagCloud. The workflow comes in a folder which also contains two metanode templates necessary to the workflow execution.



It is important to notice that the text parsing operation performed by the RSS Feed Reader node is language dependent. A few language specific tokenizers are embedded into the node, such as for English, Spanish, German, and Chinese. However, for most of the languages, no specific tokenizer is available in the RSS Feed Reader node, nor in the other Text Processing nodes. In these cases, the two basic “Open NLP Simple Tokenizer” and “Open NLP Whitespace Tokenizer” might do.

Then, for each language, a Table Creator node with URLs of the local news, is fed to the RSS Feed Reader node. You can see the different language branches in the final workflow in figure 22.1.

Document Pre-Processing

Document pre-processing mainly involves clean-up tasks such as: keep “Document” column only, remove all numbers, and erase punctuation signs.

Since those operations are language independent, the same metanode can be used for all languages. Not only eight replicas of the same metanode, but eight links to one single metanode template!

Note. The adoption of one metanode template makes maintenance much easier than using eight replicas.

To create a metanode template, just right-click the metanode in the workflow and in the context menu select "Metanode" -> "Save as Template", then select the folder destination in the KNIME Explorer panel, and click "OK". The metanode will be saved as a template and the current metanode in the workflow will be transformed into a link to that template. Now, just drag and drop the metanode template into the workflow editor to create a new link to that metanode.

Each time the metanode template is changed, you have the chance to automatically update the currently linked metanode with the new content. Maintenance becomes much easier!

All metanodes named "Documents Preprocessing" in the final workflow are just links to the one and only metanode template, named "Documents Preprocessing". You can see that from the green arrow in the lower left corner of the metanode icon.

This link to a metanode template feature was made available in KNIME Analytics Platform version 3.4.

POS Tagging

Part of Speech (POS) tagging is the detection of the grammar role of each word in the text (noun, verb, adverb, adjective, and so on). The second metanode in each branch of the final workflow is named "POS Tagger". This metanode includes a node for stop word filtering, one or more nodes for POS tagging, and a Tag Filter node. All operations are language dependent, since both match words in the text against words in one or more dictionaries. For this reason, each POS Tagger metanode has been customized to work with the branch-specific language.

The Stop Word Filter node includes built-in stop word dictionaries for English, French, Spanish, Italian, Russian, and a few other languages. For these languages, we can then select the appropriate built-in dictionary. For the remaining languages (or if we are not satisfied with the current built-in list), we need to provide an external file containing the list of stop words we would like to eliminate. This was the case for Farsi, Arabic, Russian, and Thai.

Different nodes implement this operation. The POS Tagger node, for example, implements a POS schema for the English language. The Stanford Tagger node implements a POS Tagger for English and other languages such as German. Different languages use different POS models. For example, the Stanford Tagger for the German language relies on the STTS schema (Stuttgart-Tübingen-TagSet).

There was no POS Tagger for most of the languages considered in this blog post. In these cases, we created our own custom POS Tagger metanode. First, we retrieved a POS dictionary file for the language of interest, with *word-POS-tag* pairs, associating words to their grammar role tag. Finding such a dictionary is not always easy however, linguistic departments of local universities may offer some resources. Once we had a POS dictionary file, we associated the words with their specific POS tag using a Dictionary Tagger node. We began with nouns (NN), followed by adjectives (JJ), then verbs (VB), and so on. Thus, one of the possible implementations requires looping on all lists of words (all verbs, all nouns, and so on), and tagging the words in the text accordingly. Since previous tags must not be forgotten, a recursive loop is required here (Figure 22.2).

Note. The recursive loop enables looping on the data set for a fixed number of times, or when an end condition is met. Most importantly it has memory, meaning, the output dataset of the current iteration is passed back to be the starting dataset of the next iteration.

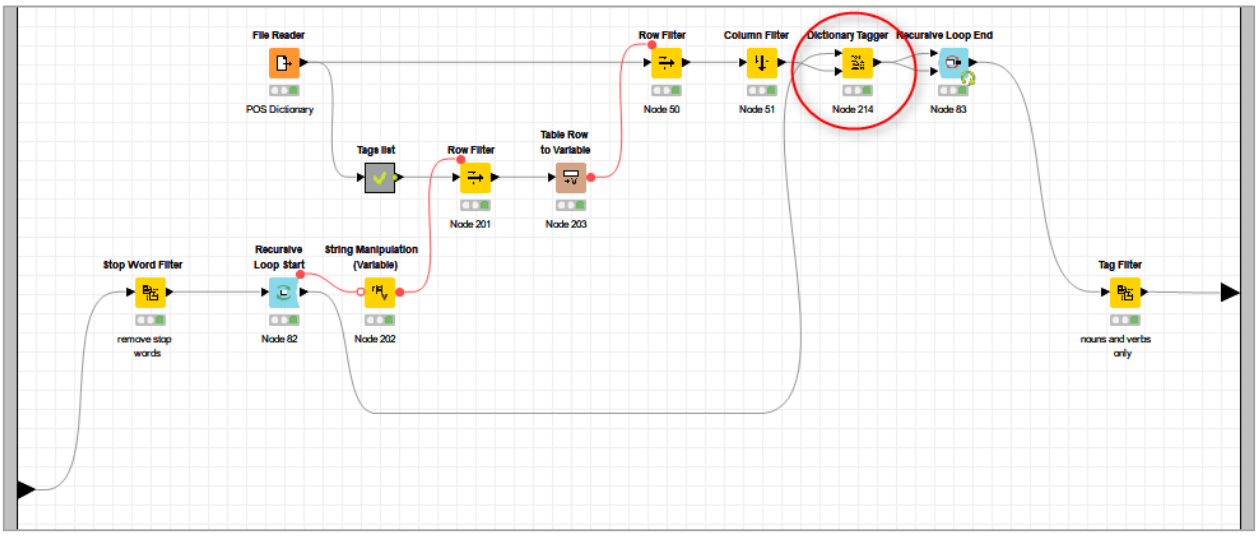
Please note, that proper POS tagging is a more complicated process than simply assigning tags from a list of words. The introduction of grammar rules together with the list of words in the POS dictionary would produce a more accurate result than what was implemented in this workflow. The basic tagging implemented here might however be useful, even if it is not entirely accurate. The quality of this basic POS tagging operation highly depends on the quality and the number of entries in the POS dictionary.

Of all tagged words, we only kept nouns and verbs using a Tag Filter node.

Note. The Tag Filter node is also language dependent, since different languages might use different POS schemas. For German, for example, the STTS schema (Stuttgart-Tübingen-TagSet) is used, instead of the POS schema used for English.

With this node, we conclude the language dependent part of our analysis.

Figure 22.2. Content of "POS Tagger" metanode for the Italian language. Notice the Dictionary Tagger node and the recursive loop.



Representation of Words and Frequencies in the Word Cloud

The next metanode, named "Prepare Terms for Visualization", extracts all words from the documents and calculates their relative frequencies using a Bag of Words Creator node and a Tag Filter node. Relative frequencies of the same terms are summed up throughout the list of documents. For each language, the top 100 terms with highest cumulative relative frequency are extracted.

The "Prepare Terms for Visualization" metanode also performs language independent operations and can be abstracted into a metanode template. In the final workflow, each branch shows a link to the same metanode template.

The final 800 words from all eight languages, are colored by language and displayed in an interactive word cloud using a Tag Cloud (JavaScript) node.

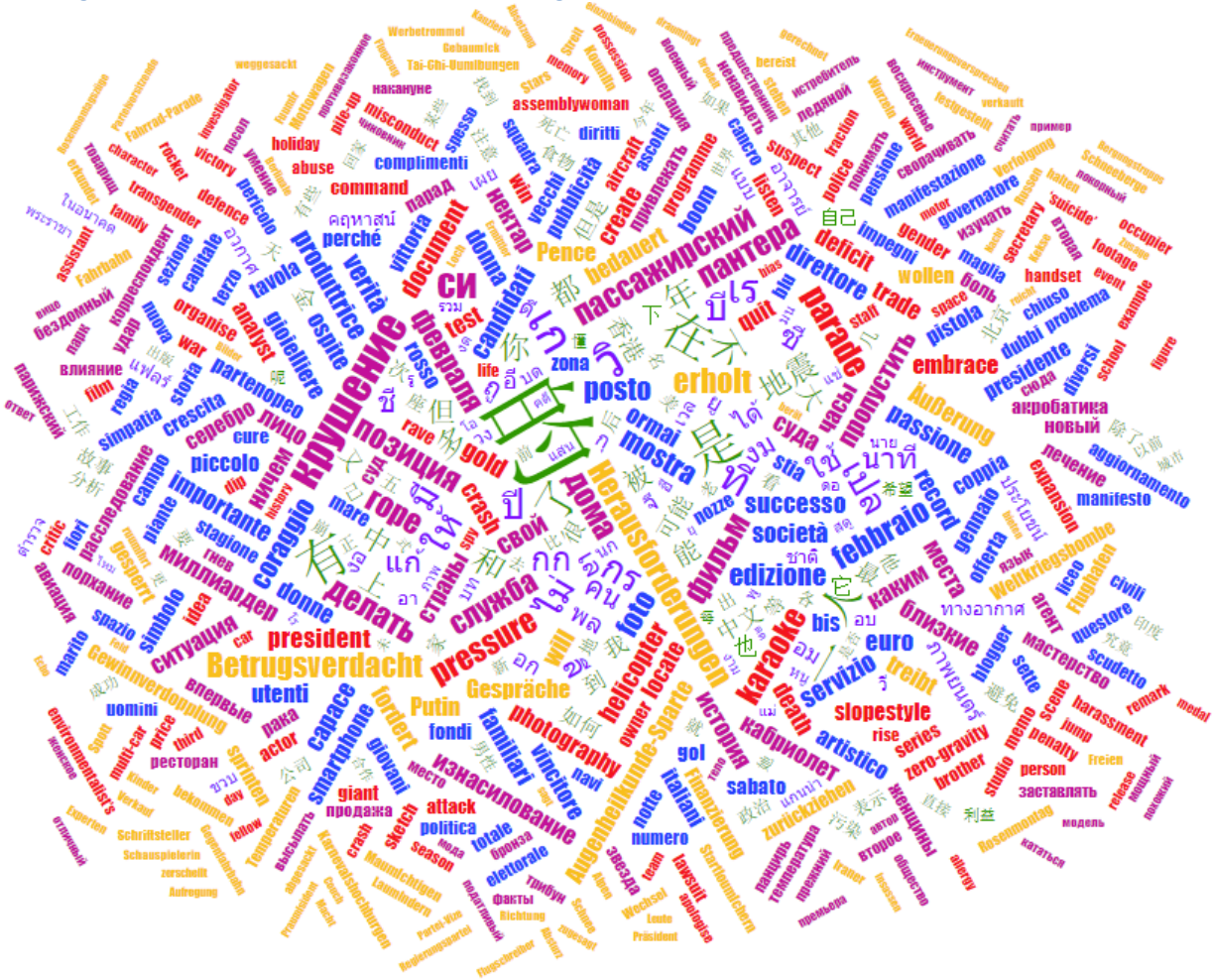
The final workflow (Figure 22.1) blends news in eight different languages: English, Italian, German, Chinese, Farsi, Arabic, Thai, and Russian. It is downloadable from the KNIME EXAMPLES server under: *08_Other_Analytics_Types/01_Text_Processing/20_BlendLanguagesInTagCloud*. Notice that the workflow comes in a folder containing also the two metanode templates.

The Results

Whilst we did manage to read newspapers in each of the eight countries we visited, the news, the languages, the different writing, the great memories are all still mixed up in our head, in a very similar way to the word cloud in Figure 22.3.

The word cloud view resulting from the Tag Cloud (JavaScript) node is interactive. Words can be selected and then extracted from the output data table, filtering all rows with values in the column "Selected (JavaScript Tag Cloud)" = true.

Figure 22.3. Word Cloud from the news feeds in English, German, Italian, Arabic, Farsi, Chinese, Russian, and Thai.



We can conclude that while the different languages and the different writing do not really blend in the word cloud, the workflow we built can extract and process several different languages and can easily be extended to process additional languages, regardless of the encoding they require.

Languages do not blend, news feeds … yes they blend!

If you want to get a taste of a Babel mix of many different languages, we encourage you to attend the KNIME Spring Summit 2018, (Berlin, March 5 – 9). The Summit is where KNIME users and enthusiasts, including leading data scientists, from all over the world come to meet and network. Despite the diverse number of spoken languages, all training courses, sessions, and workshops will be held in English.

List of resources used for this blog post

Custom Stop Word Files

- Chinese https://gist.github.com/dreampuf/5548203
- Farsi https://github.com/kharazi/persian-stopwords/blob/master/persian
- Arabic https://github.com/mohataher/arabic-stop-words/blob/master/list.txt
- Thai https://github.com/stopwords-iso/stopwords-th/blob/master/stopwords-th.txt
- Russian: https://github.com/stopwords-iso/stopwords-ru/blob/master/stopwords-ru.txt

Custom POS Word Files

- Chinese http://compling.hss.ntu.edu.sg/omw/
- Farsi http://compling.hss.ntu.edu.sg/omw/
- Arabic http://compling.hss.ntu.edu.sg/omw/
- Thai http://compling.hss.ntu.edu.sg/omw/
- Italian https://github.com/clips/pattern/tree/master/pattern/text/it
- Russian: http://wordnet.ru/

Acknowledgements

The creation of this blog post would not have been possible without the invaluable help of Mihály Medzihradszky and Björn Lohrmann, for the first inception of a workflow which could retrieve the news via RSS Feeder nodes. Also, Alfredo Roccato, for the implementation of text pre-processing and POS tagging, first for the Italian language and later for the other languages.

Node and Topic Index

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

WILL THEY BLEND?

The Blog Post Collection

Rosaria Silipo has been applying data-mining and predictive analytics techniques to big and small data in the corporate ecosystem for the last twenty years at least. She started with her master thesis at the University of Florence and she continued during her doctoral and post-doctoral positions as well as during most of her following professional positions.

Her extensive experience in predictive analytics across a wide range of business cases and industry sectors has led her to become the Principal Data Scientist at KNIME.com AG where she applies her knowledge to develop frontier studies, review classical use cases, and train the new generation of data scientists.
She is the author of more than 50 scientific publications, many scientific whitepapers, and a number of books, courses, and videos for data science practitioners.